CS11-711 Advanced NLP

PCFG Parsing

Hao Zhu

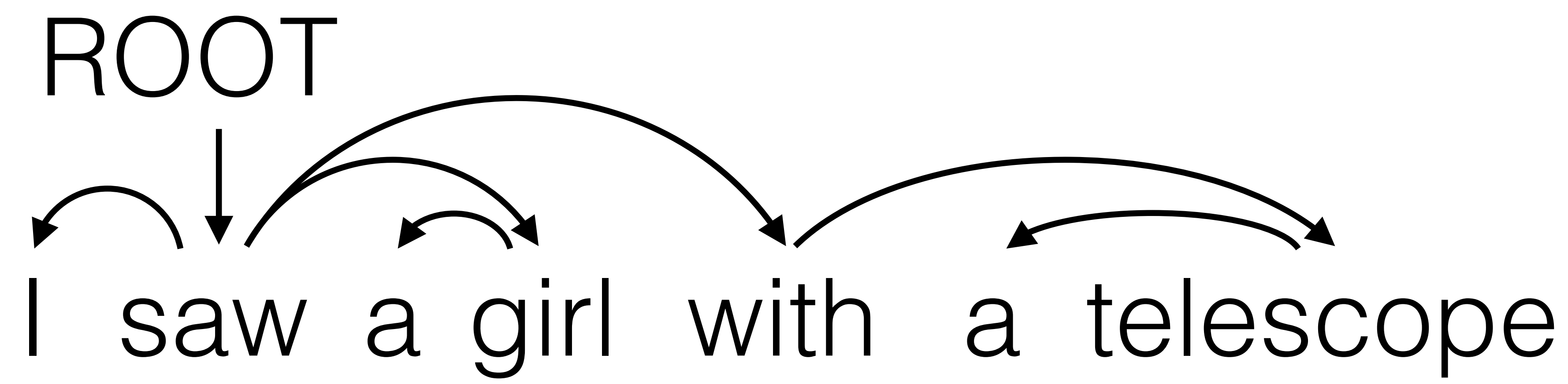
Carnegie Mellon University
Language Technologies Institute

Site

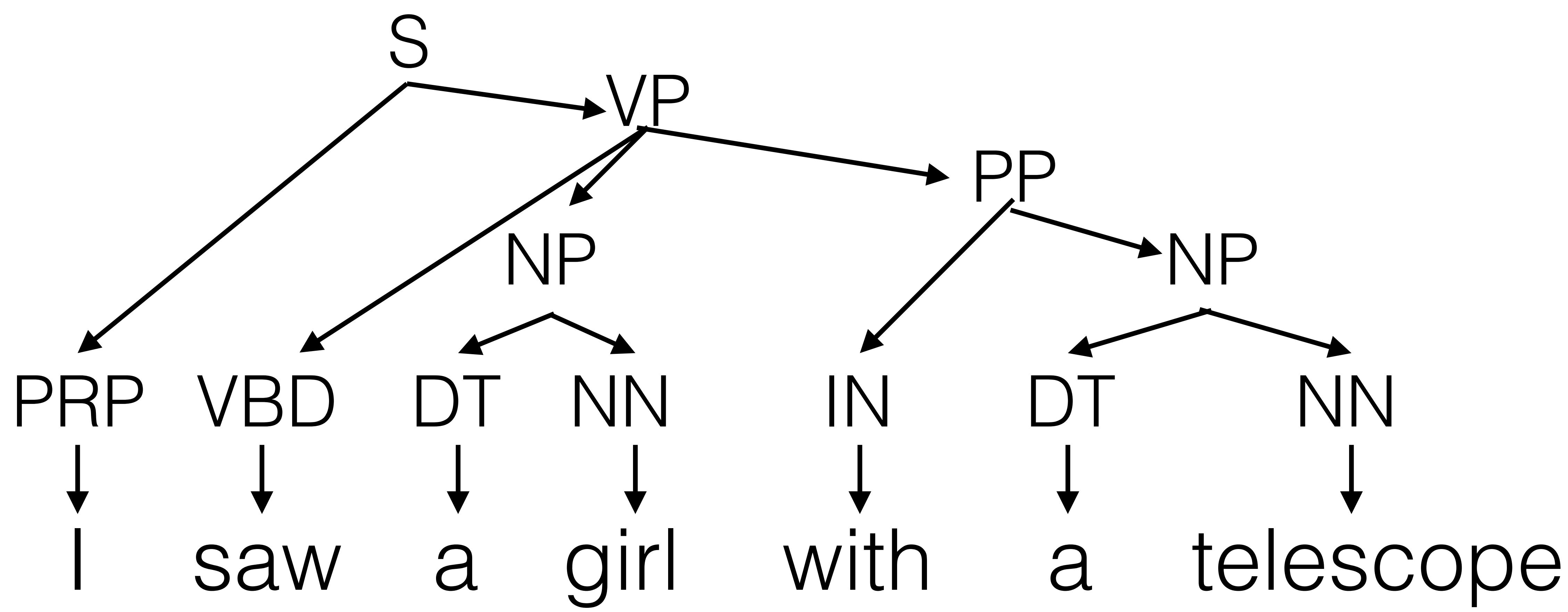
https://phontron.com/class/anlp2021/

Two Types of Linguistic Structure

- **Dependency:** focus on relations between words



- **Phrase structure:** focus on the structure of the sentence



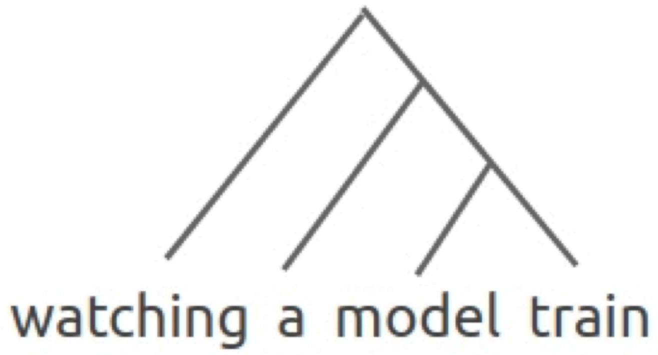
Grammar Induction (Unsupervised Parsing)

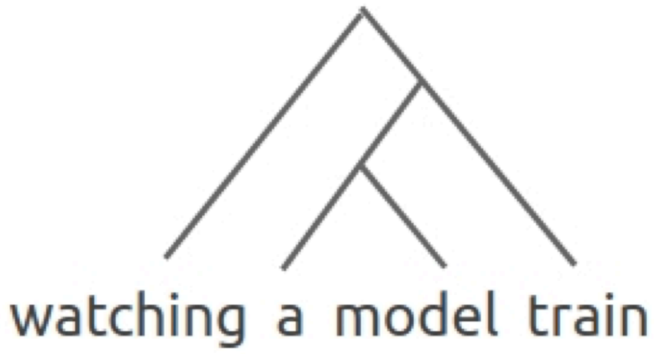
Learning a set of (probabilistic) grammar rules

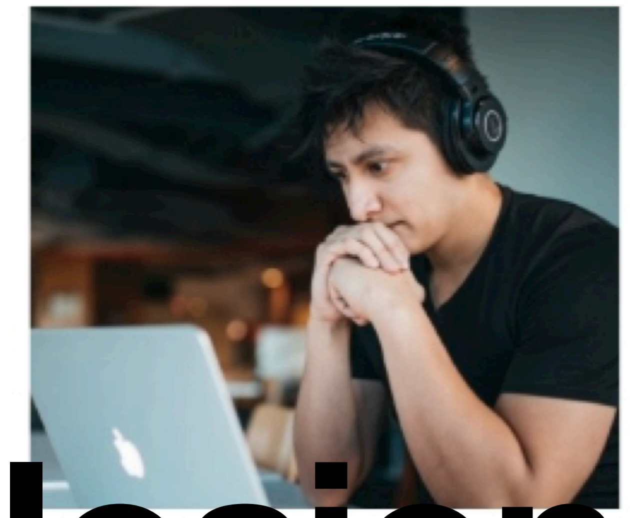
Typical grammar induction methods

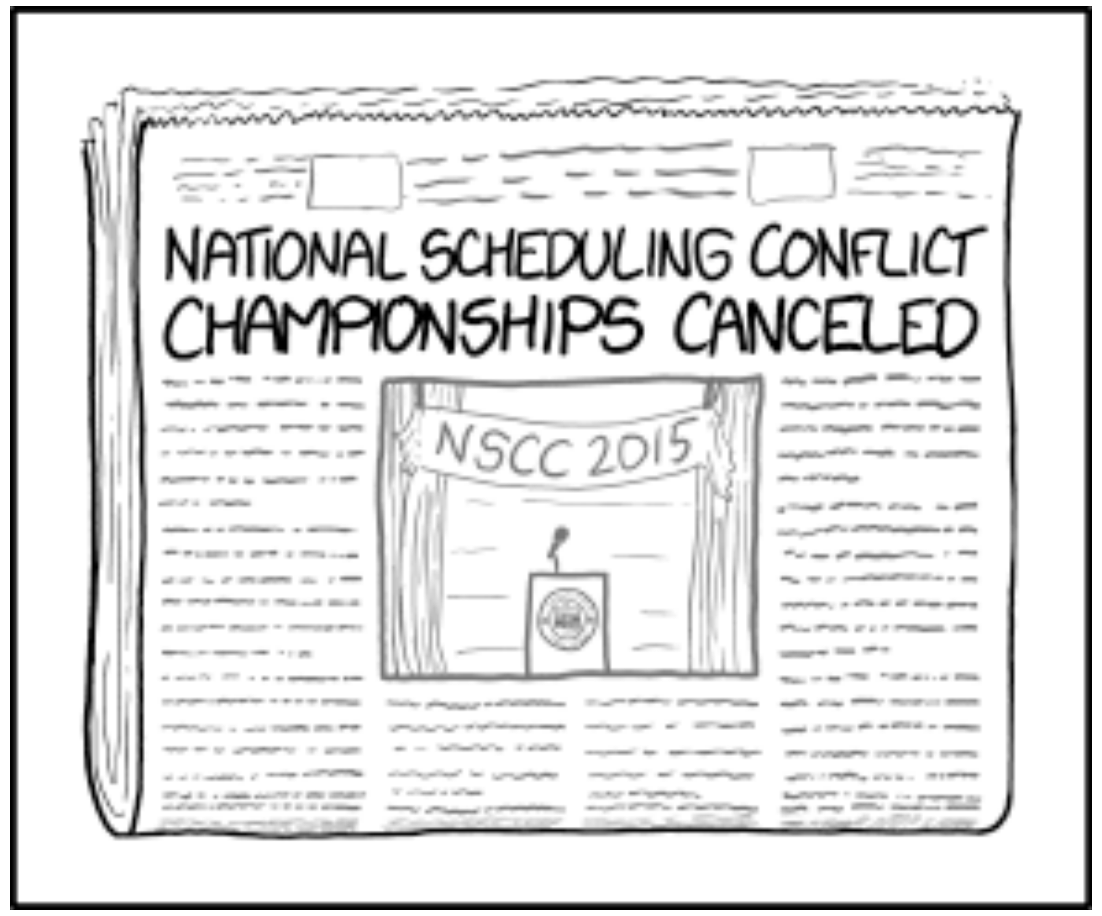
unsupervised constituency and dependency parsing

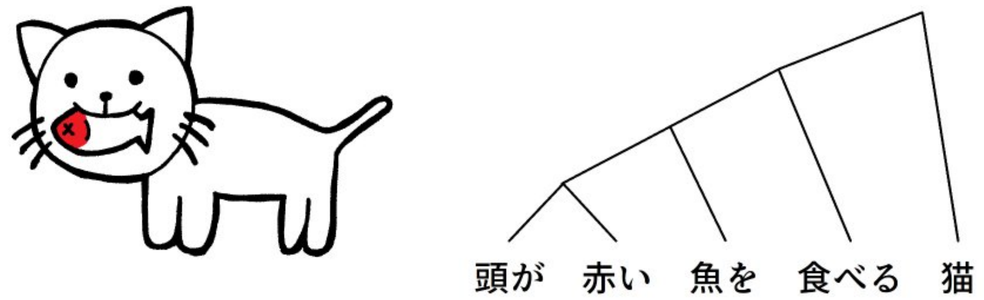
Explosion of ambiguity

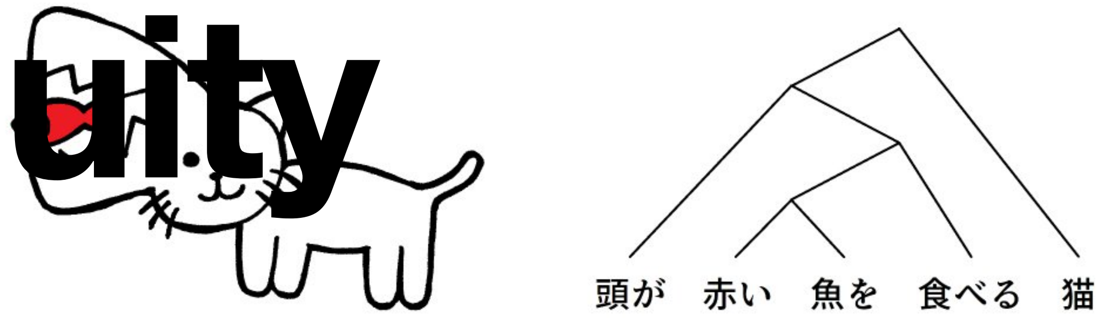


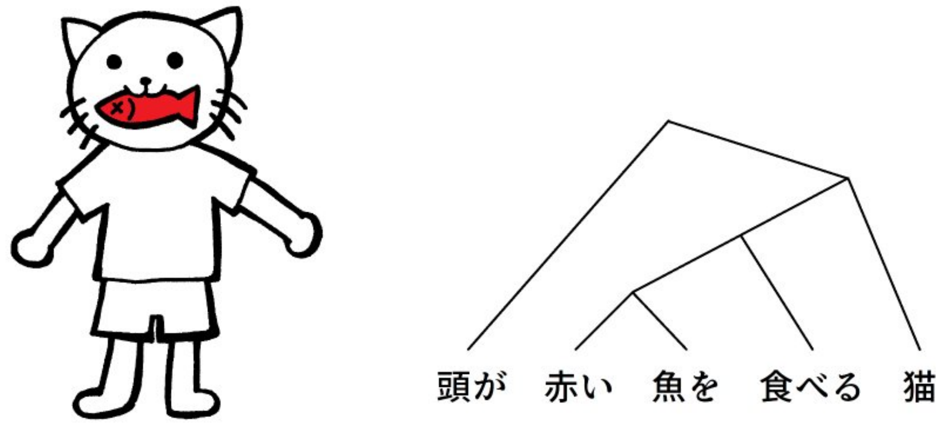


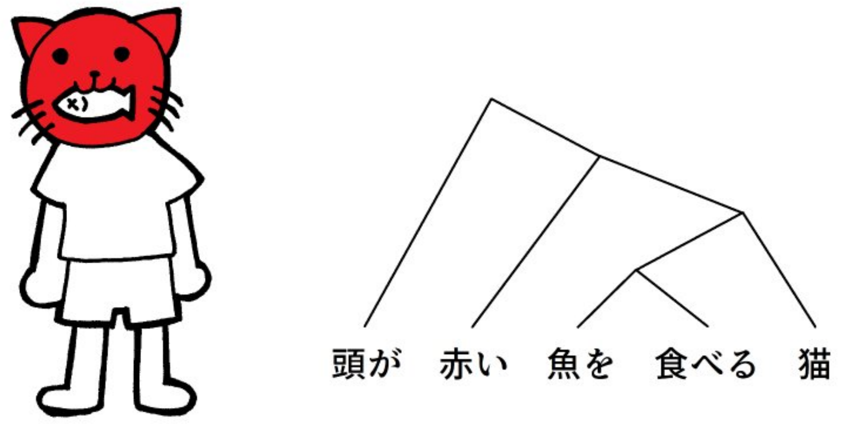






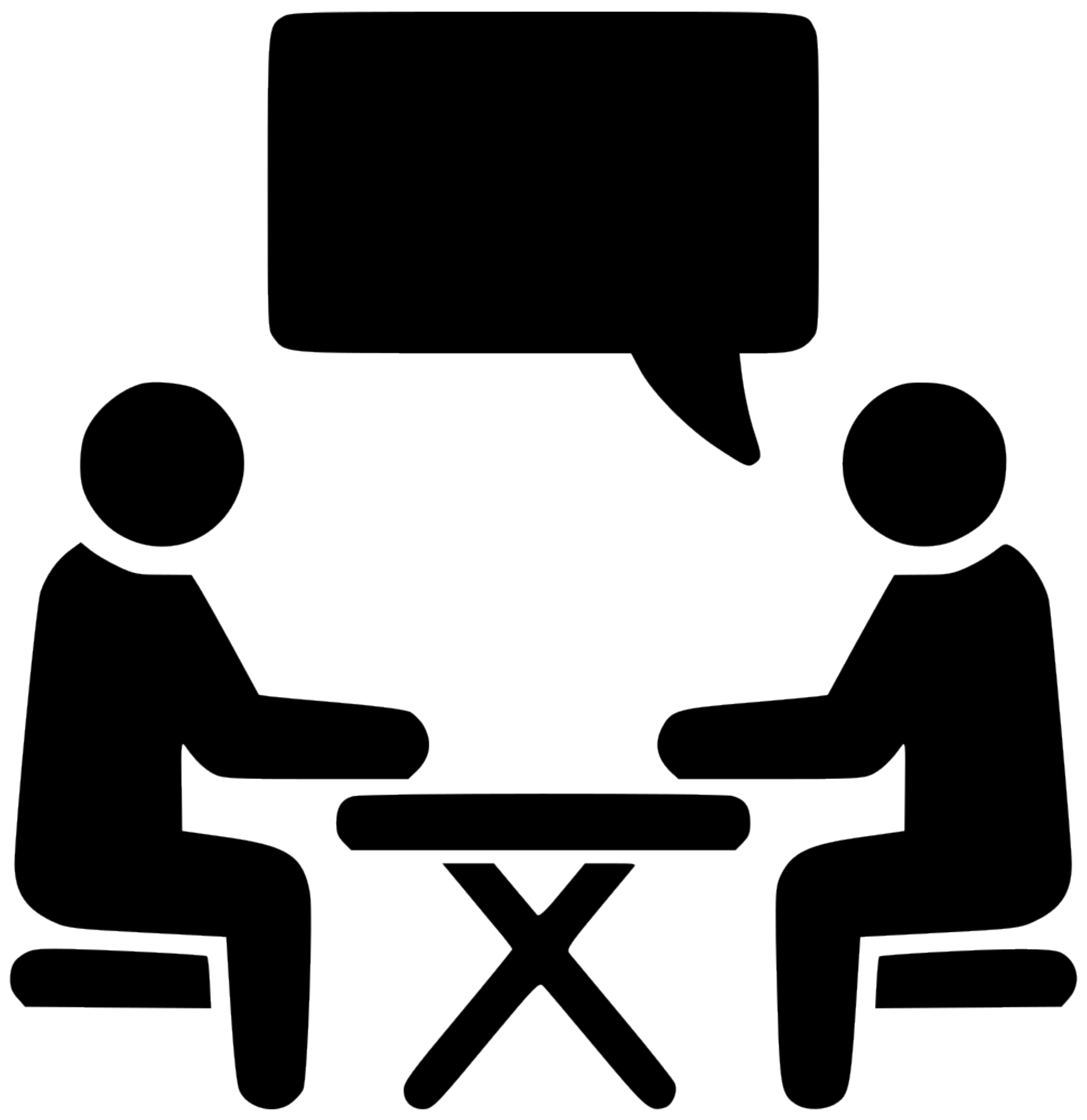




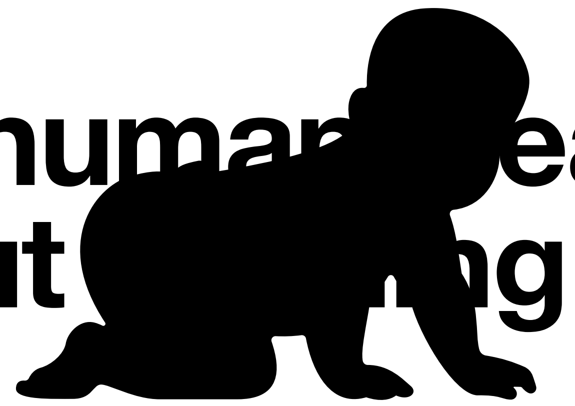


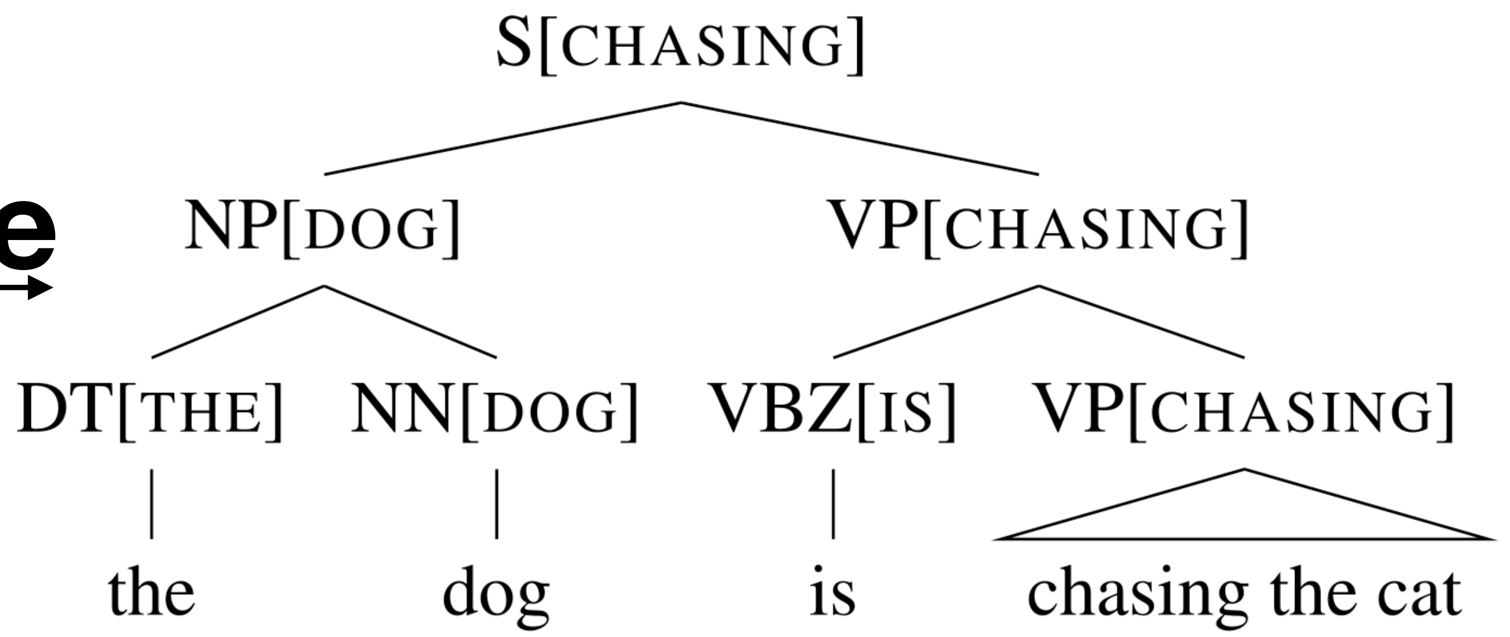
Probabilistic parsing

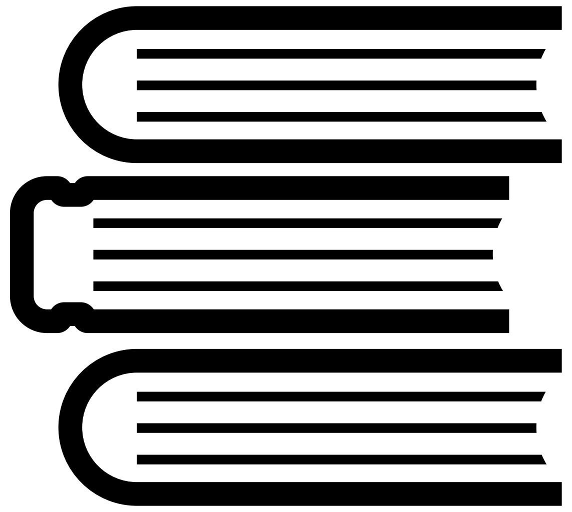
- ~~First try parsing without any weird rules, throwing them in only if needed.~~
- Better: every rule has a weight.
  - A tree’s weight is total weight of all its rules.
  - Pick the overall lightest parse of sentence.
- Best: train the weights!



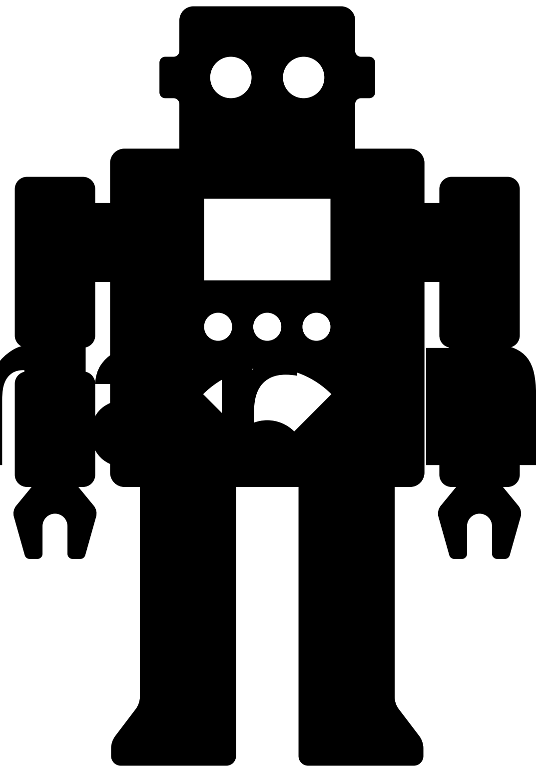
Mystery: hum earn to parse
without g to parse

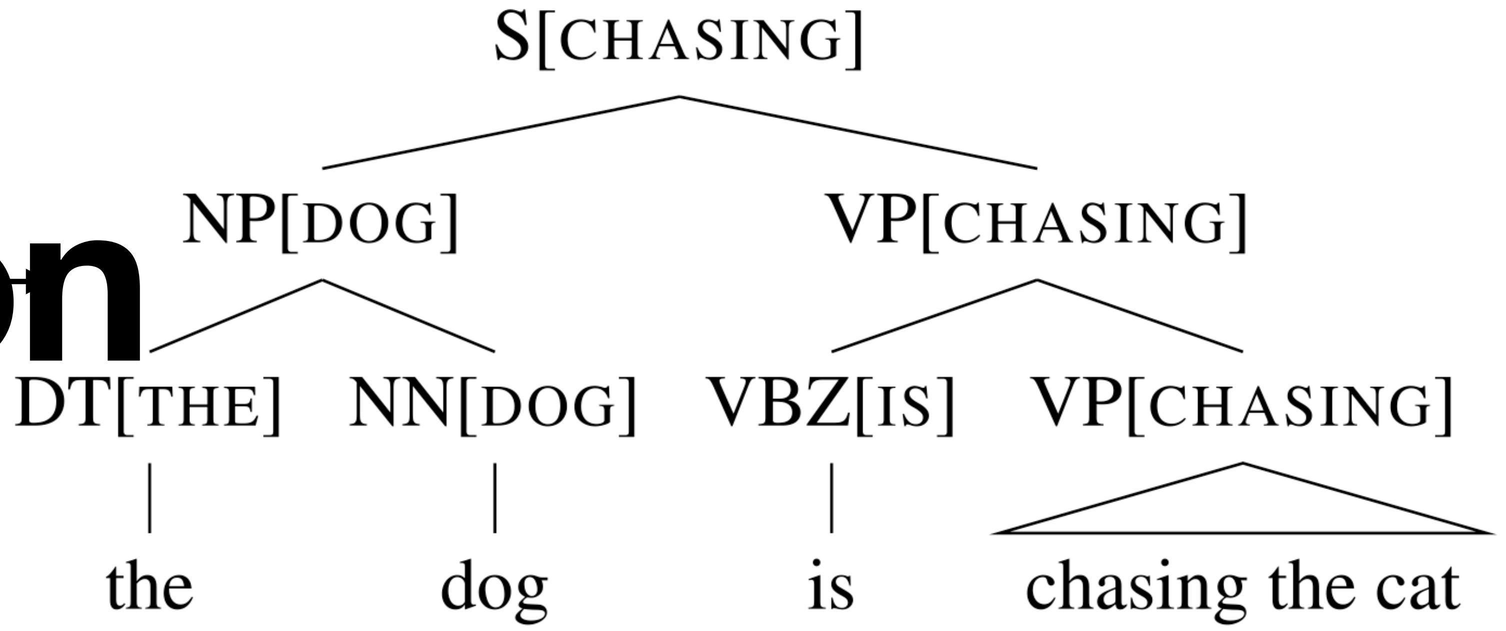






~~Grammar Induction~~





CFG definition

- $\mathscr{G} = (S, \mathscr{N}, \mathscr{P}, \Sigma, \mathscr{R})$
- $\mathscr{N}$: Set of nonterminals (constituent labels)
- $\mathscr{P}$: Set of preterminals (part-of-speech tags)
- Σ: Set of terminals (words)
- S: Start symbol
- $\mathscr{R}$: Set of rules

WLOG, only consider binary branching; Chomsky Normal Form

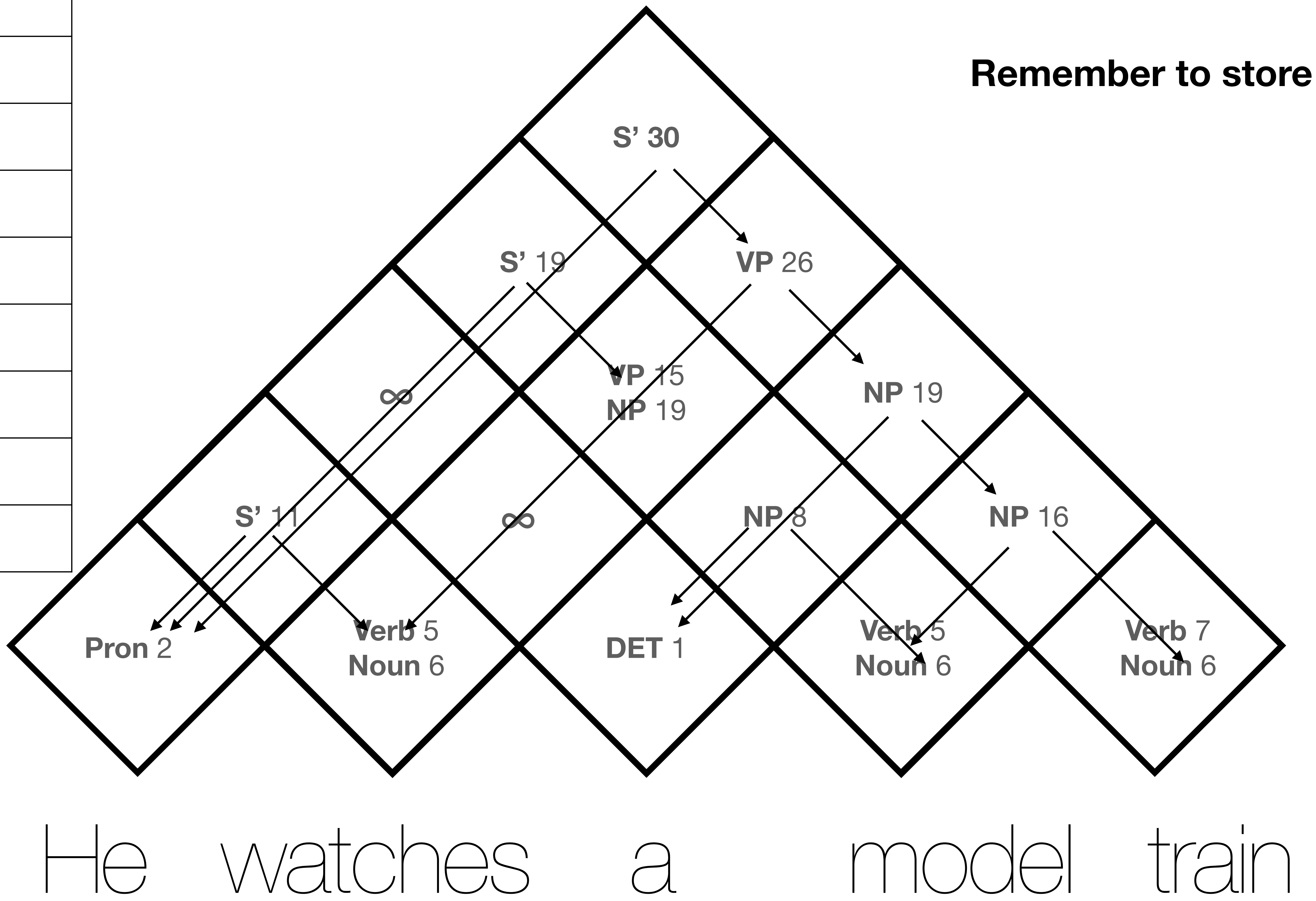
$$
\begin{aligned}
&S \rightarrow A, && A \in \mathcal{N} \\
&A \rightarrow BC, && A \in \mathcal{N}, B, C \in \mathcal{N} \cup \mathcal{P} \\
&T \rightarrow \alpha, && T \in \mathcal{P}
\end{aligned}
$$

Probalistic CFG

- For every rule, assign a probability to it.
- The summation of the probabilities of the rules with the same left hand non-terminal X is 1: $\sum_{Y} \pi(X \rightarrow Y) = 1$
- How to get the most probable tree given the probabilities of the rules?
    - Dynamic Programming

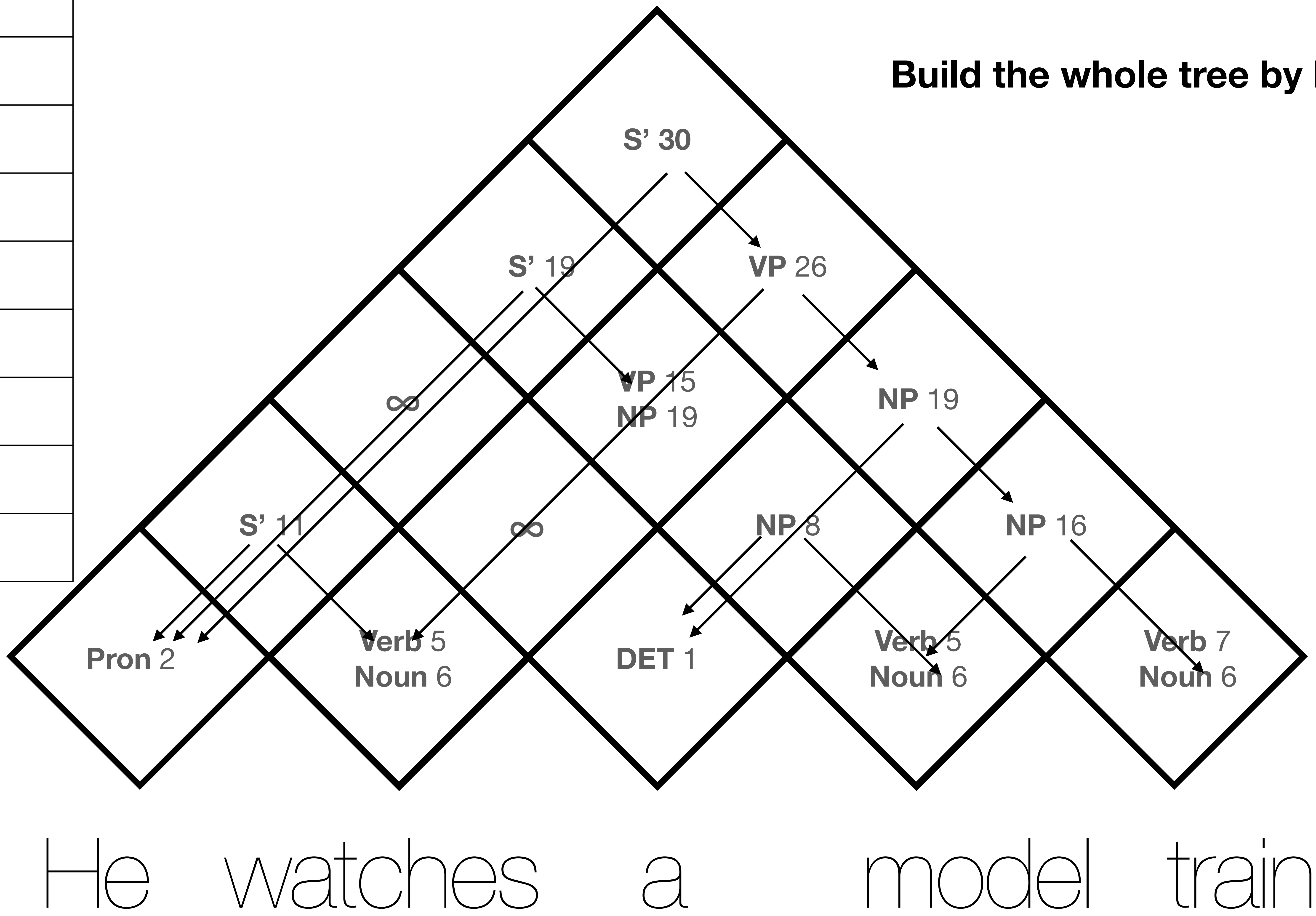
Rule	-log prob
S' →Pron Verb	4
S' →Pron VP	2
S' →NP VP	2
NP →NP Verb	5
NP→Det Noun	2
NP→Det NP	2
NP→Noun Noun	4
VP→Noun NP	5
VP→Verb NP	2
VP→VP NP	2

Remember to store back-pointer!



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

Build the whole tree by branching from root.



CYK algorithm

- The probability of a constituent with a non-terminal is often called inside probability
  - $\beta_A(x, y) = \min_{k,B,C} (-log\pi(A \rightarrow BC) + \beta_B(x, k) + \beta_C(k + 1, y))$
  - Complexity?

CYK algorithm

- We can use the same CKY algorithm to calculate the marginal probability of a sentence through
- $$\beta_A(x, y) = -\log \sum_{k,B,C} exp(log\pi(A \rightarrow BC) - \beta_B(x, k) - \beta_C(k+1, y))$$
- What else can it do?
  - Recognizer: $\beta_A(x, y) = \vee_{k,B,C} (A \rightarrow BC) \in \mathscr{R} \wedge \beta_B(x, k) \wedge \beta_C(k+1, y)$
- A general form?

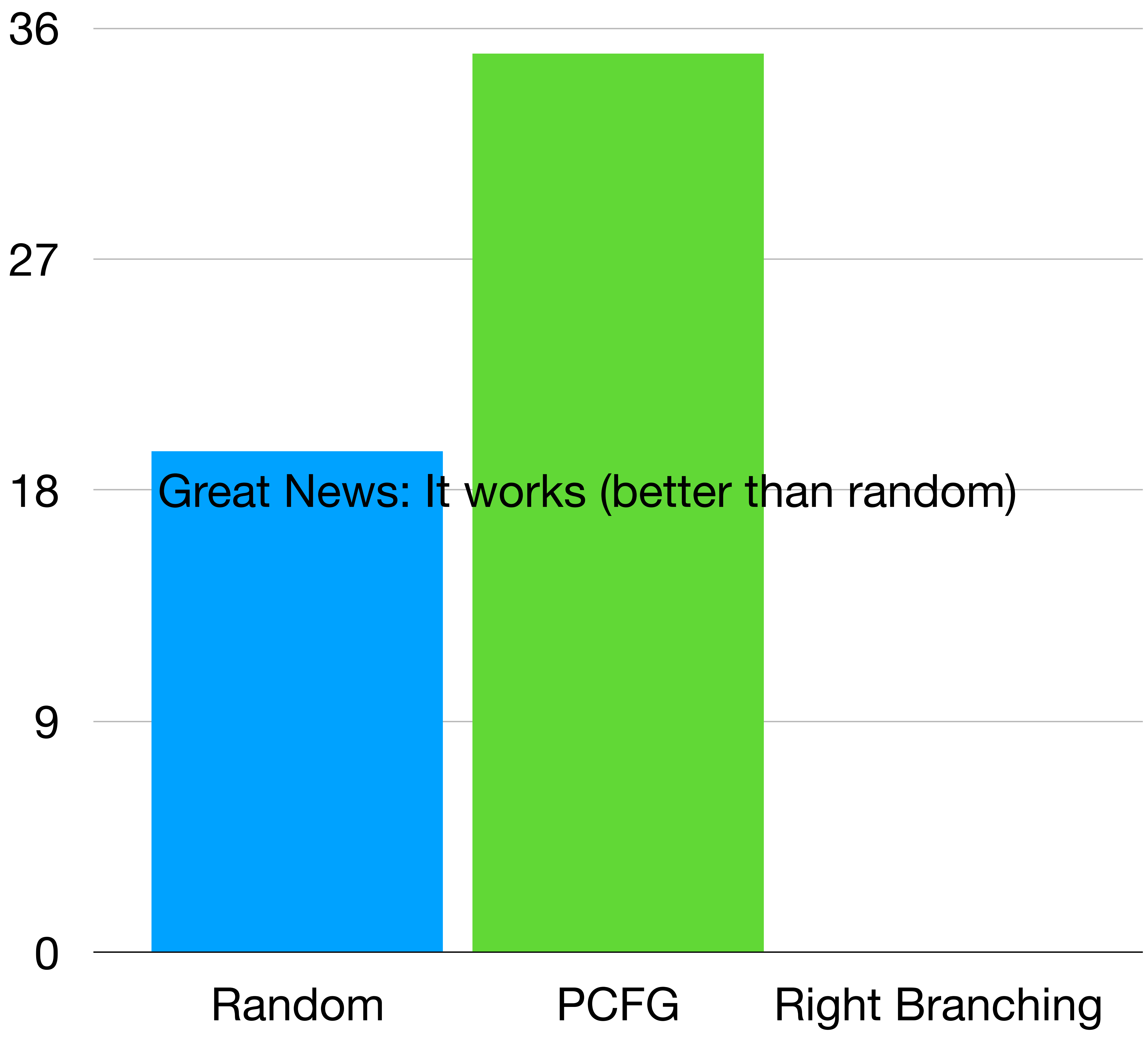
CYK algorithm

- Semiring Parsing

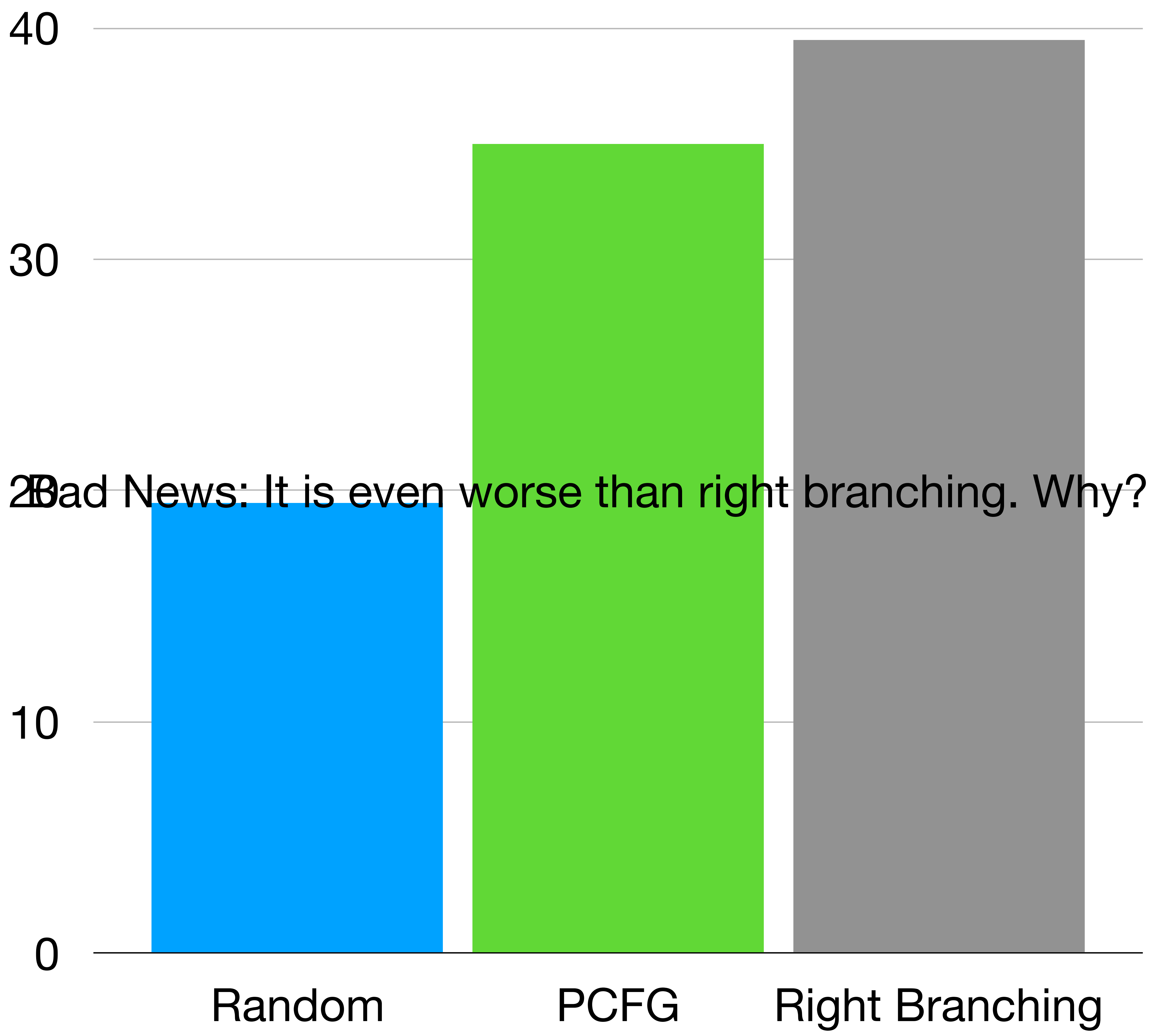
	weights	⊕	⊗	⓪	①
total prob	[0, 1]	+	x	0	1
max prob	[0, 1]	max	x	0	1
min -logp	[0, ∞]	min	+	∞	0
log prob	[-∞, 0]	logsumexp	+	-∞	0
recognizer	T/F	or	and	F	T

Optimizing PCFGs

- Traditional methods: inside-outside algorithm
- Good news: You can directly optimize the log prob calculated by CKY with autograd with the same effect and time complexity.
    - Optional reading: *Inside-Outside and Forward-Backward Algorithms Are Just Backprop*
- Similar to language models, we optimize the log probability of the sentence:
    - $\mathscr{L} = -\log \sum_{T_x} p(T_x)$



36
27
18
9
0
Great News: It works (better than random)
Random
PCFG
Right Branching



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

Neural PCFGs

- Neural parameterization for PCFGs

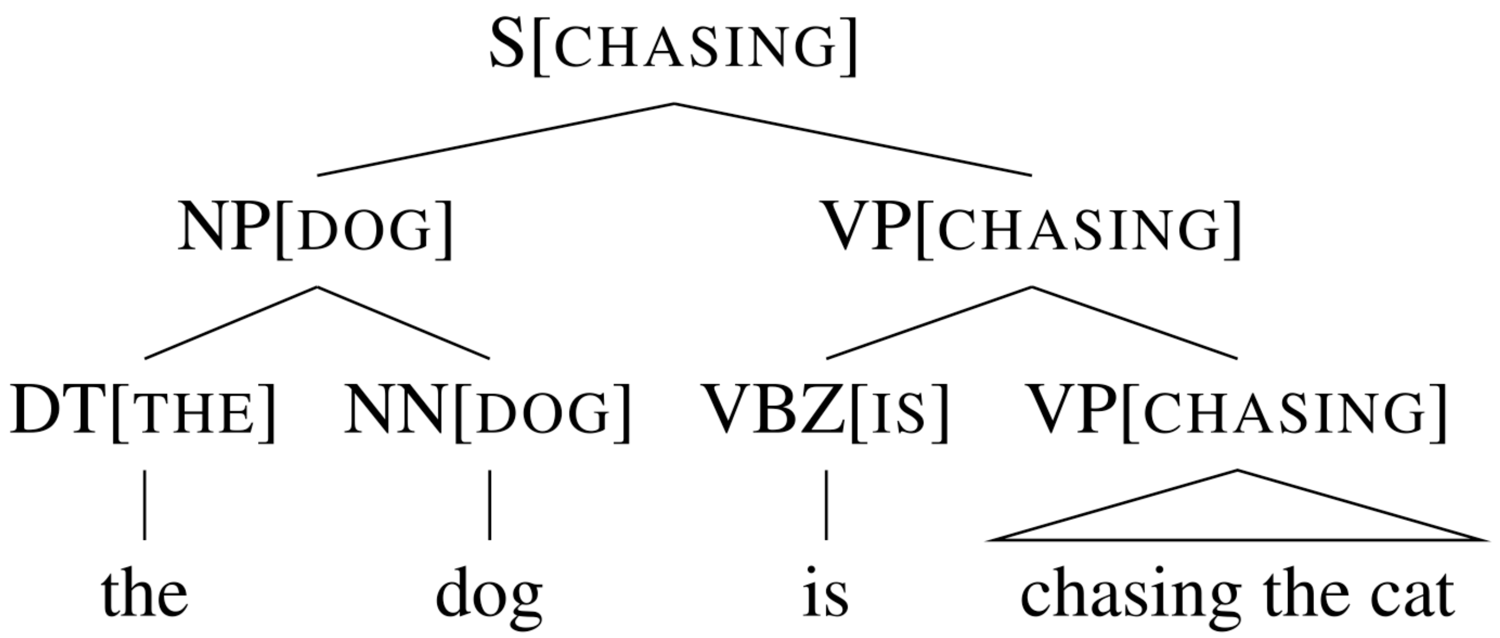
$$\pi_{T \to w} = \textsc{NeuralNet}(\mathbf{w}_T) = \frac{\exp(\mathbf{u}_w^\top f(\mathbf{w}_T))}{\sum_{w' \in \Sigma} \exp(\mathbf{u}_{w'}^\top f(\mathbf{w}_T))}$$

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

  - parameter sharing through distributed representations
  - same training method

Neural L-PCFGs

- You can further improve Neural PCFGs by adding head annotations



$$
\begin{aligned}
&S \rightarrow A, && A \in \mathcal{N} \\
&A \rightarrow BC, && A \in \mathcal{N}, B, C \in \mathcal{N} \cup \mathcal{P} \\
&T \rightarrow \alpha, && T \in \mathcal{P}
\end{aligned}
$$

$$
\begin{aligned}
&\text{1} \quad S \rightarrow A[\alpha], && A \in \mathcal{N} \\
&\text{2l} \quad A[\alpha] \rightarrow B[\alpha]C[\beta], && A \in \mathcal{N}, B, C \in \mathcal{N} \cup \mathcal{P} \\
&\text{2r} \quad A[\alpha] \rightarrow B[\beta]C[\alpha], && A \in \mathcal{N}, B, C \in \mathcal{N} \cup \mathcal{P} \\
&\text{3} \quad T[\alpha] \rightarrow \alpha, && T \in \mathcal{P}
\end{aligned}
$$

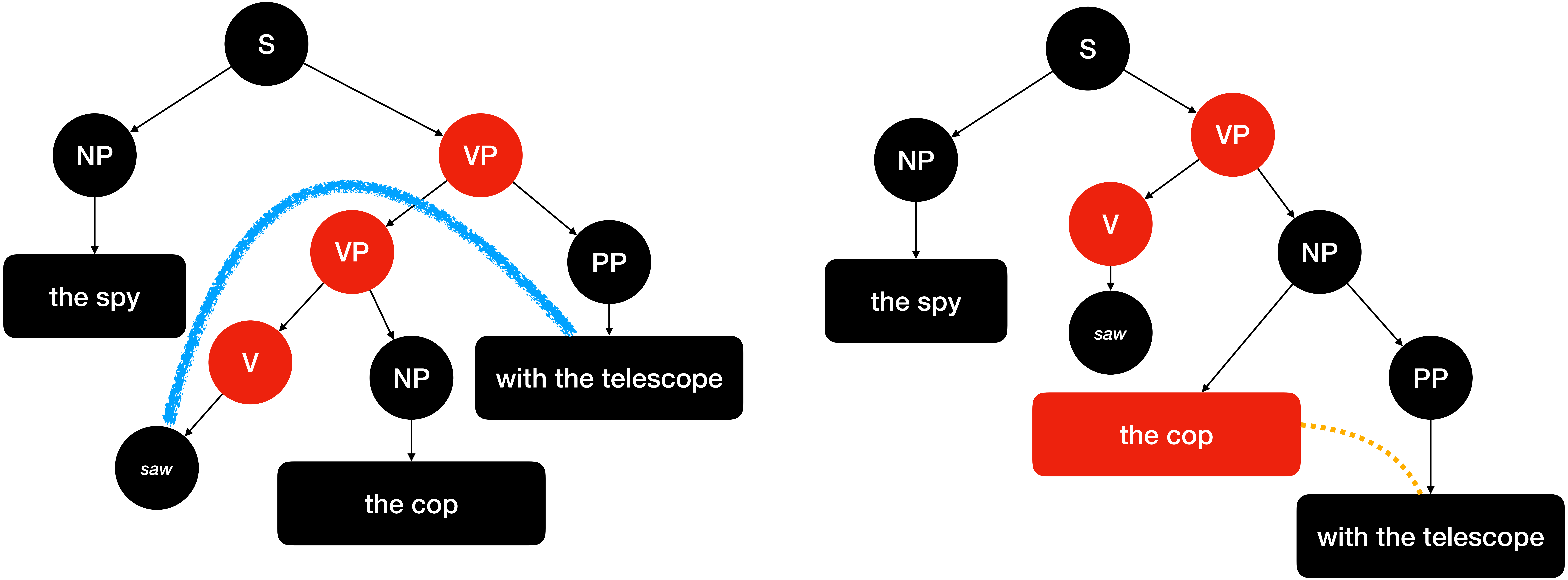
Neural L-PCFGs

- L-PCFGs did not work well because they have even MORE parameters than PCFGs

`P(Tree|sentence)`

Limitation of lexical dependencies

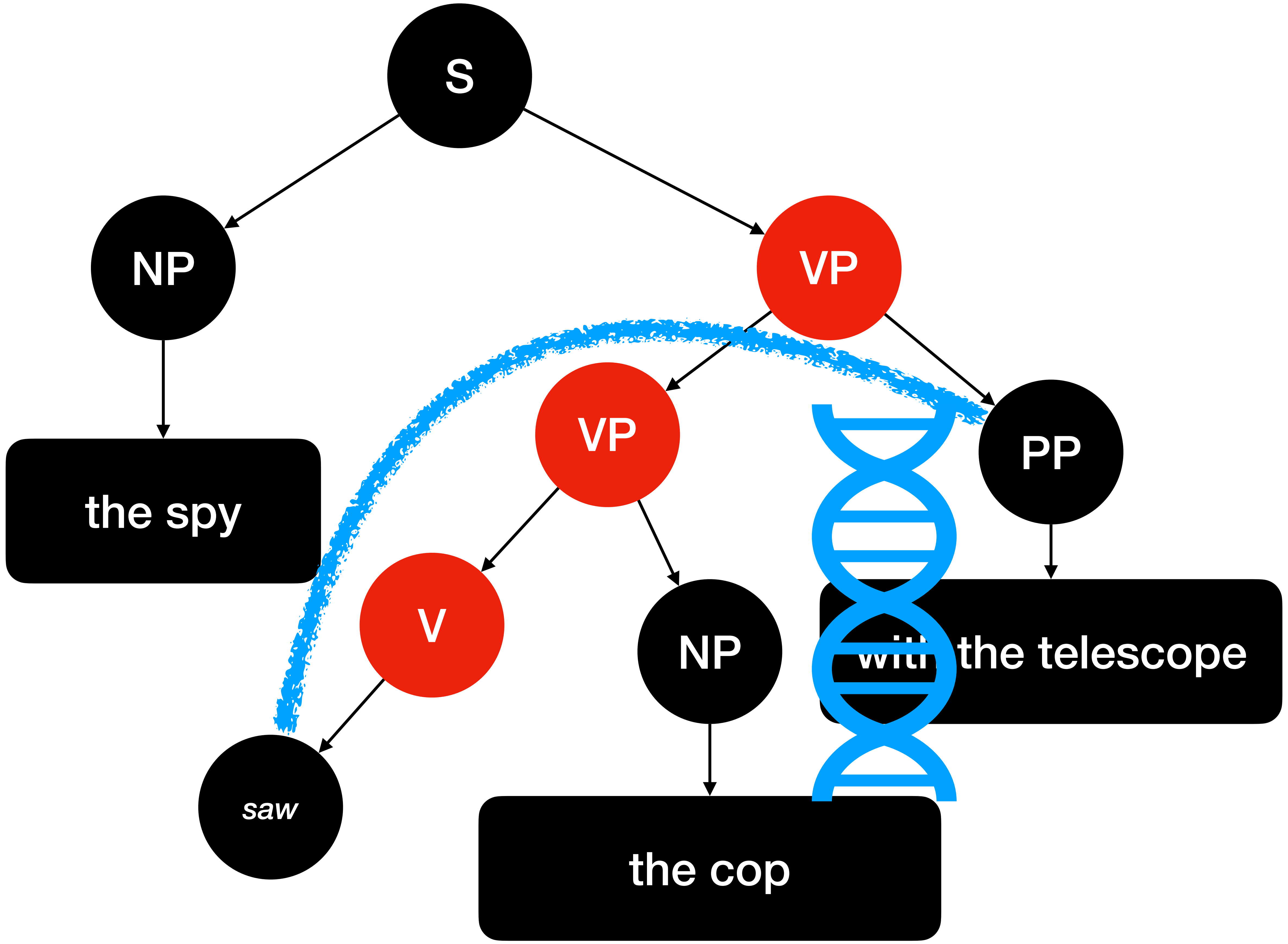
Independent head word and argument word



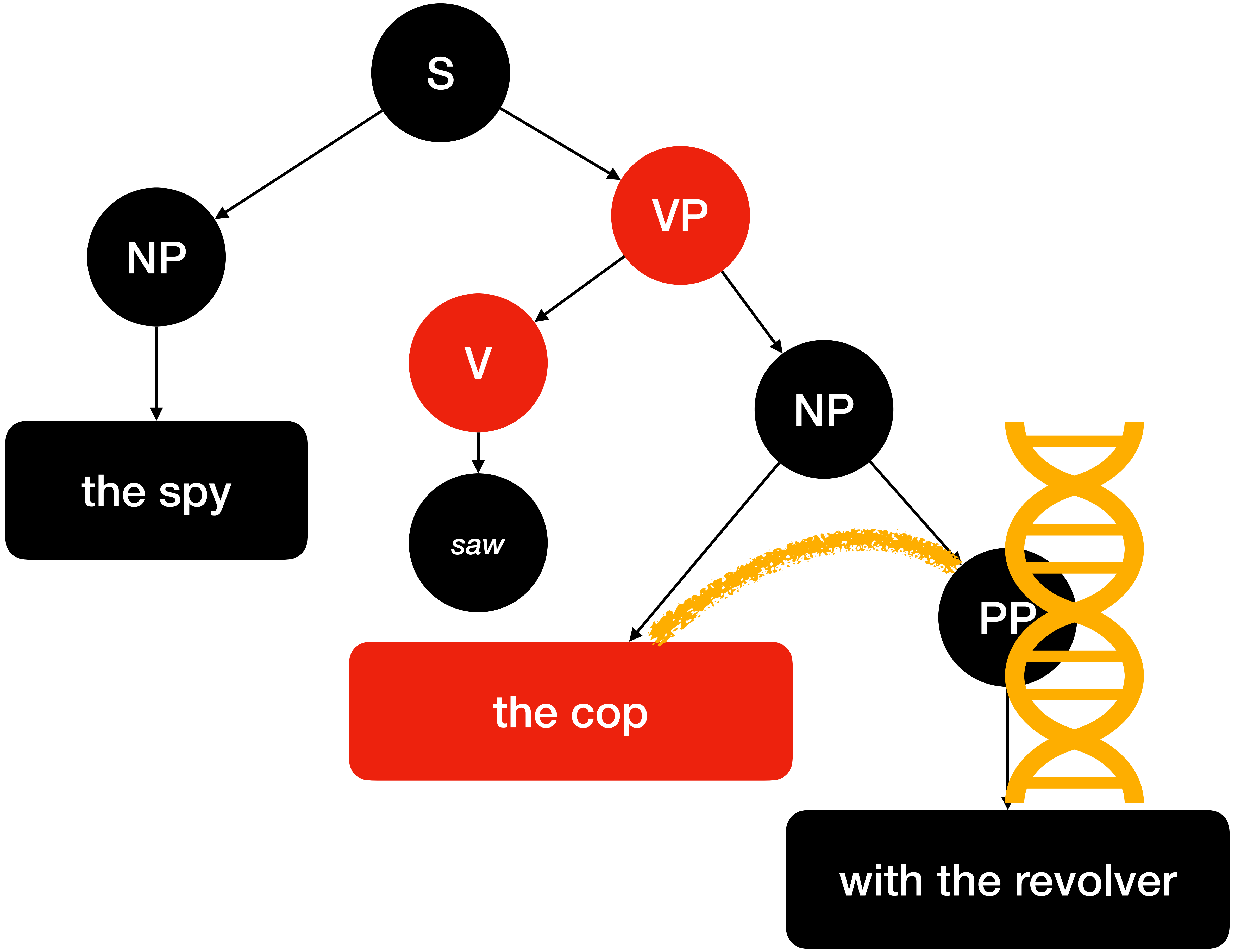
The spy saw the cop with the telescope.

Using a latent compound variable

Conditional independency

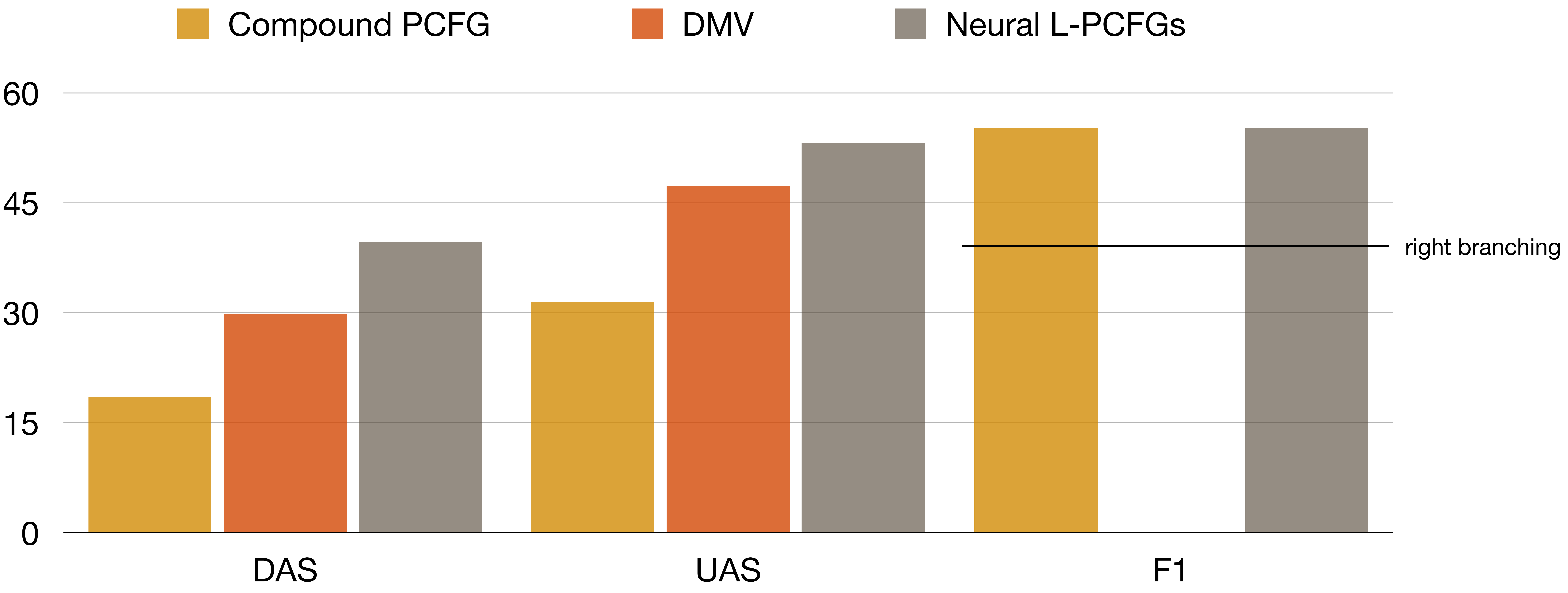


The spy saw the cop with the telescope.



The spy saw the cop with the revolver.

Results

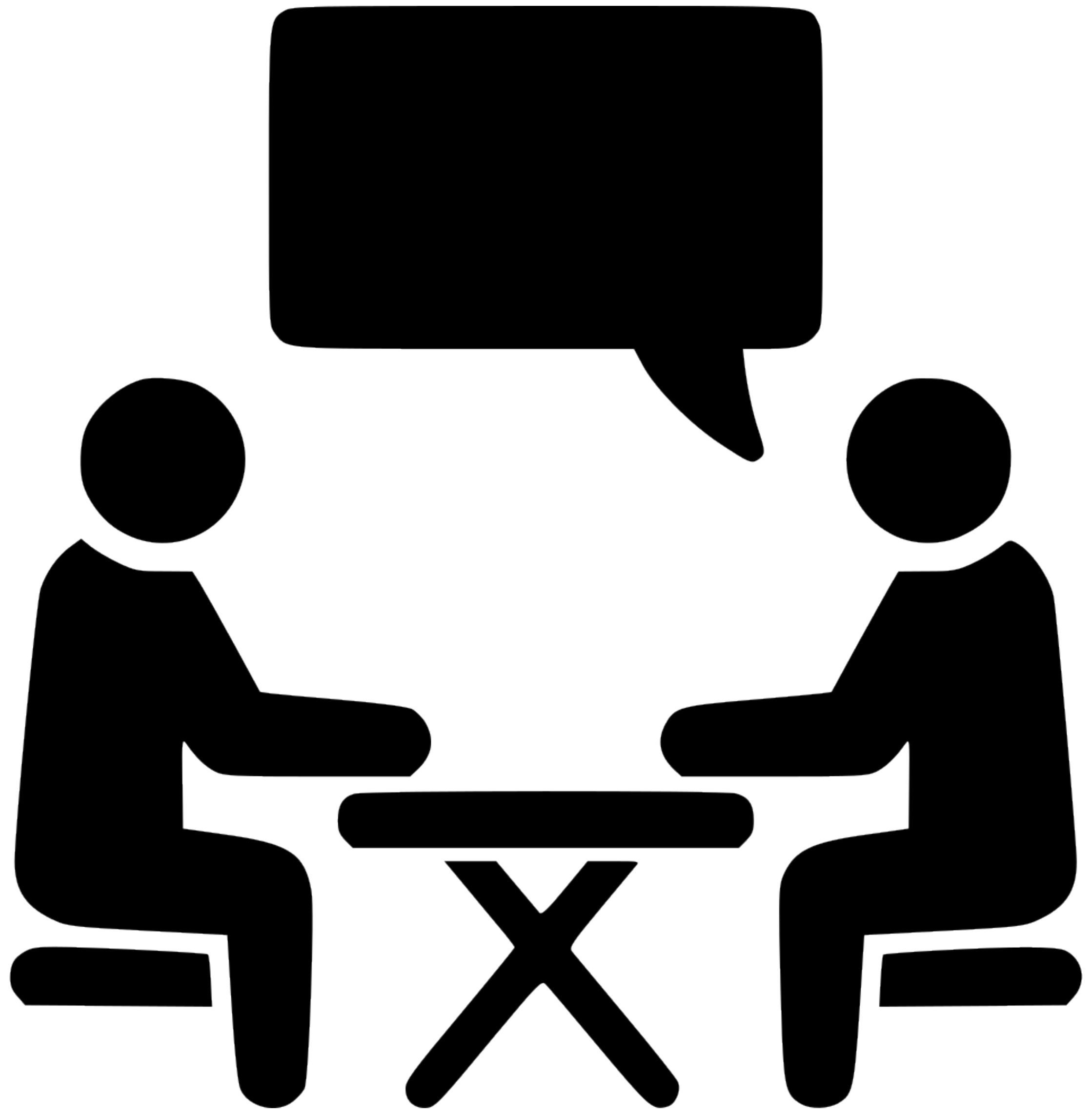

Compound PCFG
DMV
Neural L-PCFGs
60
45
30
15
0
DAS
UAS
F1
right branching

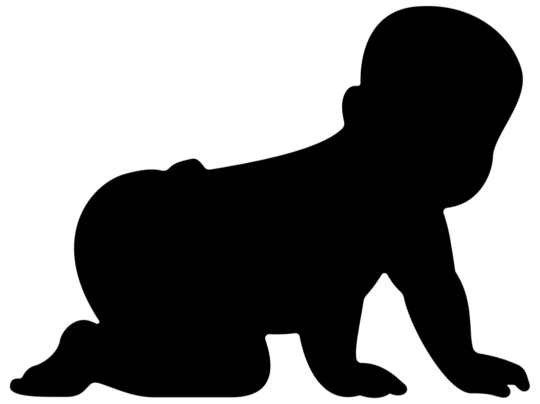
Limitations

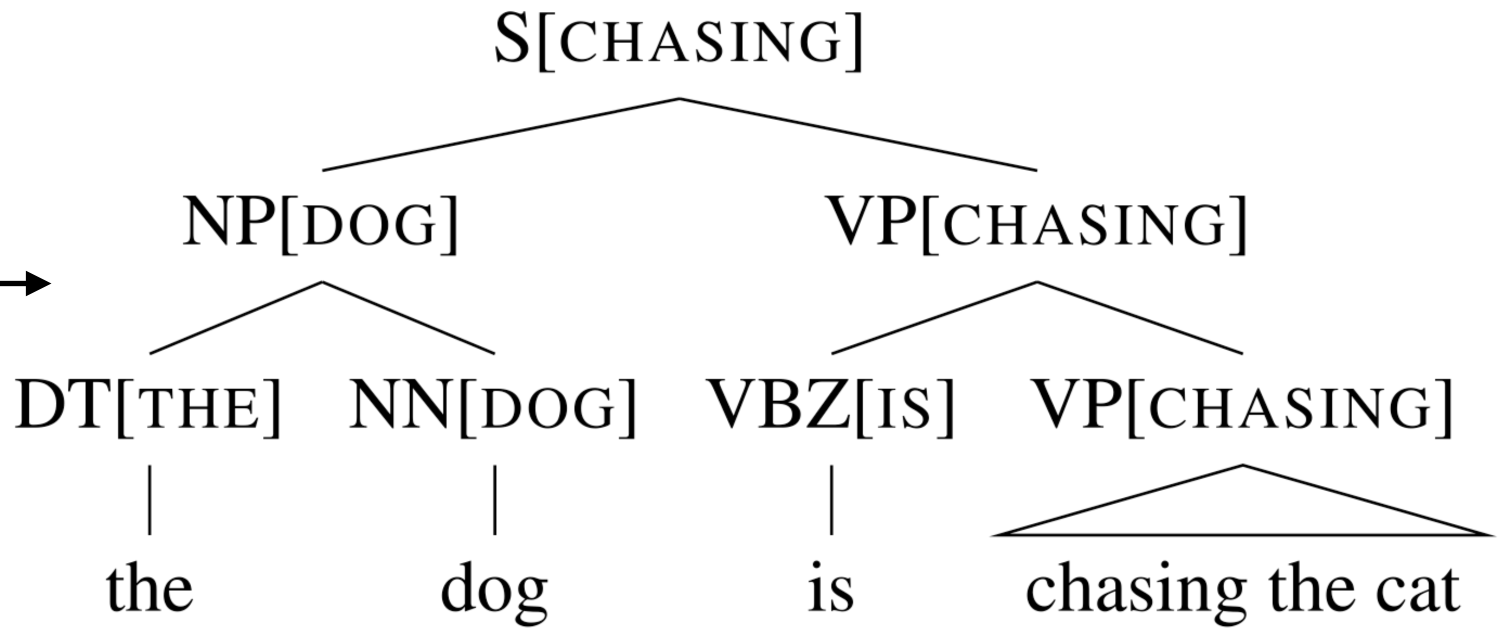
- Efficient Bilexical dependency (table assumes enough parallel workers)

	Time complexity	Space Complexity
Unilexical dependencies	$\mathcal{O}(L)$	$\mathcal{O}(L^3\lvert\mathcal{N}\rvert(\lvert\mathcal{N}\rvert+\lvert\mathcal{P}\rvert)^2)$
Bilexical Dependencies	$\mathcal{O}(L)$	$\mathcal{O}(L^4\lvert\mathcal{N}\rvert(\lvert\mathcal{N}\rvert+\lvert\mathcal{P}\rvert)^2)$

- *Neural Bi-Lexicalized PCFG Induction.*



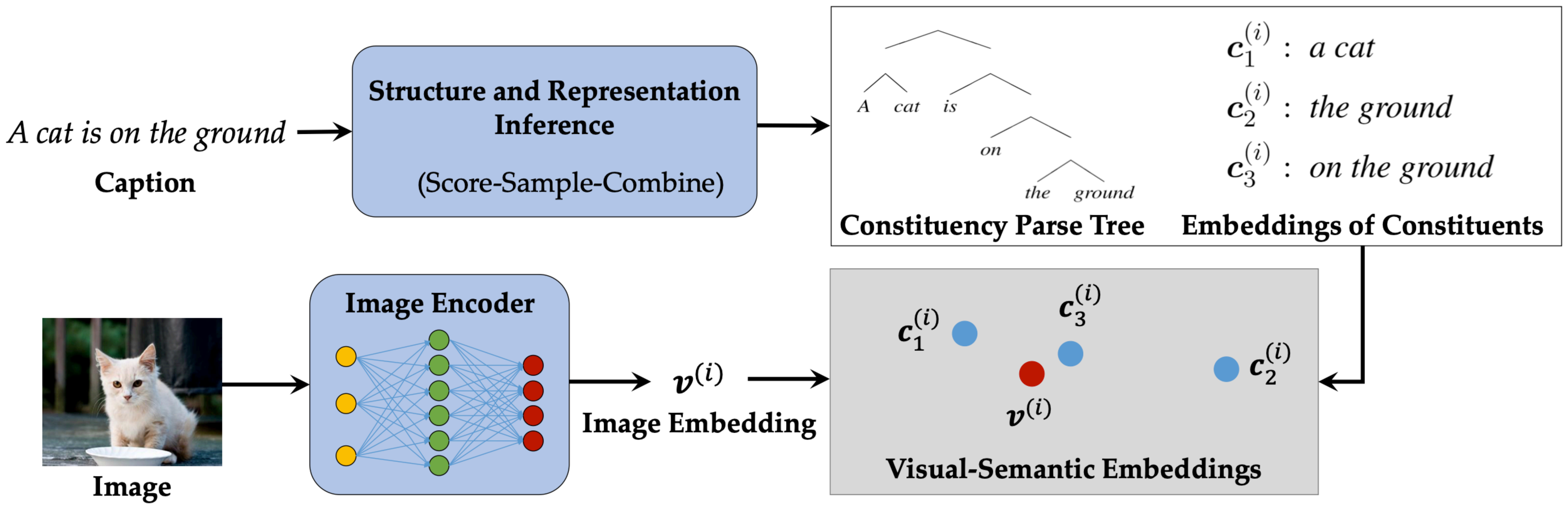


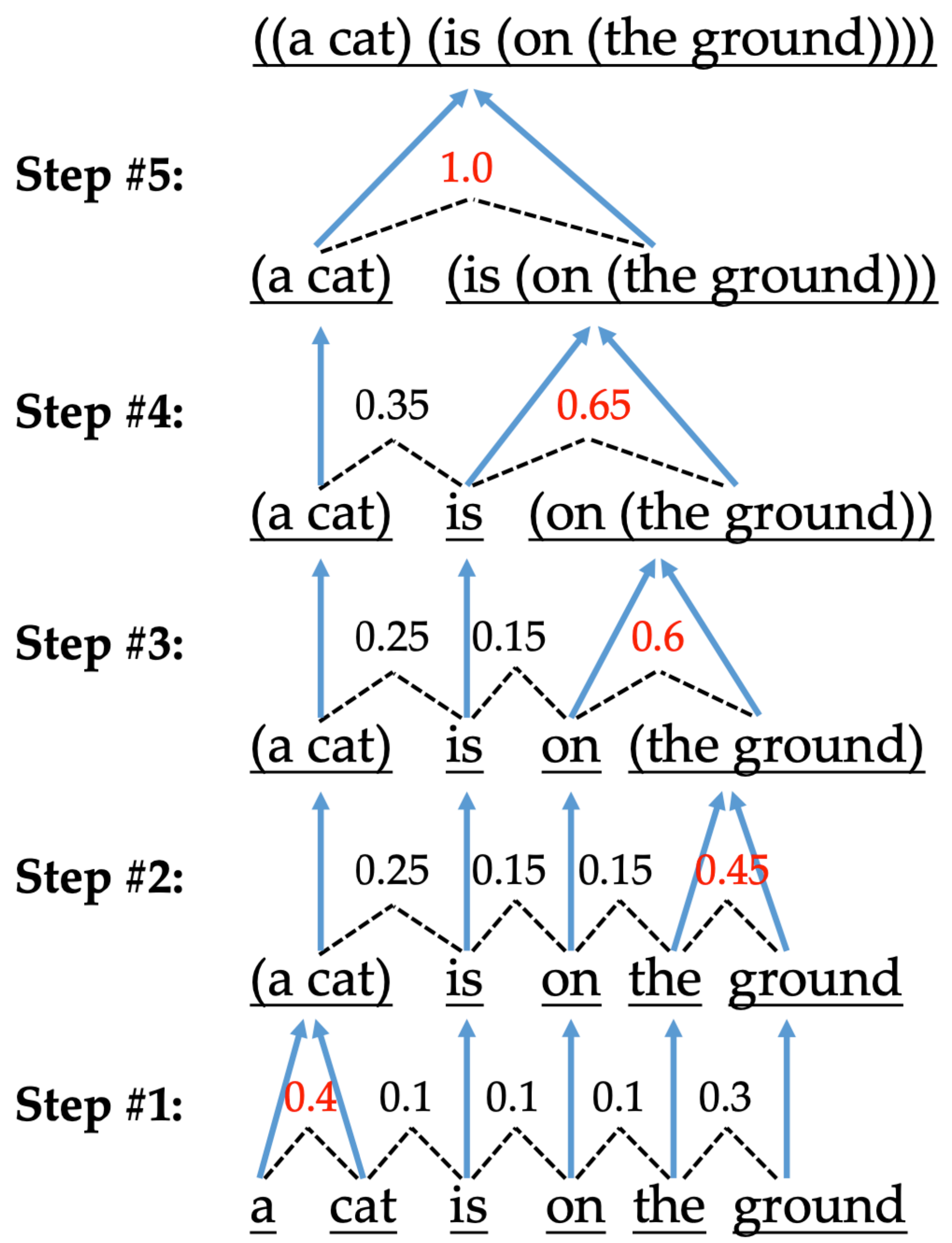


Key to the mystery: visual prior?

Visual Prior Grammar Induction

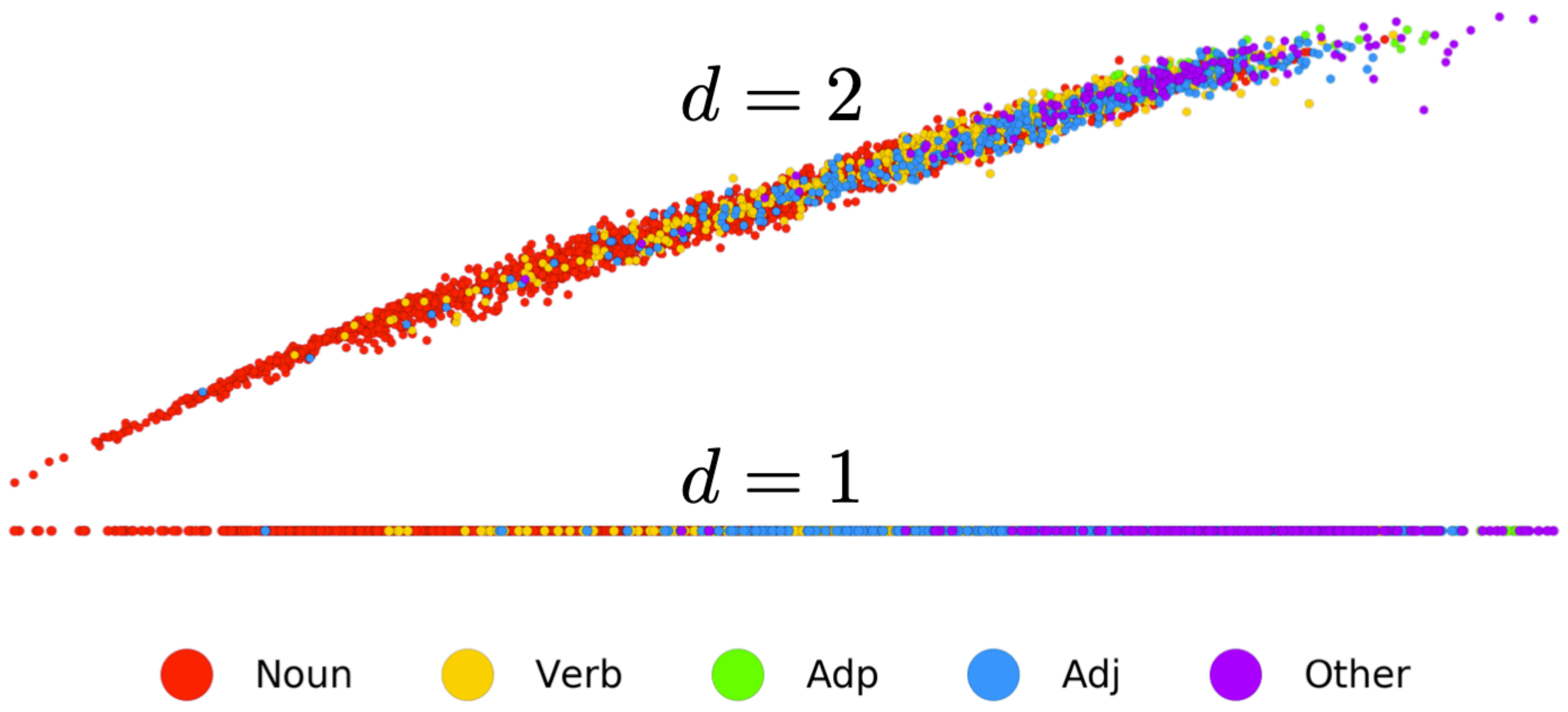
- Visual grounded neural syntax acquisition





Visual Prior Grammar Induction

- Visual grounded neural syntax acquisition
  - Similar results even if the dimension of embeddings get shrunk to 1 or 2.
  - embeddings mainly capture POS tags
  - concreteness?



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, sws, cme	77.2	17.0	53.4	18.2	49.7 ± 5.9
2, sws, cme	**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, sws, cme	74.0	35.2	62.0	24.2	51.8 ± 0.4
2, sws, cme	73.8	30.2	63.7	21.9	51.3 ± 0.1
+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, sws, cme	76.6	21.9	68.7	20.6	53.5 ± 1.4
2, sws, cme	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, sm, cmx	79.6	29.0	38.3	23.5	49.7 ± 0.2
1, smhi, cmx	77.6	**45.0**	72.3	**24.3**	$\mathbf{57.5 \pm 0.1}$
1, sm, cme	80.0	26.9	62.2	23.2	54.3 ± 0.2
1, smhi, cme	76.5	20.5	63.6	22.7	52.2 ± 0.3
1, sws, cme	77.7	26.3	**72.5**	22.0	55.5 ± 0.1
2, sws, cme	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*

References

- https://nlp.stanford.edu/seminar/details/yoonkim.pdf
- https://www.cs.jhu.edu/~jason/465/