Computational Linguistics

10

CSC 485/2501
Fall 2023

10. Unsupervised Parsing

Gerald Penn
Department of Computer Science, University of Toronto

Unsupervised parsing

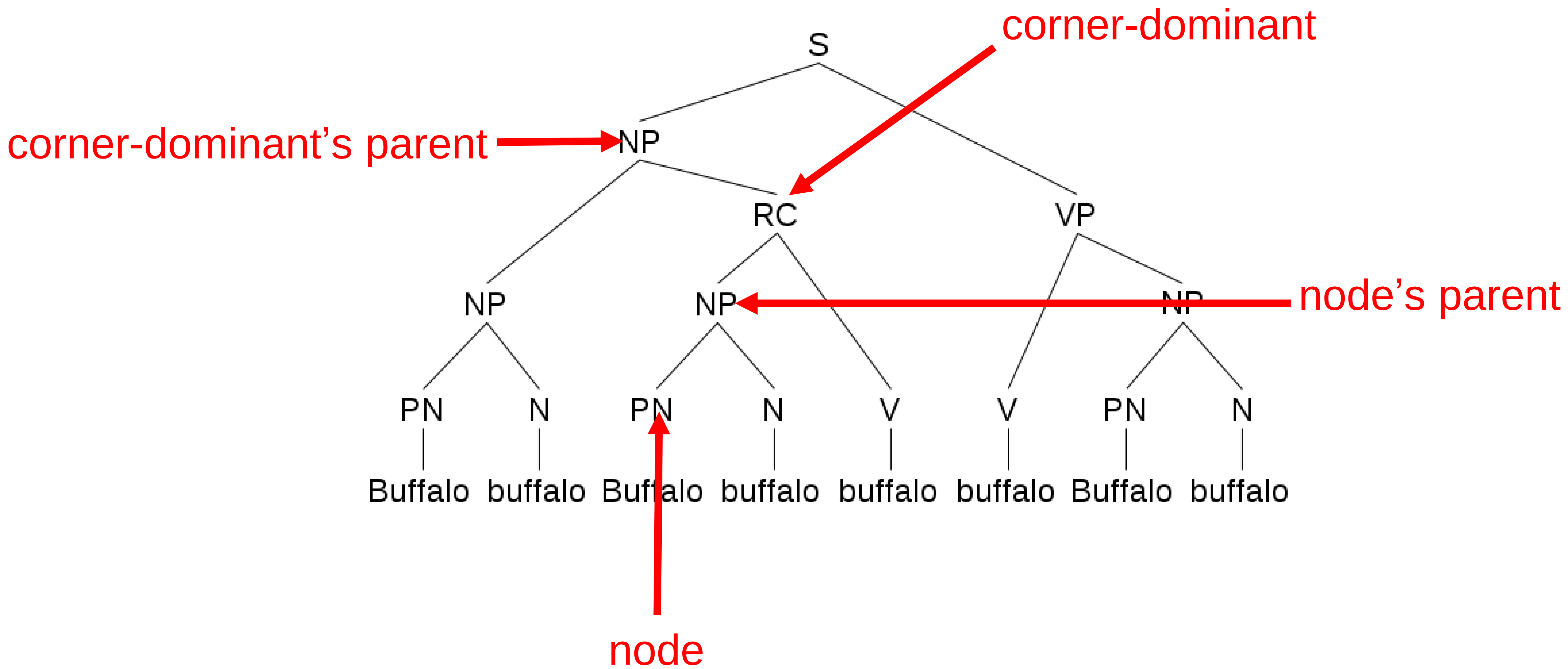
- Parsing without training on parse trees. How could such a thing be learned?
- Well, unsupervised doesn’t always mean no supervision…
  - Parts of speech
  - Binary-branch restrictions
- …and we often lower the bar in terms of what we expect the system to learn:
  - Unlabelled (binary) trees
  - Hierarchical structure without explicit, recursive rules.

PRPN: parse-read-predict

- PRPN trains a sequence of components that build a parse tree on the way to predicting the next word in a string of words – a fancy language model.
- But that means that supervising the whole system in sequence means that we must only provide words in sequence…
  - for a parser, that counts as unsupervised!
- When we are done, we can break off the later components and use the parser by itself.

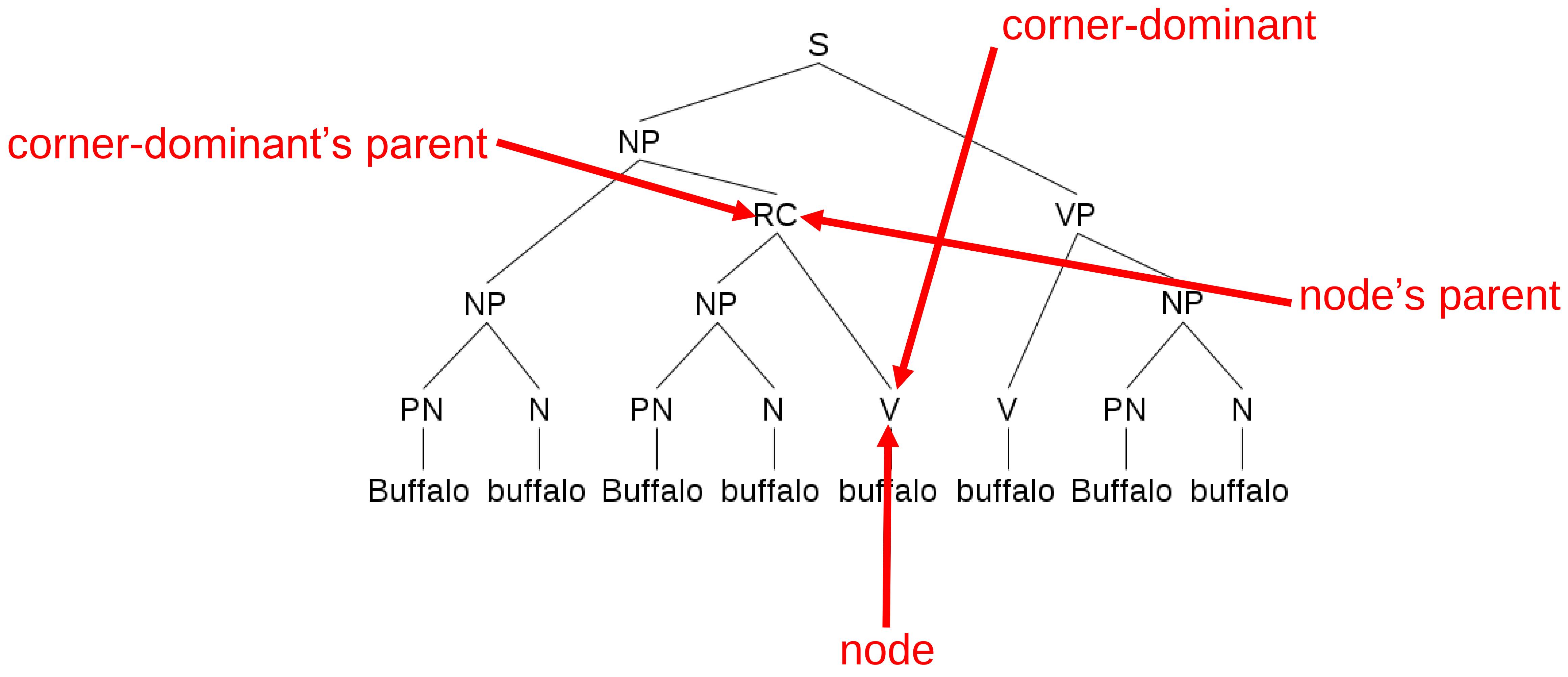
Some terminology

- "Corner-dominant"
  - The highest ancestor for which a node is the left corner, e.g.:



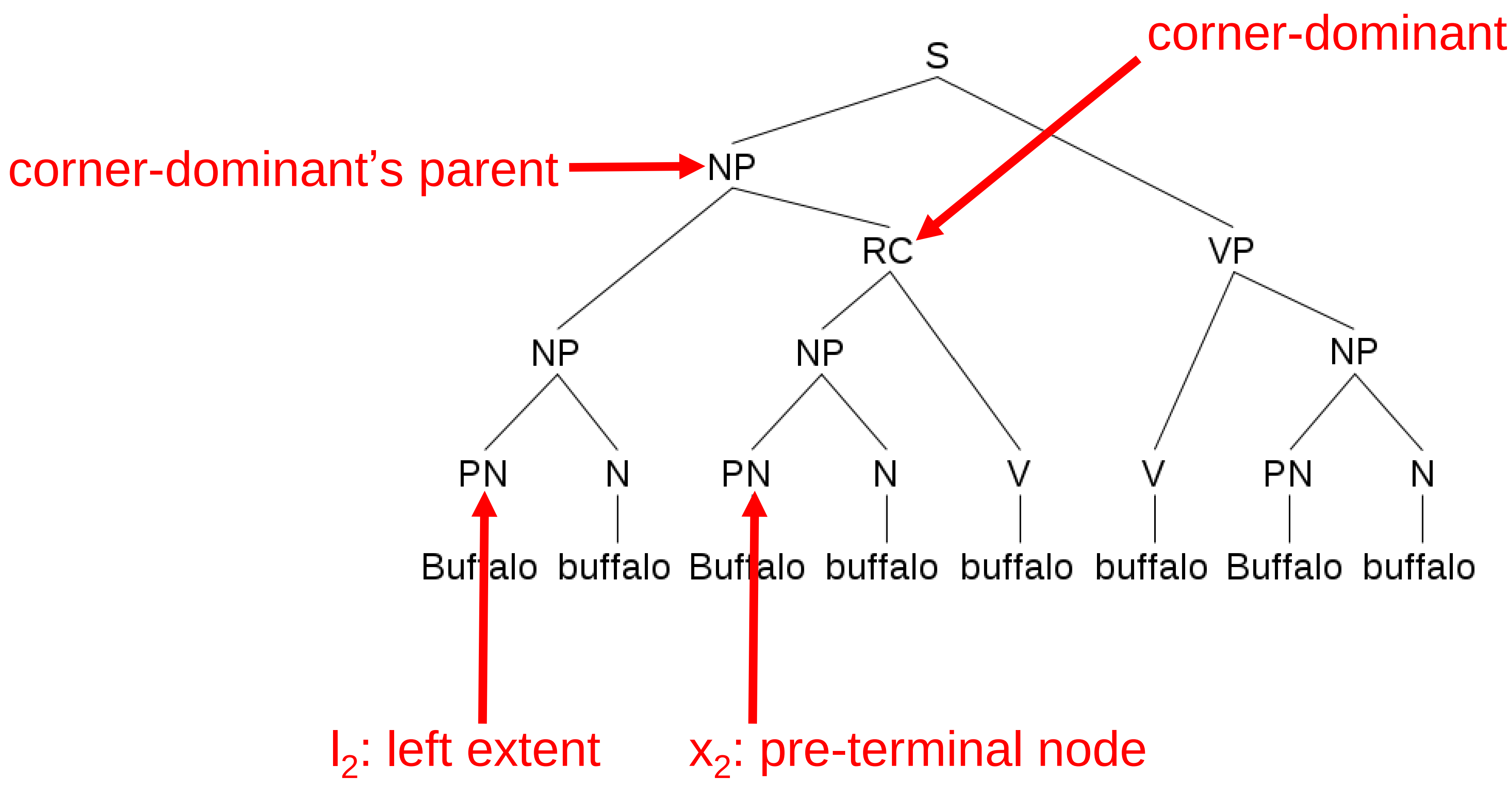
Some terminology

- “Corner-dominant”
  - The highest ancestor for which a node is the left corner, e.g.:



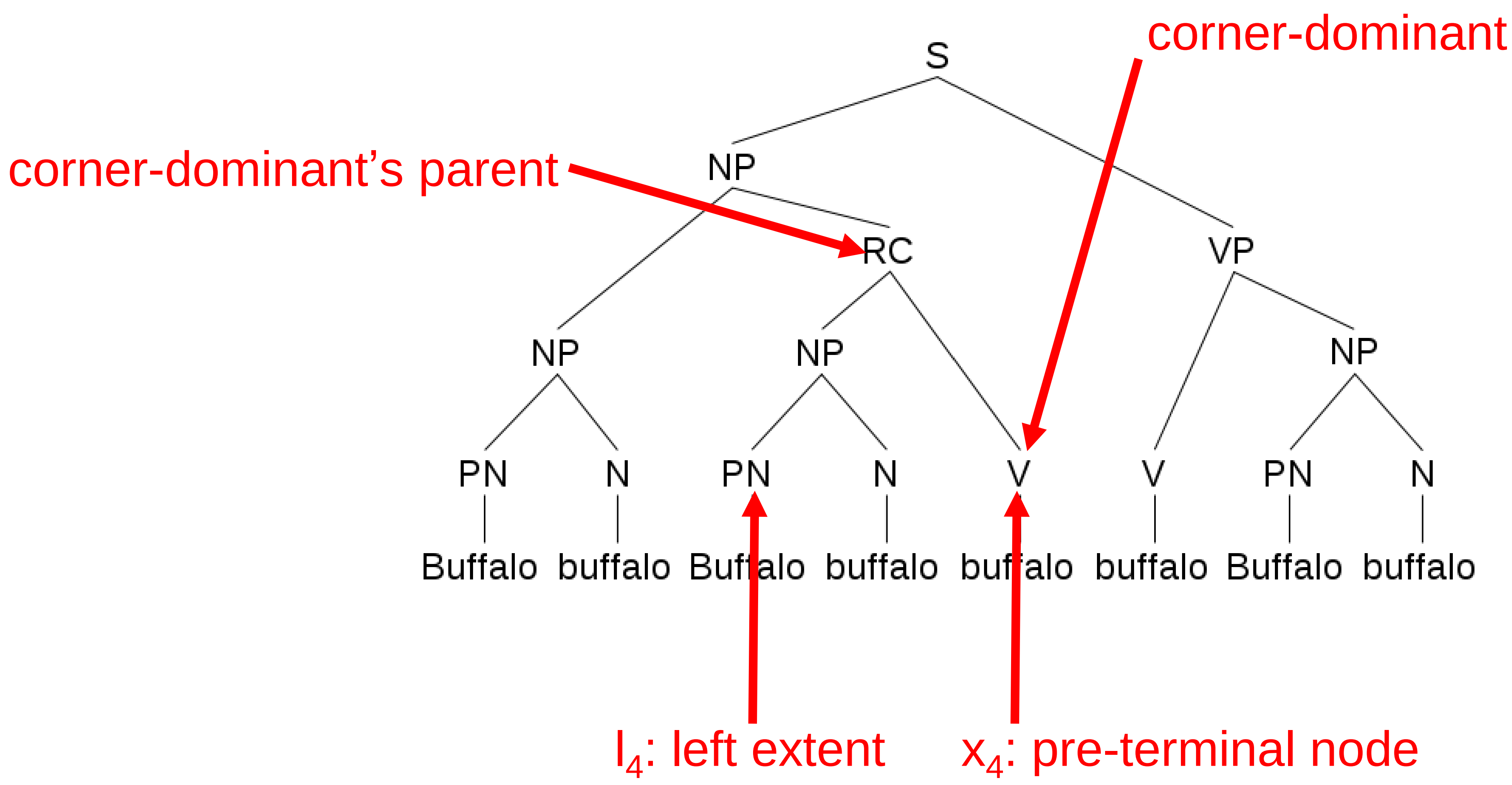
Some terminology

- "Left extent"
  - l_t = the left corner of a pre-terminal node's corner-dominant's parent, for $t > 0$, e.g.:



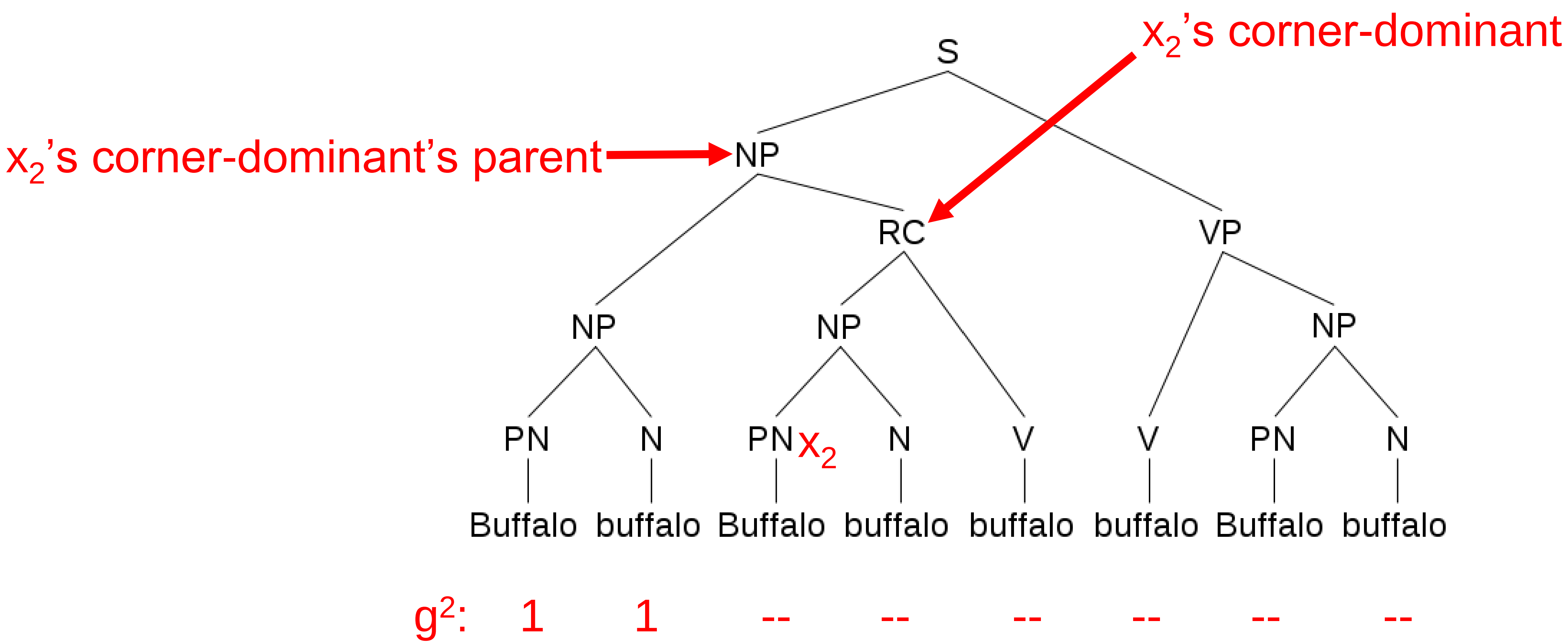
Some terminology

- "Left extent"
  - l_t = the left corner of a pre-terminal node's corner-dominant's parent, for t > 0, e.g.:



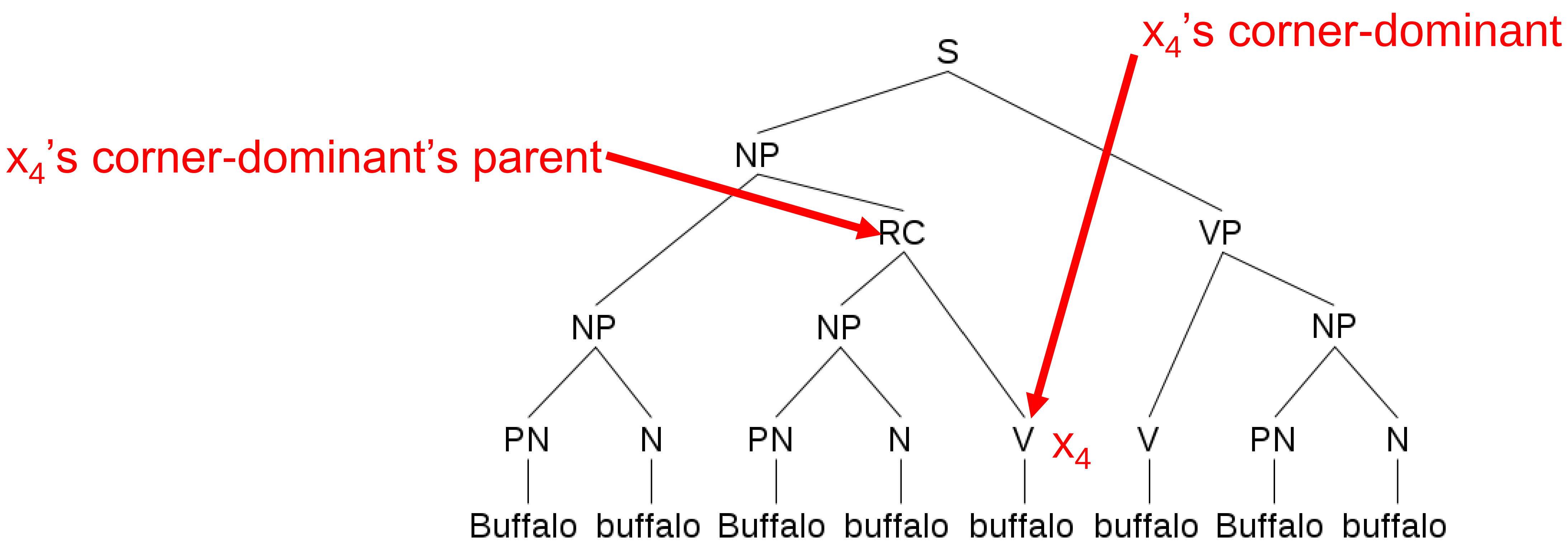
Some terminology

- "Dependency-range gate"
  - $g_i^t = \begin{cases} 1, & l_t \leq i < t \\ 0, & 0 \leq i < l_t \end{cases}$, labels left extent of x_t, e.g.:



Some terminology

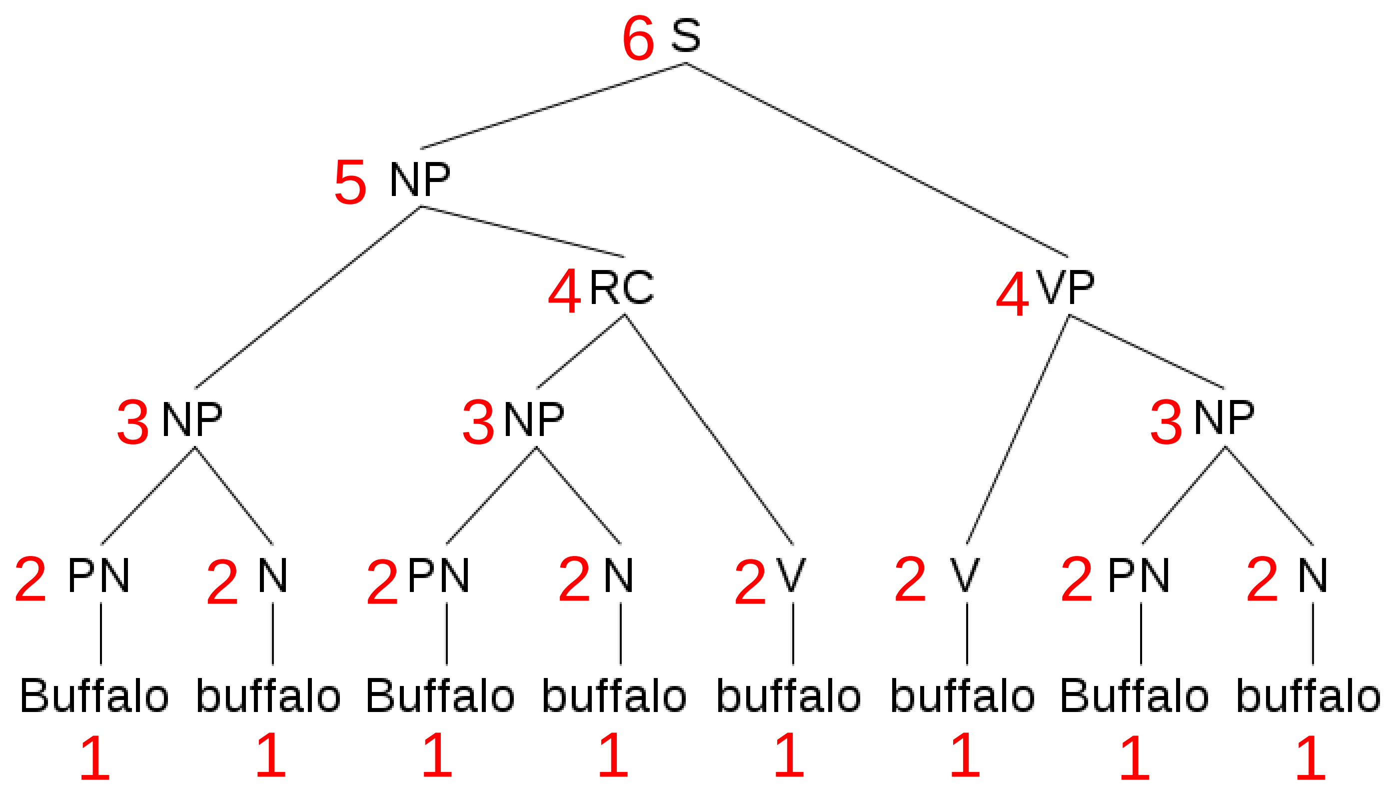
- "Dependency-range gate"
  - $g_i^t = \begin{cases} 1, & l_t \leq i < t \\ 0, & 0 \leq i < l_t \end{cases}$, labels left extent of x_t, e.g.:



g^4: 0 0 1 1 -- -- -- --

Some terminology

- "Height"
  - h(w) = 1,
  - h(n) = $\max_{m \in T_n \setminus n} h(m) + 1.$



Note: height is not depth, nor is it h(root)-depth. Count from the bottom.

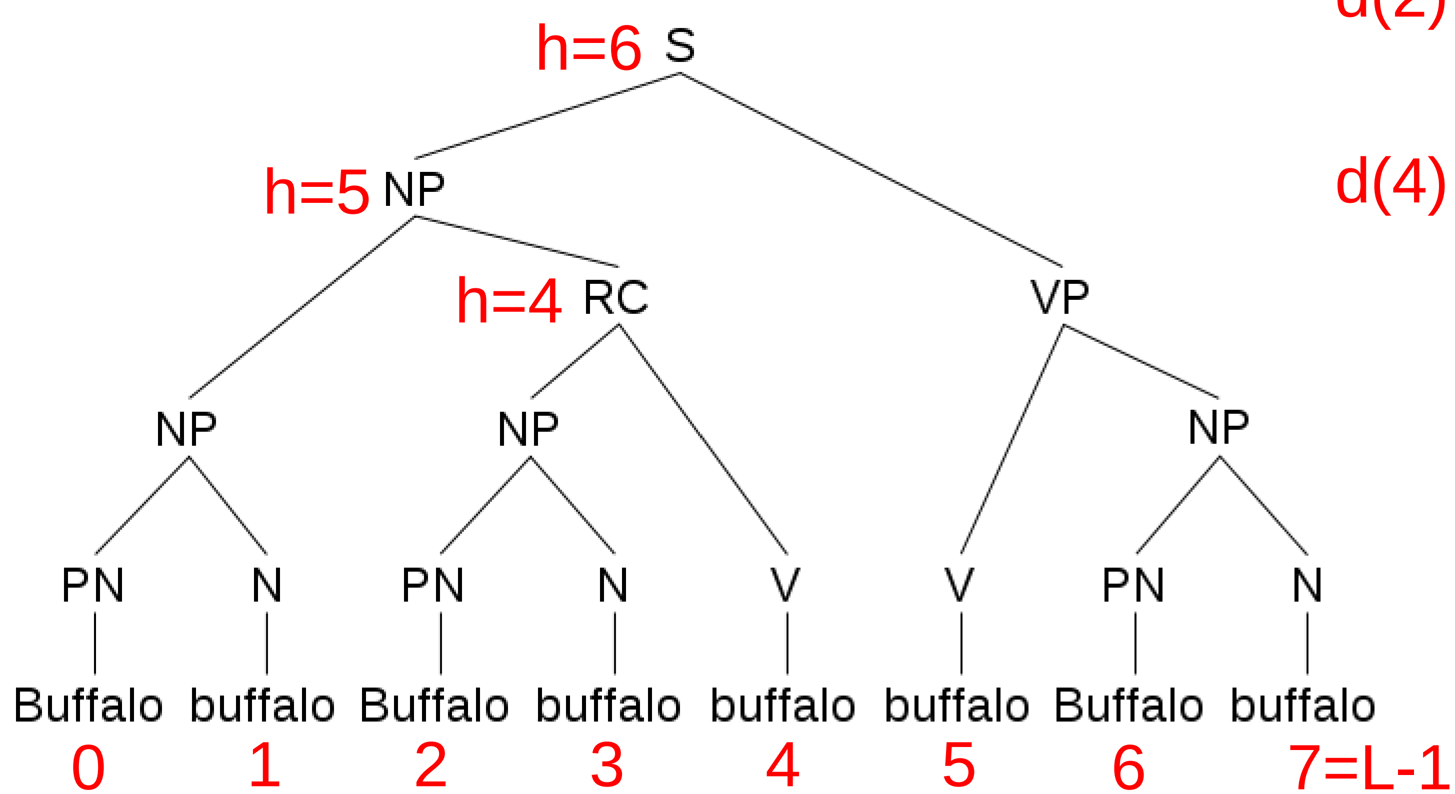
Some terminology

- "Roark-Hollingshead distance"
  - $d(i) = d_i = \frac{h(w_{i-1}, w_i) - 2}{h(r) - 1}$.

$d(0) = \frac{6+1-2}{6-1} = 1$

$d(2) = \frac{5-2}{6-1} = \frac{3}{5}$

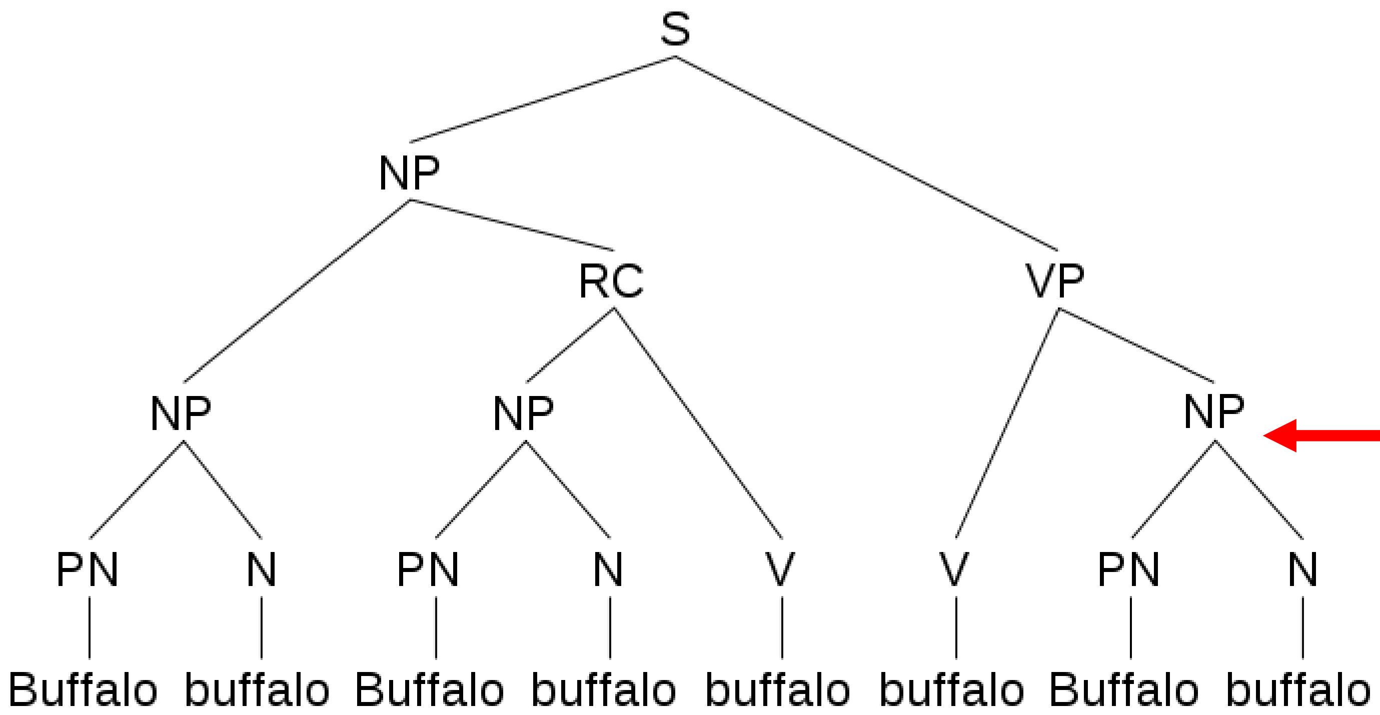
$d(4) = \frac{4-2}{6-1} = \frac{2}{5}$



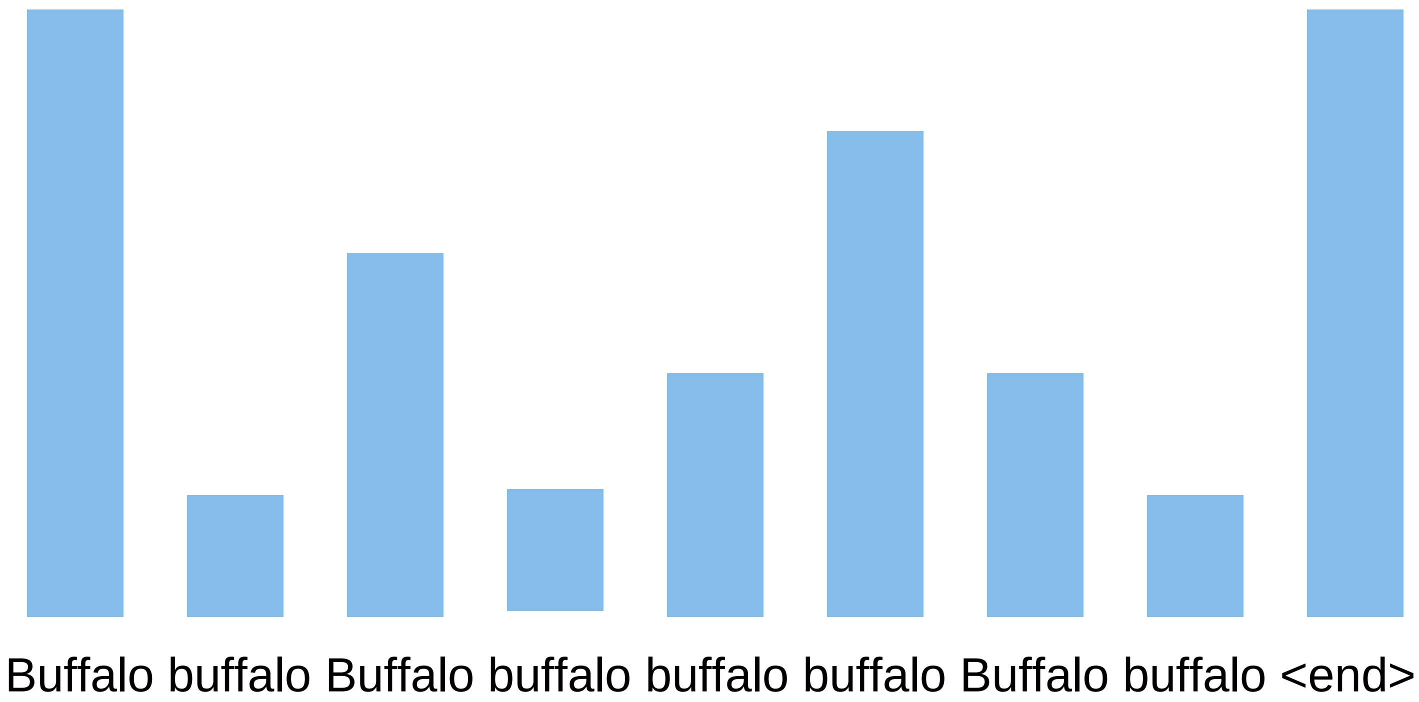
where $h(w_{-1}, w_0) = h(w_{L-1}, w_L) = h(r)+1$,

$h(u,v) = h(u \sqcup v)$ everywhere else (trees are CNF).

Roark-Hollingshead Conjecture



Q: How much of this does this preserve?

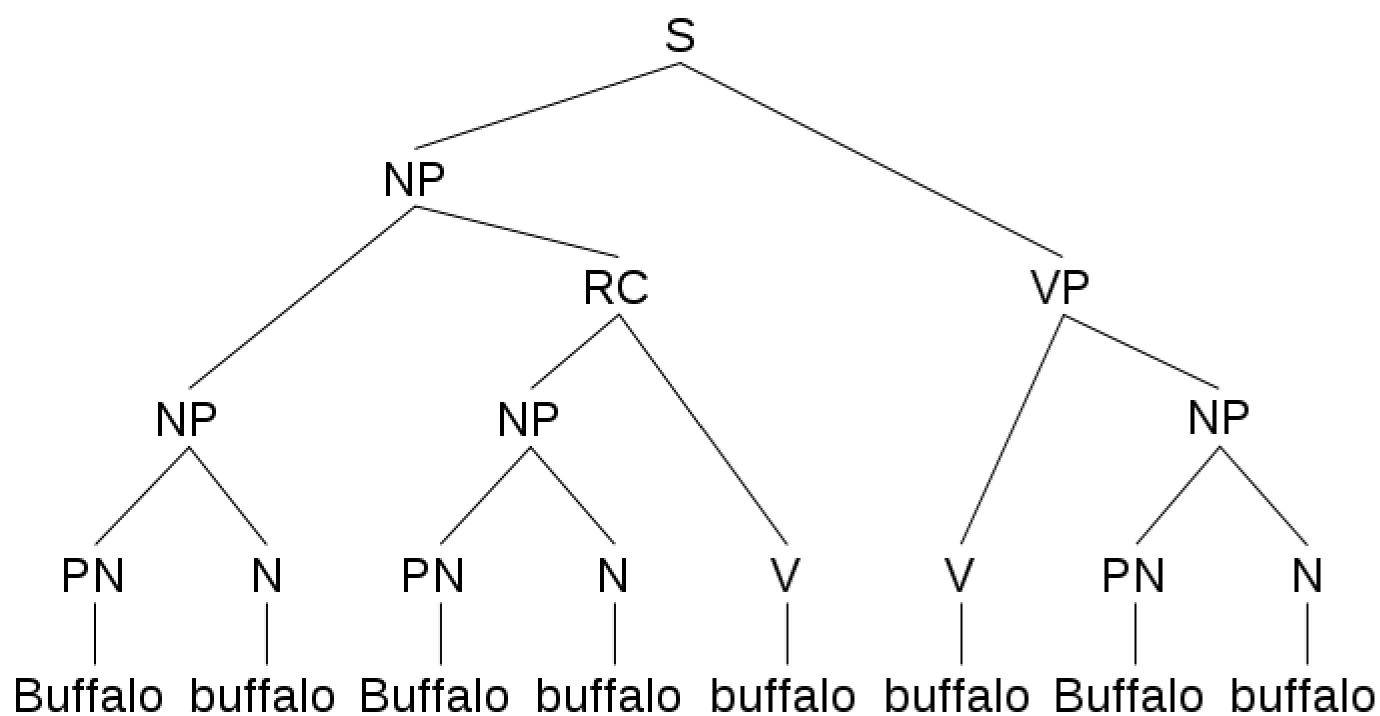


A: All of it (except labels)!

Very cool, because this is a “local” statistic.

Some terminology

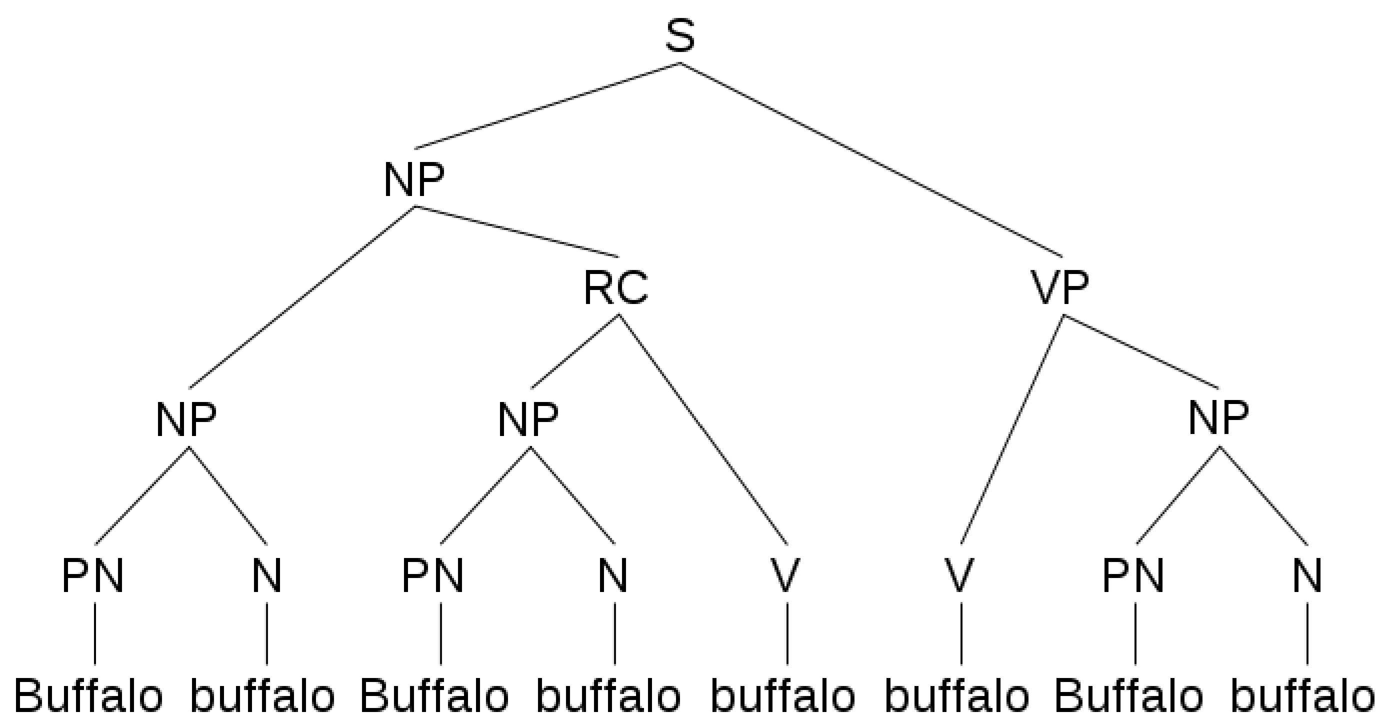
- “Dependency range”
  - $\alpha_i^t = \frac{sign(d_t - d_{i+1}) + 1}{2}$, where $i < t$.



α^6: 1 0 1 $\frac{1}{2}$ 0 $\frac{1}{2}$ -- --

PRPN’s big idea

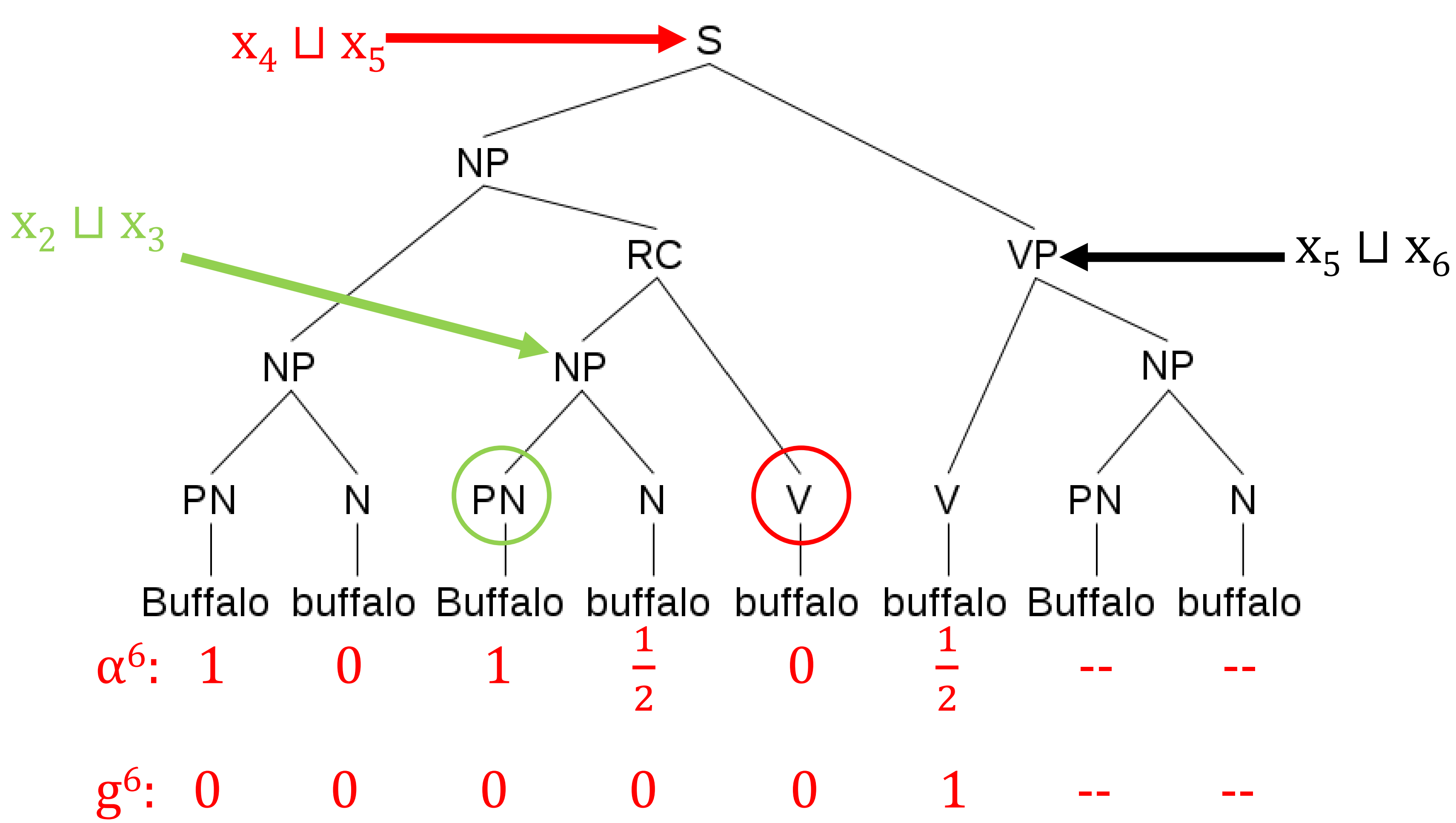
$$g_i^t = P(l_t \le i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t .$$



α^6:	1	0	1	$\frac{1}{2}$	0	$\frac{1}{2}$	--	--
g^6:	0	0	0	0	0	1	--	--

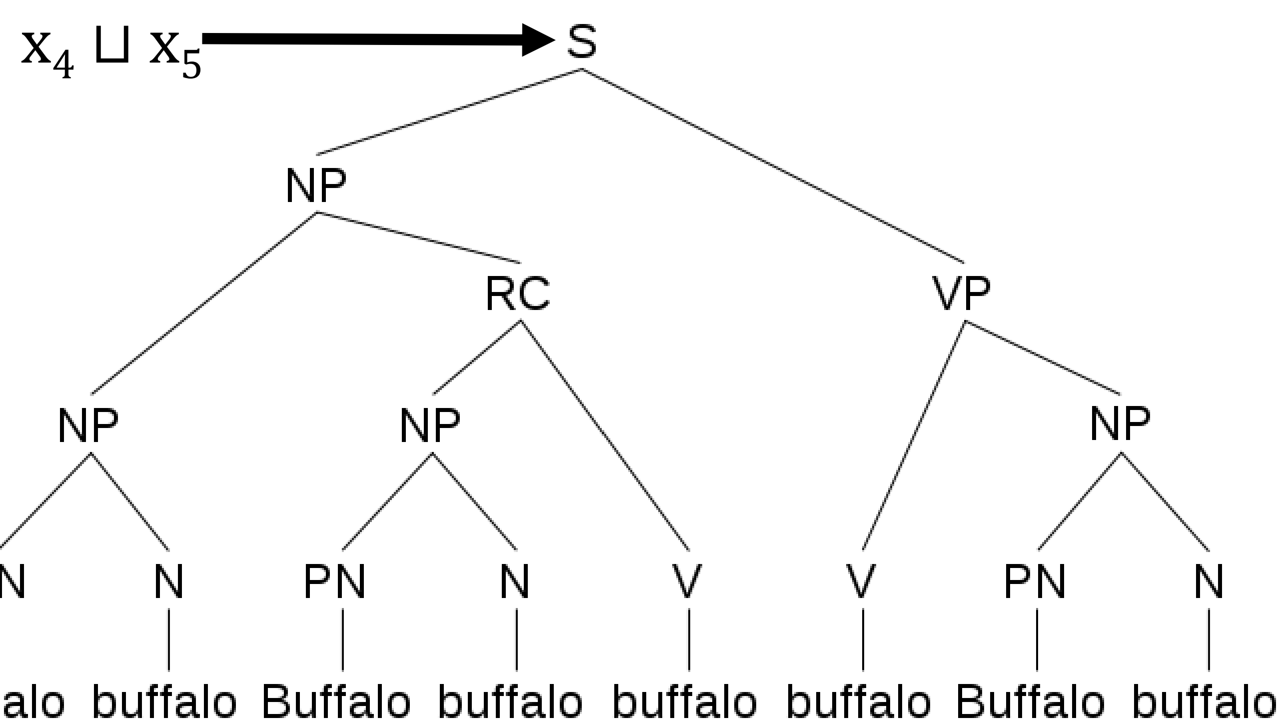
PRPN's big idea

$$g_i^t = P(l_t \leq i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t .$$



PRPN's big idea

$$g_i^t = P(l_t \le i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t .$$



α^5:	1	1	1	1	$\frac{1}{2}$	--	--	--
g^5:	1	1	1	1	1	--	--	--

PRPN's big idea

- But if $P(l_t \leq i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t$, then:

$$P(l_t \leq i) \approx \prod_{j=i+2}^{t-1} \alpha_j^t \cdot \alpha_{i+1}^t \approx P(l_t \leq i+1) \cdot \alpha_{i+1}^t, \text{ so:}$$

$$\alpha_{i+1}^t \approx P(l_t \neq i+1 \,|\, l_t \leq i+1)$$

$$1 - \alpha_i^t \approx P(l_t = i \,|\, l_t \leq i), \text{ and:}$$

$$P(l_t = i) = P(l_t = i \,|\, l_t \leq i) \cdot P(l_t \leq i)$$

$$\approx \left(1 - \alpha_i^t\right) \cdot \prod_{j=i+1}^{t-1} \alpha_j^t.$$

- This is an example of the well-known *stick-breaking process*. When $\alpha_j = 1 - \beta_j$, and the β_j are samples from beta distributions, this process is an instance of a Dirichlet process.

PRPN's big idea

- But if $P(l_t \leq i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t$, then:

$$P(l_t \leq i) \approx \prod_{j=i+2}^{t-1} \alpha_j^t \cdot \alpha_{i+1}^t \approx P(l_t \leq i+1) \cdot \alpha_{i+1}^t, \text{ so:}$$

$$\alpha_{i+1}^t \approx P(l_t \neq i+1 \,|\, l_t \leq i+1)$$

$$1 - \alpha_i^t \approx P(l_t = i \,|\, l_t \leq i), \text{ and:}$$

$$P(l_t = i) = P(l_t = i \,|\, l_t \leq i) \cdot P(l_t \leq i)$$

$$\approx \left(1 - \alpha_i^t\right) \cdot \prod_{j=i+1}^{t-1} \alpha_j^t.$$

- This is very well suited to modelling the dependence of l_t upon as many words/pre-terminals as we want.

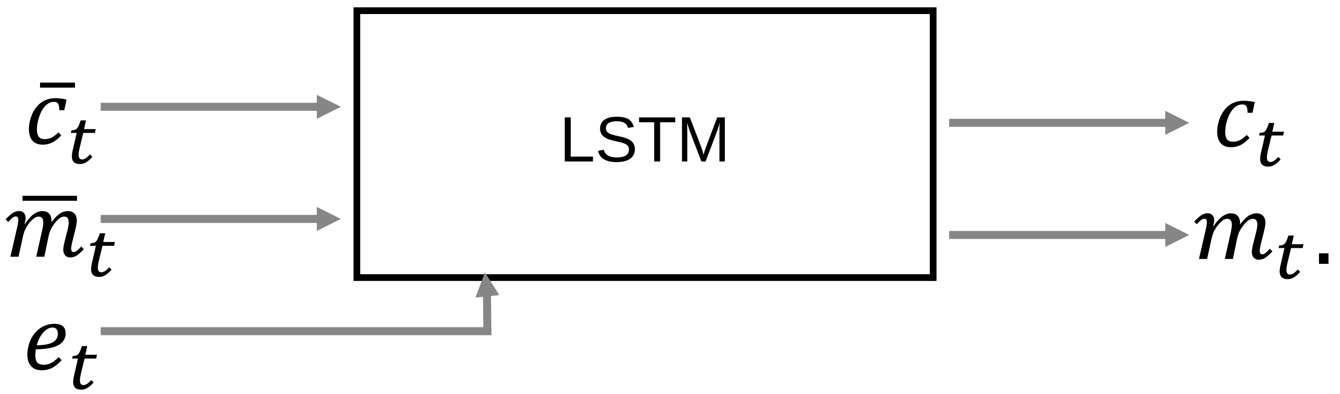
PRPN: parse

- Soften up “Dependency range:”
  - $\cancel{\alpha_i^t = \frac{sign(d_t - d_{i+1}) + 1}{2}}$, where $i < t$, becomes:
  - $\alpha_i^t = \frac{hardtanh((d_t - d_{i+1}) \cdot \tau)}{2}$, where τ is temperature,
  - and $hardtanh(x) = \max(-1, \min(1, x))$.

- Then learn RH distance with a 2-layer convolution:
  - $q_t = ReLU\left(W_q \begin{bmatrix} e_{t-L} \\ e_{t-L+1} \\ \cdots \\ e_t \end{bmatrix} + b_q\right)$,

    Word vectors for w_{i-L}, w_{i-L+1}, … w_i
  - $d_t = ReLU(W_d q_t + b_d)$.
  - But we’re not going to supervise this with d_t from actual trees…

PRPN: read

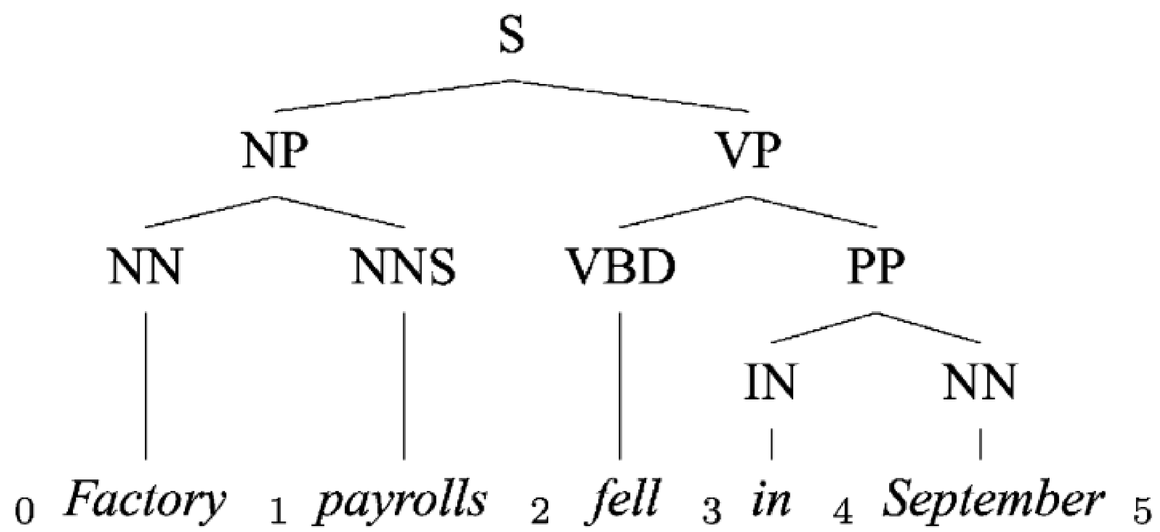
- Instead, we couple the input to memory states m_i and use RH distance to interpolate mixtures of previous time steps into “summary vectors” that predict subsequent memory states:
  - $k_t = W_m m_{t-1} + W_e e_t,$
  - $\bar{s}_i^t = softmax\left(\frac{m_i k_t^T}{\sqrt{\dim(k)}}\right),$
  - $s_i^t = \frac{g_i^t}{\sum_j g_j^t} \bar{s}_i^t,$ Big idea: depends on d_i’s now
  - $\begin{bmatrix} \bar{m}_t \\ \bar{c}_t \end{bmatrix} = \sum_{i=1}^{t-1} s_i^t \cdot \begin{bmatrix} m_i \\ c_i \end{bmatrix},$ Summary vector
  - 

PRPN: predict

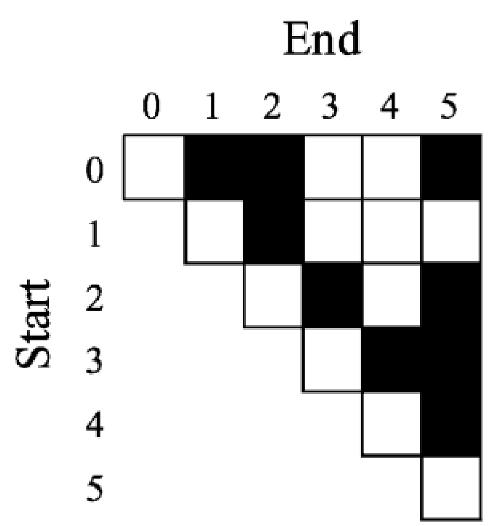
- Instead, we now predict e_{t+1}, given $m_0,\ldots,m_t$, which in turn depend upon $e_0,\ldots,e_t$:
  - $k_t = W_m m_{t-1} + W_e e_t,$
  - $\bar{s}_i^t = softmax\left(\frac{m_i k_t^T}{\sqrt{\dim(k)}}\right),$
  - $r_i^t = \frac{g_i^{t+1}}{\sum_j g_j^{t+1}} \bar{s}_i^t,$ Depends on d_{t+1}
  - $\bar{l}_t = \sum_{i=l_{t+1}}^{t-1} r_i^t \cdot m_i,$ Stick-breaking process: also depends on d_{t+1}
  - Estimate $d_{t+1} \approx ReLU\left(\widetilde{W}_d m_t + \tilde{b}_d\right),$
  - then estimate $\tilde{e}_{t+1} = \tanh(W_f \begin{bmatrix} \bar{l}_t \\ m_t \end{bmatrix} + b_f).$

CCM: brackets and spans

- Predict syntax directly, but not with trees.
- Instead, use bracket matrices and “spans,” which here consist also of yields and contexts:



(a)



(b)

Span	Label	Yield	Context
⟨0,5⟩	S	NN NNS VBD IN NN	◇ – ◇
⟨0,2⟩	NP	NN NNS	◇ – VBD
⟨2,5⟩	VP	VBD IN NN	NNS – ◇
⟨3,5⟩	PP	IN NN	VBD – ◇
⟨0,1⟩	NN	NN	◇ – NNS
⟨1,2⟩	NNS	NNS	NN – VBD
⟨2,3⟩	VBD	VBD	NNS – IN
⟨3,4⟩	IN	IN	VBD – NN
⟨4,5⟩	NN	NNS	IN – ◇

(c)

CCM: brackets and spans

- Predict syntax directly, but not with trees.
- Instead, use bracket matrices and "spans," which here consist also of yields and contexts.
- $P(S, B) = P(B)P(S|B)$
- $P(S|B) = \prod_{\langle i,j \rangle} P(\alpha_{ij}|B_{ij})P(x_{ij}|B_{ij})$
- Then, use Expectation Maximization:
  - E-step: calculate P(B|S,Θ)
  - M-step: fixing those, calculate: $\underset{\widehat{\Theta}}{\operatorname{argmax}} \sum_B P(B|S, \Theta) \log P(S, B|\widehat{\Theta})$.
- P(B) is not recalculated – it is a uniform distribution over tree-consistent bracketings.

Performance on WSJ10

Model	UF_1
LBRANCH	28.7
RANDOM	34.7
DEP-PCFG (Carroll & Charniak, 1992)	48.2
RBRANCH	61.7
CCM (Klein & Manning, 2002)	71.9
DMV+CCM (Klein & Manning, 2005)	77.6
UML-DOP (Bod, 2006)	**82.9**
PRPN	70.02
UPPER BOUND	88.1

Performance on PTB30+

Model	PTB Mean	PTB Max	CTB Mean	CTB Max
PRPN (Shen et al., 2018)	37.4	38.1	–	–
ON (Shen et al., 2019)	47.7	49.4	–	–
URNNG† (Kim et al., 2019)	–	45.4	–	–
DIORA† (Drozdov et al., 2019)	–	58.9	–	–
Left Branching	8.7		9.7	
Right Branching	39.5		20.0	
Random Trees	19.2	19.5	15.7	16.0
PRPN (tuned)	47.3	47.9	30.4	31.5
ON (tuned)	48.1	50.0	25.4	25.7
Scalar PCFG	< 35.0		< 15.0	
Neural PCFG	50.8	52.6	25.7	29.5
Compound PCFG	55.2	60.1	36.0	39.8
Oracle Trees	84.3		81.1	