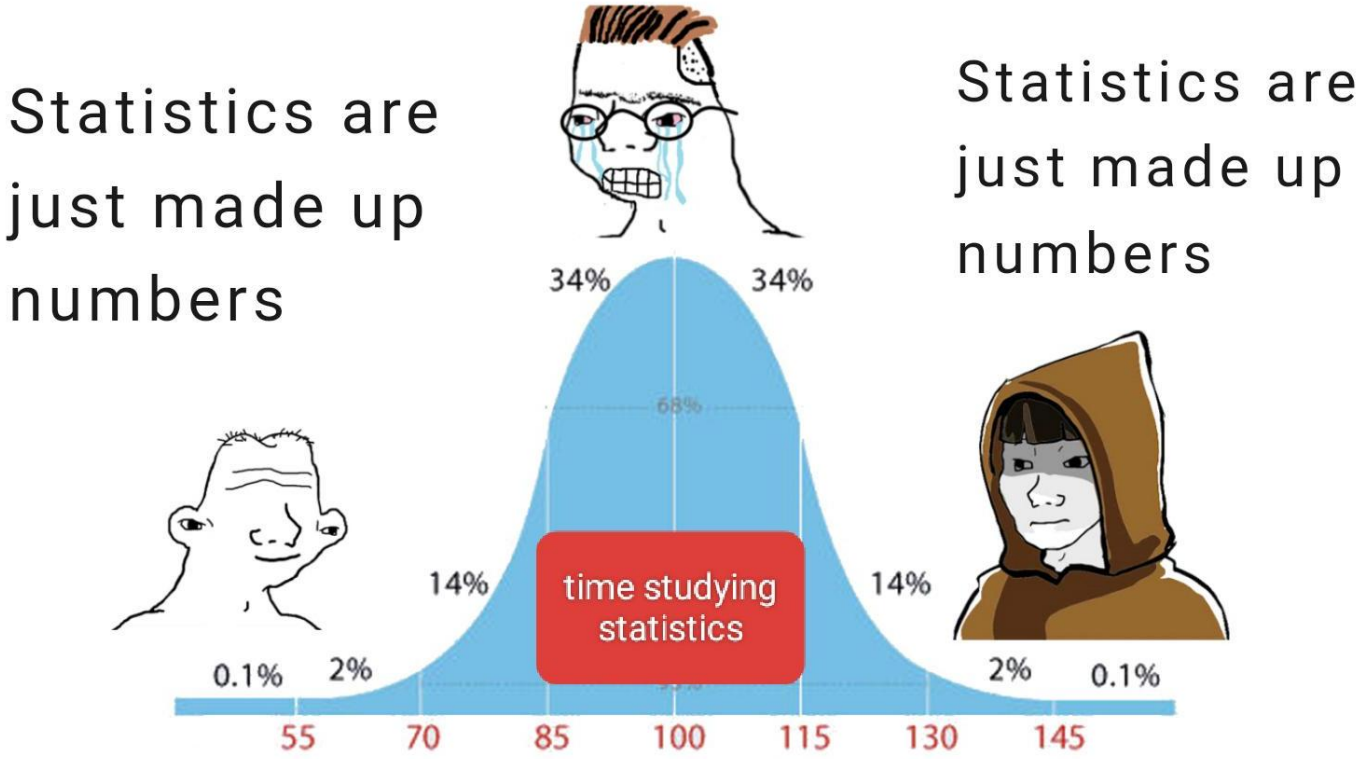
Statistics and Parsing

CSC485



Announcement

- Essay 5 for grad students posted.
- Lillian Lee, “Fast Context-Free Grammar Parsing Requires Fast Boolean Matrix Multiplication,” JACM, 49(1), 2002, 1–15.
- Due date: noon, Friday 29 November 2024.

Statistics and Parsing

- Statistical resolution of PP attachment ambiguities
- Statistical parsing
- Unsupervised parsing

Ambiguity

Recall lecture 6:

- Lexical Ambiguity
  - *The lawyer walked to the bar and addressed the jury.*
  - *The lawyer walked to the bar and ordered a beer.*
- Syntactic Ambiguity
  - Nadia saw the cop with the binoculars.
- Semantic Ambiguity
  - *Everyone here speaks two languages.*
- Pragmatic Ambiguity
  - Nadia: Do you know who's going to the party?

Ambiguity

- When all the readings make sense:
  - Develop a way to represent them all.
  - Quantifier Scope.
- When only some (typically one) reading makes sense:
  - Find a way to identify it.

Quantifier Scope Ambiguity

- A3: “Every student takes a course”
- How many readings does this sentence have?

- Every student takes a course. Gerald takes CSC401, Frank takes CSC485 and Jinman takes CSC311.
- Every student takes a course. The course is CSC485.

- How to represent the two readings?

Formal Semantics

Compositionality:

- The meaning of a complex expression is determined by the meanings of its parts and the way they are combined.
- Example: “The goalie kicked the ball” → combine meanings of “goalie,” “kick,” and “ball.”
  - In TRALE: `(… sem:goalie, …).`
- We can represent the meaning of sentences in their **logic forms**.

Logical Form

Frank screams

`Scream(Frank)`

The goalie kicked the ball

`∃x. ∃y. ( Goalie(x) ∧ Ball(y) ∧ Kicked(x,y) )`

- Predicates: Represent properties or relations (e.g., Loves(x, y)).
- Arguments: Entities participating in the relations (e.g., John, Mary).
- Quantifiers: Express quantities (e.g., every, exists).
- Connectives: Logical operations (e.g., AND (∧), OR (∨), NOT (¬)).

- **Every student takes a course.** Gerald takes CSC401, Frank takes CSC485 and Jinman takes CSC311.

$$\forall x.(\text{student}(x) \Rightarrow \exists y.(\text{course}(y) \wedge \text{take}(x, y)))$$

- **Every student takes a course.** The course is CSC485.

$$\exists y.(\text{course}(y) \wedge \forall x.(\text{student}(x) \Rightarrow \text{take}(x, y)))$$

- **Every student takes a course.** Gerald takes CSC401, Frank takes CSC485 and Jinman takes CSC311.

  $\forall x.(student(x) \Rightarrow \exists y.(course(y) \wedge take(x, y)))$

- **Every student takes a course.** The course is CSC485.

  $\exists y.(course(y) \wedge \forall x.(student(x) \Rightarrow take(x, y)))$

- **Every professor takes a course.**

  $\forall x.(professor(x) \Rightarrow \exists y.(course(y) \wedge take(x, y)))$

  $\exists y.(course(y) \wedge \forall x.(professor(x) \Rightarrow take(x, y)))$

- **Every student eats a cookie.**

  $\exists y.(cookie(y) \wedge \forall x.(student(x) \Rightarrow eat(x, y)))$

  $\forall x.(student(x) \Rightarrow \exists y.(cookie(y) \wedge eat(x, y)))$

- **Every student takes a course.** Gerald takes CSC401, Frank takes CSC485 and Jinman takes CSC311.

  $\forall x.(\text{student}(x) \Rightarrow \exists y.(\text{course}(y) \wedge \text{take}(x, y)))$

- **Every student takes a course.** The course is CSC485.

  $\exists y.(\text{course}(y) \wedge \forall x.(\text{student}(x) \Rightarrow \text{take}(x, y)))$

- **Every professor takes a course.**

  $\forall x.(\text{professor}(x) \Rightarrow \exists y.(\text{course}(y) \wedge \text{take}(x, y)))$

  $\exists y.(\text{course}(y) \wedge \forall x.(\text{professor}(x) \Rightarrow \text{take}(x, y)))$

- **Every student eats a cookie.**

  $\exists y.(\text{cookie}(y) \wedge \forall x.(\text{student}(x) \Rightarrow \text{eat}(x, y)))$

  $\forall x.(\text{student}(x) \Rightarrow \exists y.(\text{cookie}(y) \wedge \text{eat}(x, y)))$

Beta Reduction

- **Solution**: Represent every constituent with a function.

```
def f(x):
    return 2 * x

f(3)

Result: 6
```

```
(lambda x: x*2)(3)

Result: 6
```

- Do this with logical form?

```
(lambda x: 'scream(' + x + ')')('Frank')

'scream(Frank)'
```

Beta Reduction

- **Solution**: Represent every constituent with a function.
- *Frank screams:* `scream(Frank)`

- *Frank:*
  - `Frank`
- *Screams*:
  - `λx. scream(x)`
- *Frank screams:*
  - `(λx. scream(x))(Frank) = scream(Frank)`

Beta Reduction

- A function's input can also be a function.

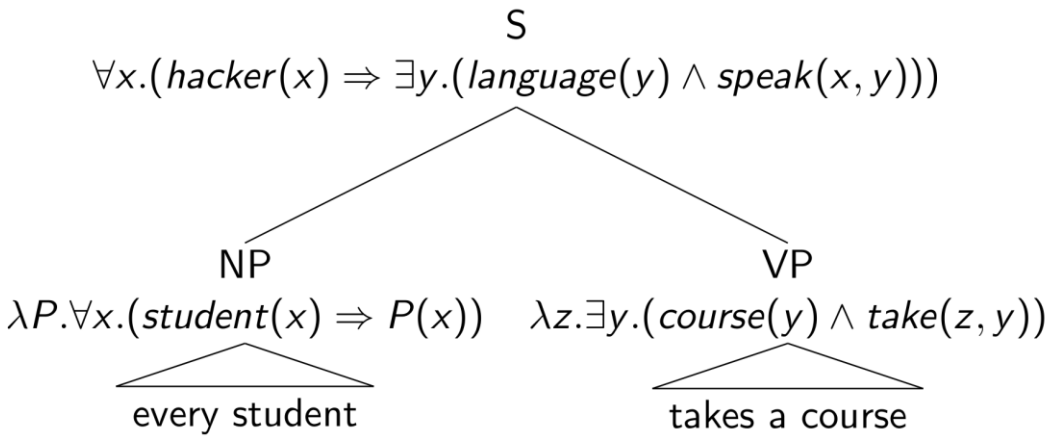
$$
\begin{aligned}
& (\lambda F.\forall y.F(y))(\lambda x.f(x)) \\
\Leftrightarrow_\beta\ & \forall y.(\lambda x.f(x))(y) \\
\Leftrightarrow_\beta\ & \forall y.f(y)
\end{aligned}
$$

```
(lambda f: f(3))(lambda x: x**2)
```

Beta Reduction: A3 Example

- Understanding the sentence as we parse it.
- Every student takes a course.
  `∀x.(hacker(x) ⇒∃y.(language(y) ∧ speak(x, y)))`

- Every student
  `λP.∀x.(student(x) ⇒ P(x))`
- takes a course
  `λz.∃y.(course(y) ∧ take(z, y))`



Beta Reduction: A3 Example

- Every student
  $\lambda P.\forall x.(\texttt{student}(x) \Rightarrow P(x))$
- takes a course
  $\lambda z.\exists y.(\texttt{course}(y) \wedge \texttt{take}(z, y))$

$$\lambda P.\forall x.(\texttt{student}(x) \Rightarrow P(x))(\lambda z.\exists y.(\texttt{course}(y) \wedge \texttt{take}(z, y)))$$
$$\Leftrightarrow_\beta \forall x.(\texttt{student}(x) \Rightarrow \lambda z.\exists y.(\texttt{course}(y) \wedge \texttt{take}(z, y))(x))$$
$$\Leftrightarrow_\beta \forall x.(\texttt{student}(x) \Rightarrow \exists y.(\texttt{course}(y) \wedge \texttt{take}(x, y)))$$

Quantifier Storage

- **Every student takes a course.** Gerald takes CSC401, Frank takes CSC485 and Jinman takes CSC311.

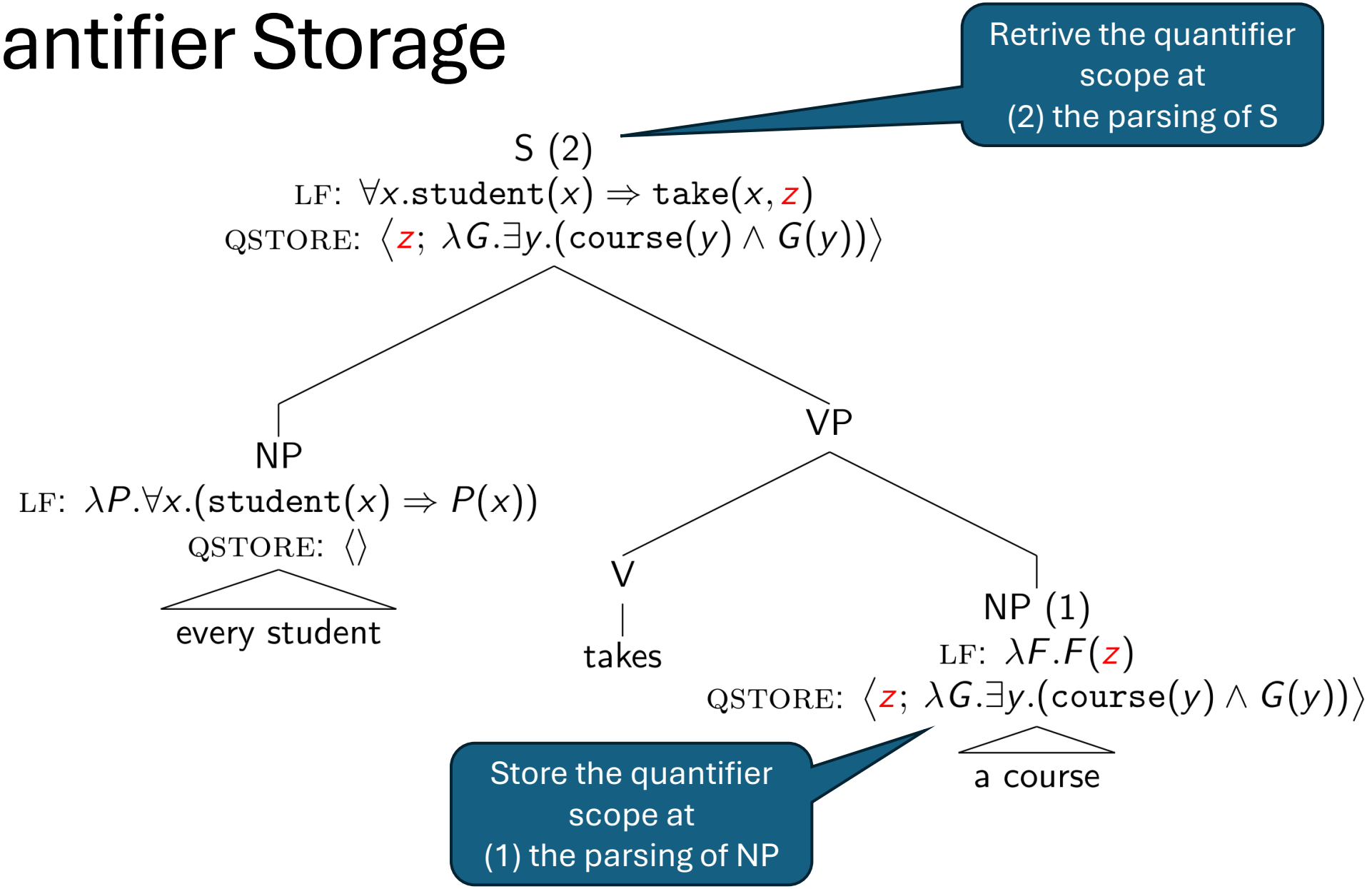
  $\forall x.(student(x) \Rightarrow \exists y.(course(y) \wedge take(x, y)))$

- **Every student takes a course.** The course is CSC485.

  $\exists y.(course(y) \wedge \forall x.(student(x) \Rightarrow take(x, y)))$

- How to get the second reading?

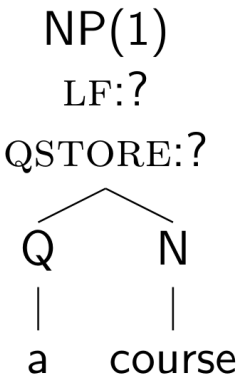
Quantifier Storage



Quantifier Storage

- Storage at (1) NP

	LF	QSTORE
1.	$\lambda G.\exists y.(\texttt{course}(y) \wedge G(y))$	$\langle\rangle$
2.	$\lambda F.F(z)$	$\langle z;\ \lambda G.\exists y.(\texttt{course}(y) \wedge G(y))\rangle$



1. Replace LF of the NP at (1) with a placeholder $\lambda F.F(z)$
2. Store the actual LF and the free variable z in qstore

- Retrieval at (2) S

	LF	QSTORE
S(2)	$\forall x.\texttt{hacker}(x) \Rightarrow \texttt{speak}(x, z)$	$\langle z;\ \lambda G.\exists y.(\texttt{language}(y) \wedge G(y))\rangle$
1.	$\lambda z.\forall x.\texttt{hacker}(x) \Rightarrow \texttt{speak}(x, z)$	$\langle z;\ \lambda G.\exists y.(\texttt{language}(y) \wedge G(y))\rangle$
2.	$(\lambda G.\exists y.(\texttt{language}(y) \wedge G(y)))$ $(\lambda z.\forall x.\texttt{hacker}(x) \Rightarrow \texttt{speak}(x, z))$	$\langle\rangle$
3.	$\exists y.(\texttt{language}(y) \wedge \forall x.(\texttt{hacker}(x) \Rightarrow \texttt{speak}(x, y)))$	$\langle\rangle$

1. We construct a function $\lambda z.L_S$, where L_S is the current LF, and z is the variable paired in the qstore entry.
2. Then, we apply this function to the LF from the qstore entry.
3. Finally, we beta normalise. Using beta normalisation, we obtain the second reading of the sentence.

Quiz 11

- Complete this beta reduction.

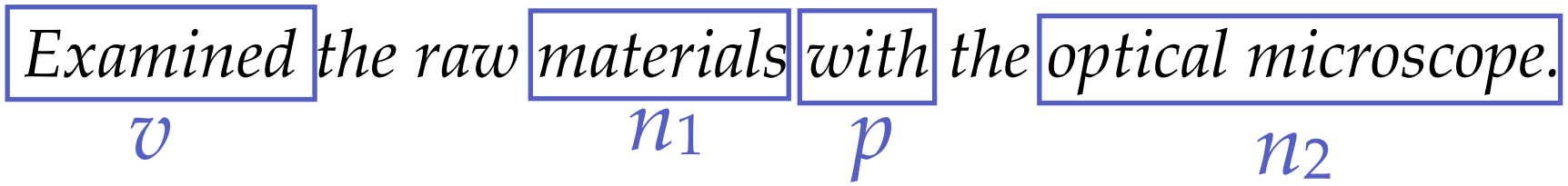
$$(\lambda f.\lambda x.(f(x))(x+1))(\lambda y.\lambda z.y^z)(2)$$

Ambiguity

- When all the readings make sense:
  - Develop a way to represent them all.
  - Quantifier Scope.
- **When only some (typically one) reading makes sense:**
  - **Find a way to identify it.**

Statistical PP attachment methods

- A classification problem.
- Input: *verb*, $noun_1$, *preposition*, $noun_2$
  Output: *V*-attach or *N*-attach
- Example:

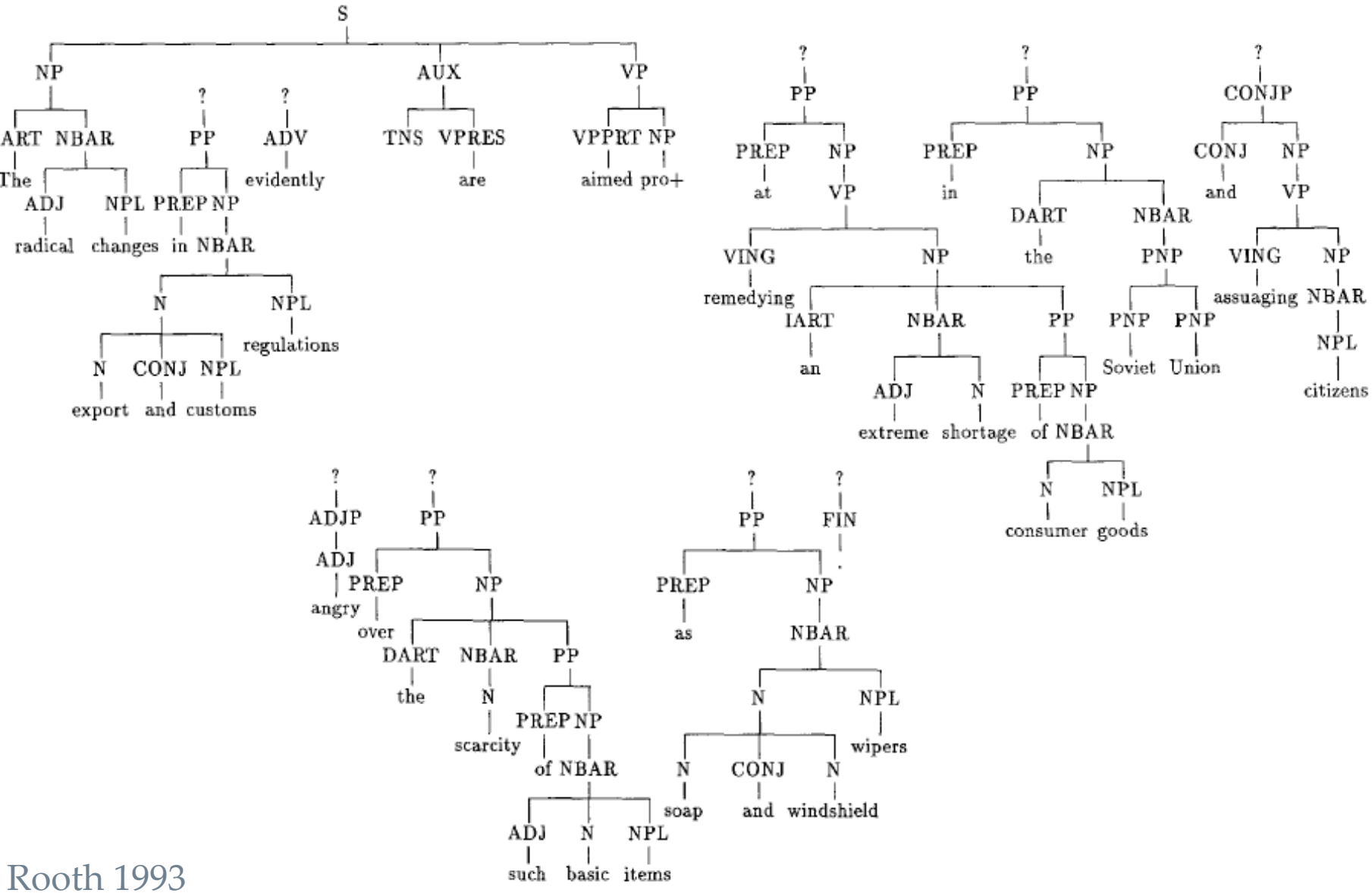


- Does not cover all PP problems.

Hindle & Rooth 1993: Input

- **Corpus:** *Partially parsed* news text.
- "Partially parsed":
  - Automatic.
  - Many attachment decisions punted.
  - A collection of parse fragments for each sentence.

The radical changes in export and customs regulations evidently are aimed at remedying an extreme shortage of consumer goods in the Soviet Union and assuaging citizens angry over the scarcity of such basic items as soap and windshield wipers.



From Hindle & Rooth 1993

Hindle & Rooth 1993: Input

- **Data:** [*v*,*n*,*p*] triples; *v* or *p* may be null; *v* may be –.

The radical changes in export and customs regulations evidently are aimed at remedying an extreme shortage of consumer goods in the Soviet Union and assuaging citizens angry over the scarcity of such basic items as soap and windshield wipers.

v	*n*	*p*
–	change	in
aim	PRO	at
remedy	shortage	of
NULL	good	in
assuage	citizen	*NULL*
NULL	scarcity	of

P attached to v or n?

Hindle & Rooth 1993: Algorithm

- Idea: Compute **lexical associations** (LAs) between p and each of v, n.
  — Is the p more associated with the v or with the n?
- Learn a way to compute LA for each [v,n,p] triple.
- Use to map from [v,n,p] to {*V*-attach, *N*-attach}.

Hindle & Rooth 1993: Algorithm

Method: Bootstrapping.

1. Label unambiguous cases as N- or V-attach:
   When v is NULL, n is pronoun, or p is of.
2. Iterate (until nothing changes):
   a. Compute LA score for each triple from data labelled so far.
   b. Label the attachment of any new triples whose score is over threshold.
3. Deal with “leftovers” (random assignment).

Test cases: Compute the LA score (or fail).

Hindle & Rooth 1993: Algorithm

- **Lexical association** score: log-likelihood ratio of verb- and noun-attachment.

$$LA(v, n, p) = \log_2 \frac{P(V\text{-attach } p|v, n)}{P(N\text{-attach } p|v, n)}$$

- Can't get these probabilities directly — data are too sparse.
- So, estimate them from the data that we can get.

Hindle & Rooth 1993: Algorithm

- **Lexical association** score: log-likelihood ratio of verb- and noun-attachment.

$$LA(v,n,p) = \log_2 \frac{P(V\text{-attach } p|v,n)}{P(N\text{-attach } p|v,n)}$$

$$\approx P(V\text{-attach } p|v)P(null|n) \qquad \approx P(N\text{-attach } p|n)$$

Hindle & Rooth 1993: Example

Moscow sent more than 100,000 soldiers into Afghanistan …

- Choose between:

  V-attach: [VP *send* [NP … *soldier NULL*] [PP *into*…]]

  N-attach: [VP *send* [NP … *soldier* [PP *into*…]]…]

Hindle & Rooth 1993: Example

1. P(*V*-attach *into* | *send*, *soldier*)

   ≈ P(*V*-attach *into* | *send*) · P(*NULL* | *soldier*)

$$\frac{c(send, into)}{c(send)} = .049 \qquad \frac{c(soldier, NULL)}{c(soldier)} = .800$$

2. P(*N*-attach *into* | *send*, *soldier*)

   ≈ P(*N*-attach *into* | *soldier*)

$$\frac{c(soldier, NULL)}{c(soldier)} = .800$$

$$LA(send, soldier, into) = \log_2(.049 \times .800/.0007) \approx 5.81$$

Hindle & Rooth 1993: Results

- Training: 223K triples
  Testing: 1K triples
  Results: 80% accuracy
  (Baselines: 66% by noun attachment; 88% by humans.)

Hindle & Rooth 1993: Discussion

- **Advantages**: Unsupervised?; gives degree of preference.
- **Disadvantages**: Needs lots of partially parsed data. Other words don't get a vote.
- **Importance** to CL:
  - Use of large amounts of unlabelled data, with clever application of linguistic knowledge, to learn useful statistics.

Brill & Resnik 1994: Method

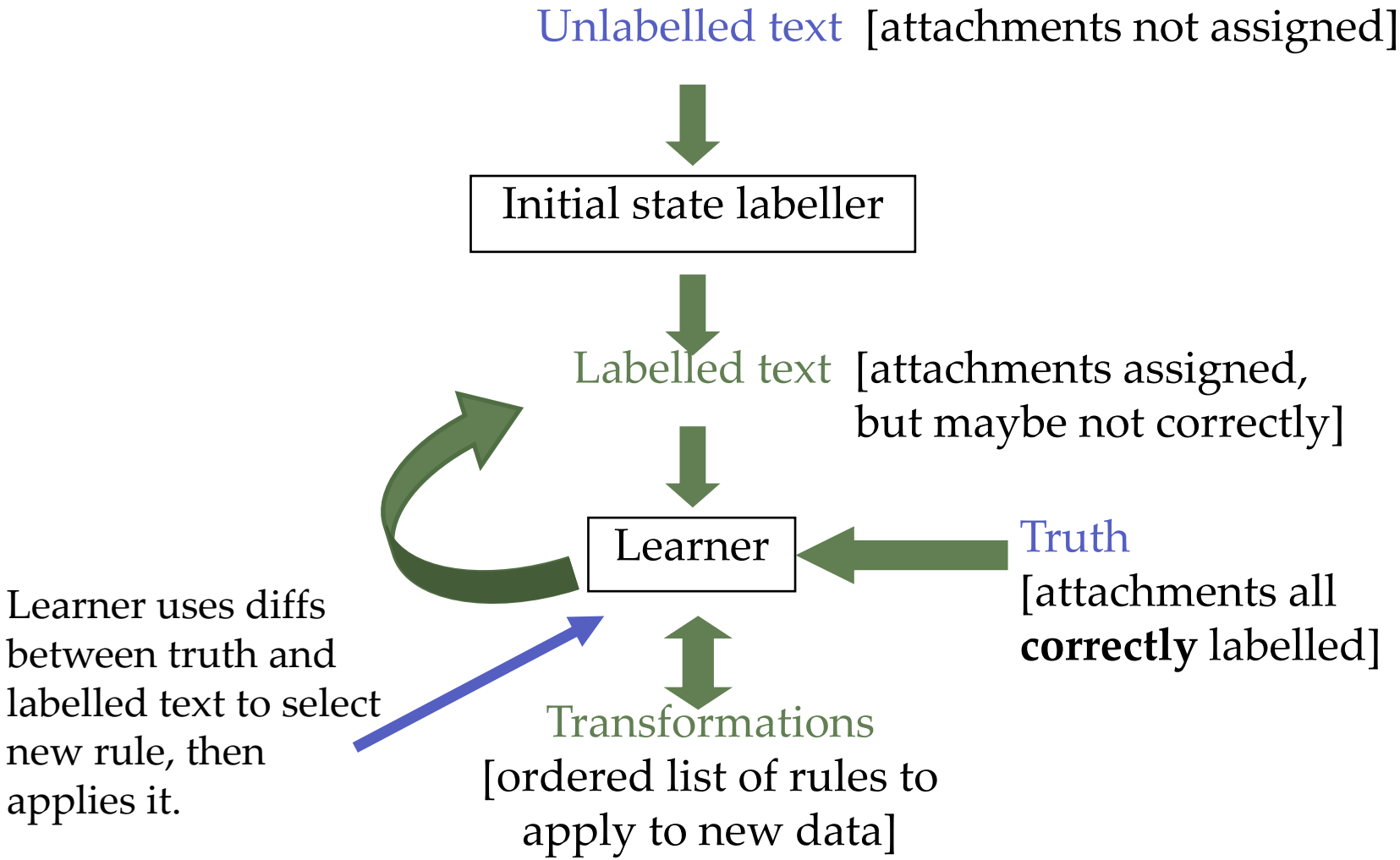
- Corpus-based, non-statistical method.
- Transformation-based learning: Learns sequence of rules to apply to each input item.
- Form of transformation rules:
  - Flip attachment decision (from V to N_1 or vice versa) if $\{v, n_1, p, n_2\}$ is w_1 [and $\{v, n_1, p, n_2\}$ is w_2].

A quad: Uses head noun of PP too

Optional conjunct

- All rules apply, in order in which they are learned.

Brill & Resnik 1994: Method



Brill & Resnik 1994: Example

Some rules learned:

Start by assuming N_1 attachment, and then change attachment …

1. from N_1 to V if p is at.
2. from N_1 to V if p is as.

⋮

6. from N_1 to V if $n2$ is year.
8. from N_1 to V if p is in and n_1 is amount.

⋮

15. from N_1 to V if v is $have$ and p is in.
17. from V to $N1$ if p is of.

Brill & Resnik 1994: Results

- Training: 12K annotated quads
  Testing: 500 quads
  Results: 80% accuracy
  (Baseline: 64% by noun attachment)

Brill & Resnik 1994: Discussion

- **Advantages**: Readable rules (but may be hard); can build in bias in initial annotation; small number of rules.
- **Disadvantages**: Supervised; no strength of preference. Very memory-intensive.
- **Importance to CL**:
  - Successful general method for non-statistical learning from annotated corpus.
  - Based on popular (and relatively easily modified) tagger.

Since then…

- Modestly better methods exist (e.g., Ratnaparkhi 1998; Belinkov et al. 2014) that leverage:
  - large amounts of noisy, unannotated data (most of the partial parses were not being used anyway)
  - early attempts such as Hindle & Rooth 1993, where they are known to be very accurate
  - vector-based language models (neural methods for English?)
- …but the field mostly lost interest when it emerged that parsing decisions could be made with the assistance of language models.

Since then…

- Modestly better methods exist (e.g., Ratnaparkhi 1998; Belinkov et al. 2014).
- … but the field mostly lost interest when it emerged that parsing decisions could be made with the assistance of language models:
  - Far more context taken into account
  - Much better numbers (but lots of easy decisions folded in that inflate these – PP attachment now in high 80s)
  - PP attachment still very important for FWO languages (Do & Rehbein 2020).

Statistics and Parsing

- Statistical resolution of PP attachment ambiguities
- **Statistical parsing**
- Unsupervised parsing

Statistical Parsing

- ***General idea***:
  - Assign probabilities to rules in a context-free grammar.
    - Use a likelihood model.
  - Combine probabilities of rules in a tree.
    - Yields likelihood of a parse.
  - The best parse is the most likely one.

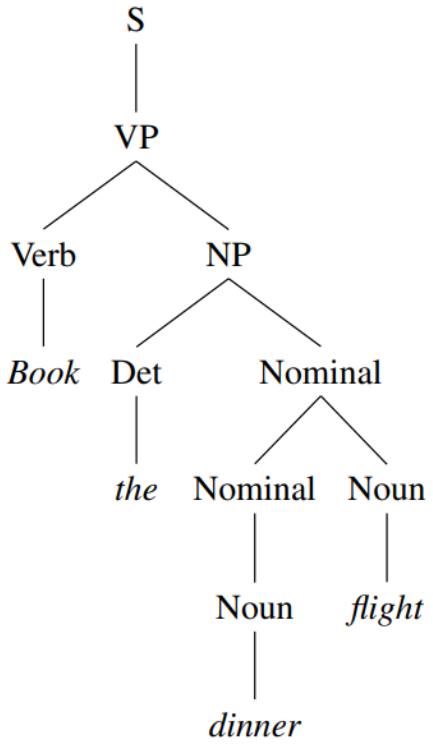
Statistical Parsing

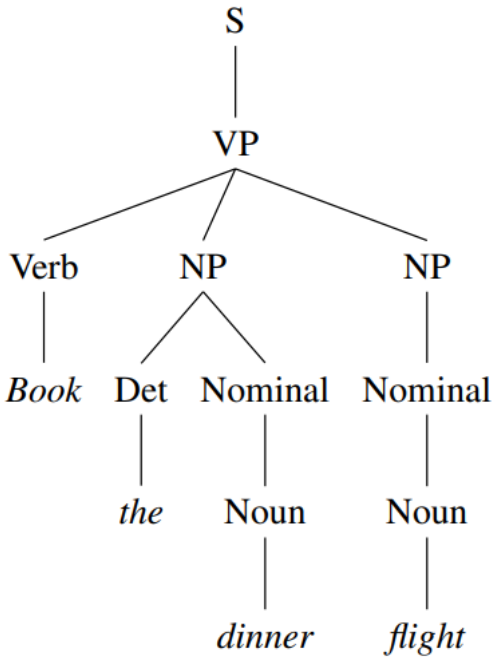
- PCFG Example

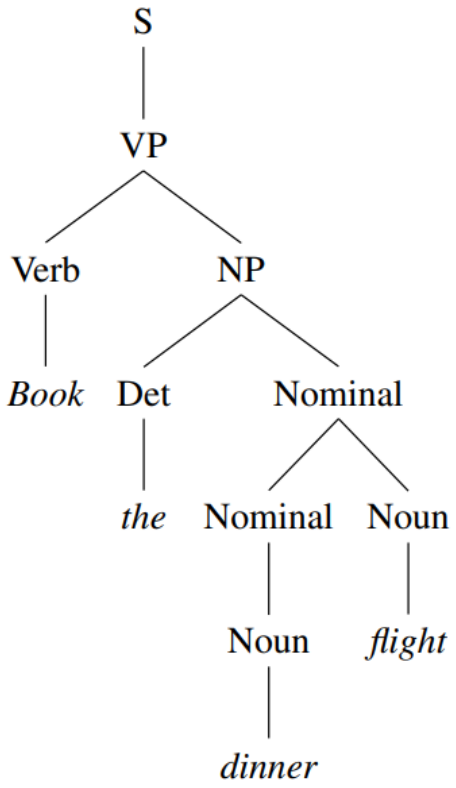
Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.10] | a [.30] | the [.60]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$ [.10] | $trip$ [.30]
$S \rightarrow VP$	[.05]	| $meal$ [.05] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flight$ [.40] | $dinner$ [.10]
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book$ [.30] | $include$ [.30]
$NP \rightarrow Det\ Nominal$	[.20]	| $prefer$ [.40]
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I$ [.40] | she [.05]
$Nominal \rightarrow Noun$	[.75]	| me [.15] | you [.40]
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston$ [.60]
$Nominal \rightarrow Nominal\ PP$	[.05]	| NWA [.40]
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does$ [.60] | can [.40]
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from$ [.30] | to [.30]
$VP \rightarrow Verb\ NP\ PP$	[.10]	| on [.20] | $near$ [.15]
$VP \rightarrow Verb\ PP$	[.15]	| $through$ [.05]
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	

Statistical Parsing

- PCFG for disambiguation:
- Example: *Book the dinner flight*



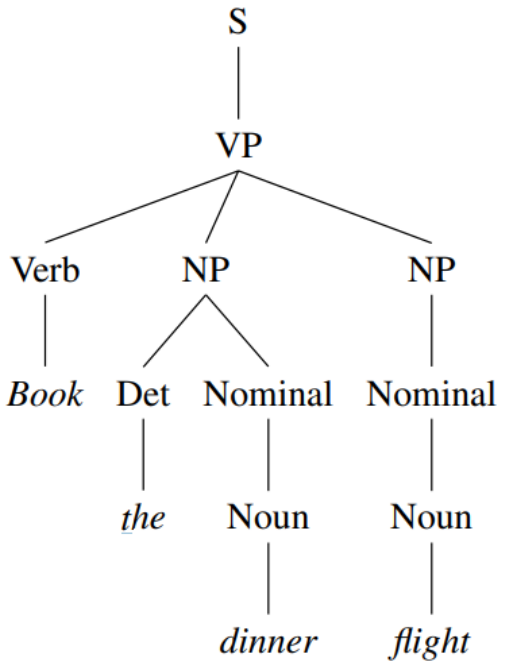




Rules			P
S	→	VP	.05
VP	→	Verb NP	.20
NP	→	Det Nominal	.20
Nominal	→	Nominal Noun	.20
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flight	.40

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

Rules			P
S	→	VP	.05
VP	→	Verb NP NP	.10
NP	→	Det Nominal	.20
NP	→	Nominal	.15
Nominal	→	Noun	.75
Nominal	→	Noun	.75
Verb	→	book	.30
Det	→	the	.60
Noun	→	dinner	.10
Noun	→	flight	.40



$$P(T_{right}) = .05 * .10 * .20 * .15 * .75 * .75 * .30 * .60 * .10 * .40 = \mathbf{6.1 \times 10^{-7}}$$

Statistical Parsing

- Motivations:
  - Uniform process for attachment decisions.
  - Use lexical preferences in all decisions.

General Approaches

1. Assign a probability to each rule of grammar, including lexical productions.
   - Parse string of input words with probabilistic rules.
     The can will rust.

2. Assign probabilities only to non-lexical productions.
   - Probabilistically tag input words with syntactic categories using a **part-of-speech tagger**.
   - Consider the pre-terminal syntactic categories to be terminals, parse that string with probabilistic rules.
     Det N Modal Verb.

Statistical chart parsing

- Consider tags as terminals
  (i.e., use a PoS tagger to pre-process input texts).
  Det N Modal Verb.
- For probability of each grammar rule, use MLE.
- Probabilities derived from hand-parsed corpora (treebanks).
- Count frequency of use c of each rule, for each non-terminal C and each different RHS.
- What are some problems with this approach?

Statistical chart parsing

- Consider tags as terminals
  (i.e., use a PoS tagger to pre-process input texts).
  Det N Modal Verb.
- For probability of each grammar rule, use MLE.
- Probabilities derived from hand-parsed corpora (treebanks).
- Count frequency of use c of each rule, for each non-terminal C and each different RHS.
- What are some problems with this approach?
  - Sparsity
  - Tied to the grammar of the original treebank.

Statistical chart parsing

- MLE probability of rules:
  - For each rule: $C \rightarrow \alpha$

$$P(C \rightarrow \alpha | C) = \frac{c(C \rightarrow \alpha)}{\sum_{\beta} c(C \rightarrow \beta)} = \frac{c(C \rightarrow \alpha)}{c(C)}$$

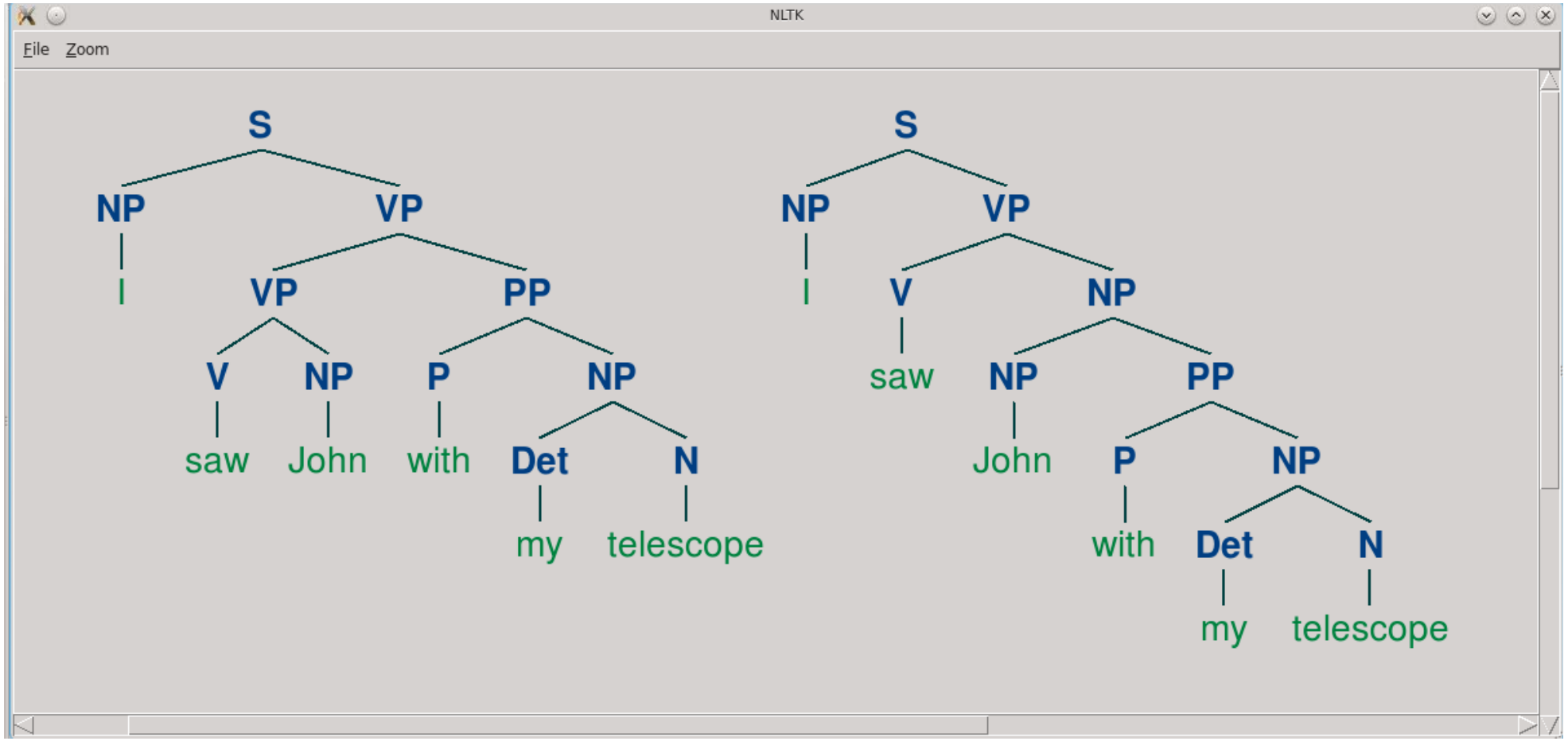
- Takes no account of the context of use of a rule: ***independence assumption***.
- ***Source-normalized***: assumes a top-down generative process.
- NLTK's pchart demo doesn't POS-tag first (words are generated top-down), and it shows $\mathrm{P(t)}$ rather than $\mathrm{P(t|s)}$'.

```
>>> import nltk
>>> nltk.parse.pchart.demo()

  1: I saw John with my telescope
     <Grammar with 17 productions>

  2: the boy saw Jack with Bob under the table with a telescope
     <Grammar with 23 productions>

Which demo (1-2)? 1

s: I saw John with my telescope
parser: <nltk.parse.pchart.InsideChartParser object at 0x7f61288f3290>
grammar: Grammar with 17 productions (start state = S)
    S -> NP VP [1.0]
    NP -> Det N [0.5]
    NP -> NP PP [0.25]
    NP -> 'John' [0.1]
    NP -> 'I' [0.15]
    Det -> 'the' [0.8]
    Det -> 'my' [0.2]
    N -> 'man' [0.5]
    N -> 'telescope' [0.5]
    VP -> VP PP [0.1]
    VP -> V NP [0.7]
    VP -> V [0.2]
    V -> 'ate' [0.35]
    V -> 'saw' [0.65]
    PP -> P NP [1.0]
    P -> 'with' [0.61]
    P -> 'under' [0.39]
```

```
|[-] . . . . .| [0:1] 'I'                    [1.0]
|. [-] . . . .| [1:2] 'saw'                  [1.0]
|. . [-] . . .| [2:3] 'John'                 [1.0]
|. . . [-] . .| [3:4] 'with'                 [1.0]
|. . . . [-] .| [4:5] 'my'                   [1.0]
|. . . . . [-]| [5:6] 'telescope'            [1.0]
|. . . . . [-]| [5:6] 'telescope'            [1.0]
|. . . . [-] .| [4:5] 'my'                   [1.0]
|. . . [-] . .| [3:4] 'with'                 [1.0]
|. . [-] . . .| [2:3] 'John'                 [1.0]
|. [-] . . . .| [1:2] 'saw'                  [1.0]
|[-] . . . . .| [0:1] 'I'                    [1.0]
|. [-] . . . .| [1:2] V  -> 'saw' *          [0.65]
|. > . . . . .| [1:1] VP -> * V NP           [0.7]
|. > . . . . .| [1:1] V  -> * 'saw'         [0.65]
|. . . [-] . .| [3:4] P  -> 'with' *         [0.61]
|. . . > . . .| [3:3] PP -> * P NP           [1.0]
|. . . [-> . .| [3:4] PP -> P * NP           [0.61]
|. . . > . . .| [3:3] P  -> * 'with'         [0.61]
|. . . . . [-]| [5:6] N  -> 'telescope' *    [0.5]
|. . . . . > .| [5:5] N  -> * 'telescope'    [0.5]
|. [-> . . . .| [1:2] VP -> V * NP           [0.455]
|. > . . . . .| [1:1] VP -> * V              [0.2]
|. . . . [-] .| [4:5] Det -> 'my' *          [0.2]
|. . . . > . .| [4:4] NP -> * Det N          [0.5]
|. . . . > . .| [4:4] Det -> * 'my'          [0.2]
```

```
|. . . . > . .| [4:4] S  -> * NP VP          [1.0]
|. . . . > . .| [4:4] NP -> * NP PP          [0.25]
|. . . . [--->| [4:6] S  -> NP * VP          [0.05]
|. [---] . . .| [1:3] VP -> V NP *           [0.0455]
|[-> . . . . .| [0:1] NP -> NP * PP          [0.0375]
|. . . [-----]| [3:6] PP -> P NP *           [0.0305]
|. . [-> . . .| [2:3] NP -> NP * PP          [0.025]
|[---] . . . .| [0:2] S  -> NP VP *          [0.0195]
|. [-> . . . .| [1:2] VP -> VP * PP          [0.013]
|. . . . [--->| [4:6] NP -> NP * PP          [0.0125]
|[-----] . . .| [0:3] S  -> NP VP *          [0.006825]
|. [---> . . .| [1:3] VP -> VP * PP          [0.00455]
|. . [-------]| [2:6] NP -> NP PP *          [0.0007625]
|. . [------->| [2:6] S  -> NP * VP          [0.0007625]
|. [---------]| [1:6] VP -> V NP *           [0.0003469375]
|. . [------->| [2:6] NP -> NP * PP          [0.000190625]
|. [---------]| [1:6] VP -> VP PP *          [0.000138775]
|[===========]| [0:6] S  -> NP VP *          [5.2040625e-05]
|. [--------->| [1:6] VP -> VP * PP          [3.469375e-05]
|[===========]| [0:6] S  -> NP VP *          [2.081625e-05]
|. [--------->| [1:6] VP -> VP * PP          [1.38775e-05]
```

```
Draw parses (y/n)? y
    please wait...
```



Statistical chart parsing

- In this view of chart parsing, probability of chart entries is relatively simple to calculate. For completed constituents, maximize over C_1, ..., C_n:

$$\begin{aligned} P(e_0) &= P(C_0 \rightarrow C_1 \ldots C_n \mid C_0) \times P(e_1) \times \cdots \times P(e_n) \\ &= P(C_0 \rightarrow C_1 \ldots C_n \mid C_0) \times \prod_{i=1}^{n} P(e_i) \end{aligned}$$

- e_0 is the entry for current constituent, of category C_0;
- e_1 ... e_n are chart entries for C_1 ... C_n in the RHS of the rule.
- NB: Unlike for PoS tagging above, the C_i are not necessarily lexical categories.

Statistical chart parsing

- Consider a complete parse tree, t, with root label S.
- Recasting ❺, t has the probability:

$$P(t) = P(S) * \Pi_n P(rule(n)|cat(n))$$ ❻

  where n ranges over all nodes in the tree t;

  $rule(n)$ is the rule used for n;

  $cat(n)$ is the category of n.

- $P(S) = 1!$
- "Bottoms out" at lexical categories.
- Note that we're parsing bottom-up, but the generative model "thinks" top-down regardless.

Statistical chart parsing

- But just like non-statistical chart parsers, this one only answers “yes” or “no” (with a probability) in polynomial time:
  - It’s not supposed to matter how we got each constituent. Just the non-terminal label and the span are all that should matter.
- There might be exponentially many trees in this formulation.
- And we’re not calculating the probability that the input is a sentence – this is only the probability of one interpretation (tree).

Announcement

- Extra zoom office hours:
  - Saturday 2-4: Jinyue.
  - Sunday 10-12: Frank.
- We will post zoom link soon on Piazza.

Evaluation

- Evaluation method:
  - Train on part of a parsed corpus.
    (I.e., gather rules and statistics.)
  - Test on a different part of the corpus.
    - Development test: early stopping, meta-parameters
    - Evaluation test: evaluate (and then done)
- In one sense, the best evaluation of a method like this would be data likelihood, but since we're scoring trees instead of strings, it's difficult to defend any sort of intuition about the numbers assigned to them.

Evaluation

- Evaluation: PARSEVAL measures compare parser output to known correct parse:
  - Labelled ***precision***, labelled ***recall***.

Fraction of constituents in output that are correct.

Fraction of correct constituents in output.

  - ***F-measure*** = harmonic mean of precision and recall
    = $2PR / (P + R)$

Evaluation

- **Evaluation**: PARSEVAL measures compare parser output to known correct parse:
  - Penalize for **cross-brackets** per sentence:
    Constituents in output that overlap two (or more) correct ones;
    e.g., `[[A B] C]` for `[A [B C]]`.

    ```
    [[Nadia] [[smelled] [the eggplant]]]
    [[[Nadia] [smelled]] [the eggplant]]
    ```

The labels on the subtrees aren't necessary for this one.

Evaluation

- PARSEVAL is a ***classifier accuracy*** score – much more extensional. All that matters is the right answer at the end.
- But that still means that we can look at parts of the right answer.
- Can get ~75% labelled precision, recall, and F with above methods.

https://github.com/nikitakit/self-attentive-parser

BERT-based parser

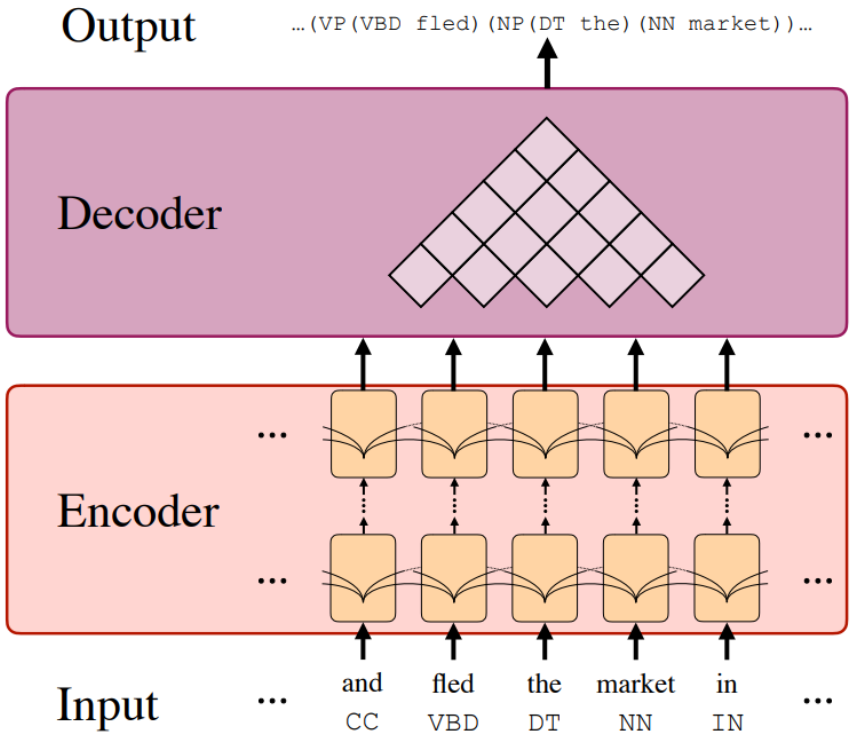


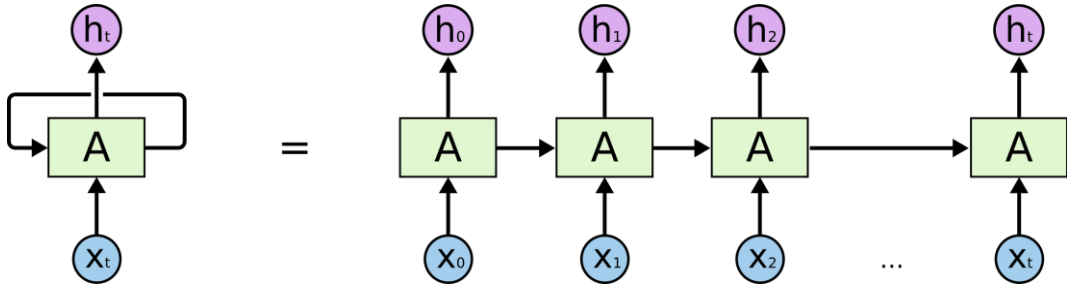
Figure 1: Our parser combines a chart decoder with a sentence encoder based on self-attention.

Berkeley Neural Parser. Kitaev and Klein. 2018. Constituency Parsing with a Self-Attentive Encoder.

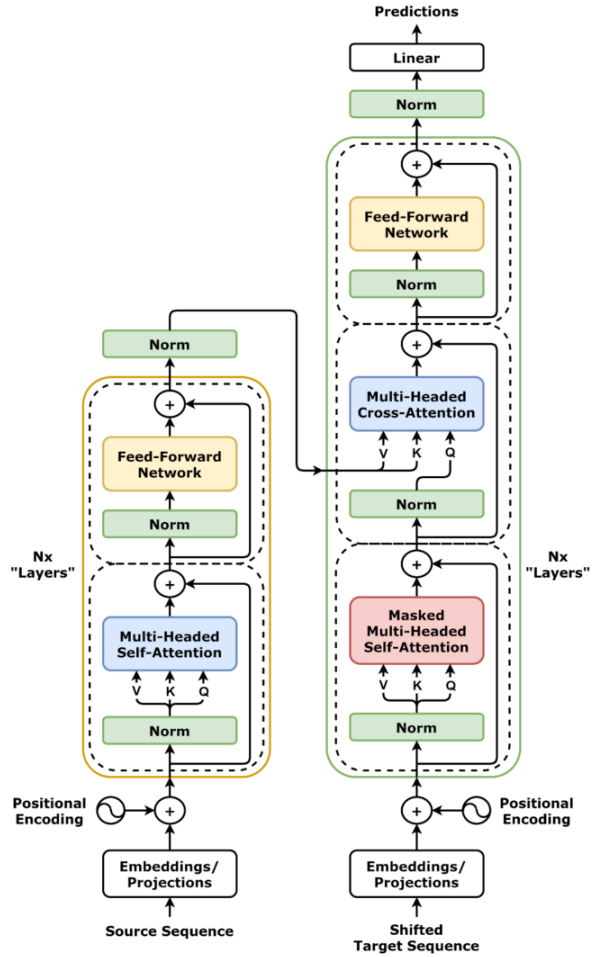
Unsupervised Parsing

- 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.

Before we start

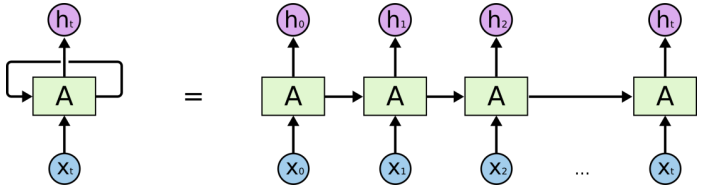


RNN

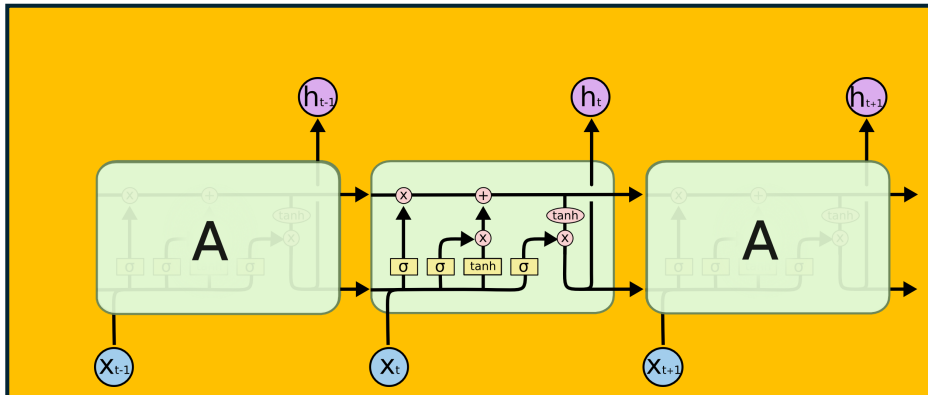


Transformers

Before we start

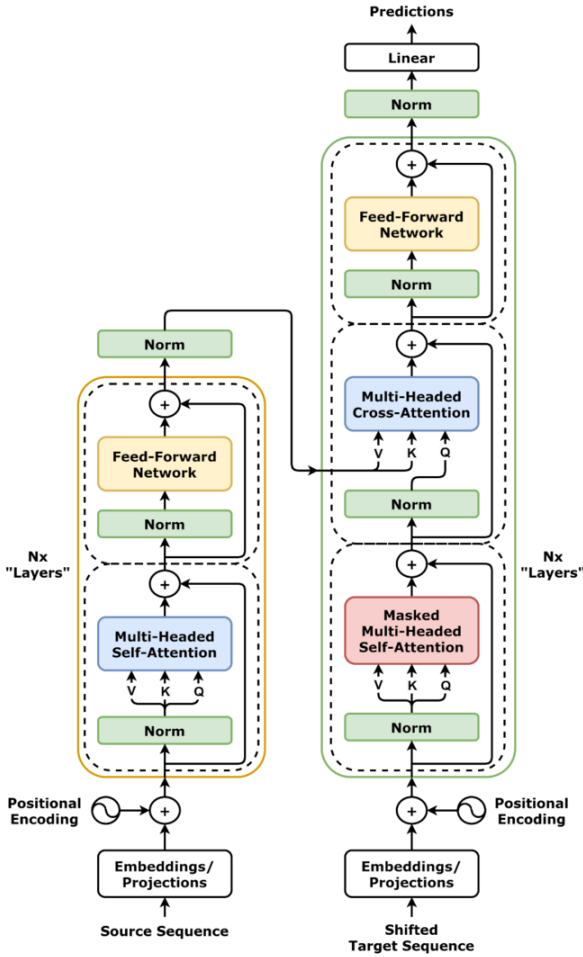


RNN



Long Short Term Memory

LSTM!



Transformers

LSTM: Crash Course

- We talked about RNN's problem: vanishing gradient.
- RNN:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b)$$

current input embedding | previous hidden state | bias

- LSTM:
  - RNN works fine if the input sentence is short
  - Learn the sentence piece by piece
    - Reset the hidden state periodically
  - … When to reset the hidden state?
  - The ML approach:
    - Let another NN model to predict it.
    - (actually, we use a module of the current model to predict it)

LSTM: Crash Course

- We talked about RNN's problem: vanishing gradient.
- RNN:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b)$$

current input embedding — previous hidden state — bias

- LSTM:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

Nothing fancy, just renaming "hidden" to "output"

LSTM: Crash Course

- We talked about RNN's problem: vanishing gradient.
- RNN:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b)$$

current input embedding — previous hidden state — bias

- LSTM:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \cdot \sigma(c_t)$$

The "real" hidden state is just the output state…
With a few things forget.
"\cdot" is element-wise multiplication

$\sigma(c_t)$ ranges from 0-1

LSTM: Crash Course

- We talked about RNN's problem: vanishing gradient.
- RNN:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b)$$

current input embedding — previous hidden state — bias

- LSTM:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \cdot \sigma(c_t)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t$$

$$\tilde{c}_t = \sigma(W_c x_t + U_c h_{t-1} + b_c)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_i + U_f h_{t-1} + b_i)$$

Cell state: the component tell the model what to forget.

Forget gate: what information to wipe from the previous cell state.

Input gate: what information to keep from the current cell state.

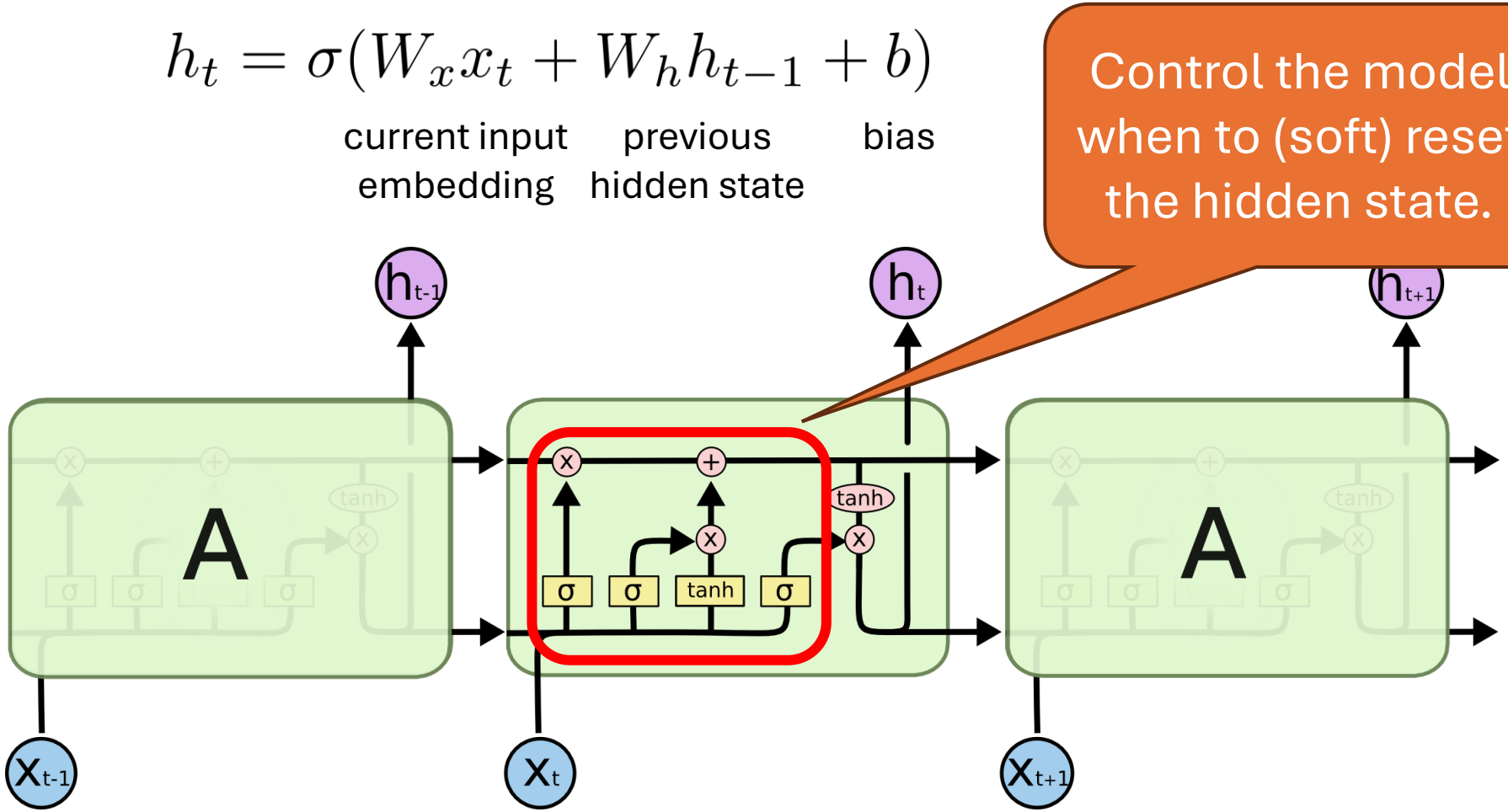
LSTM: Crash Course

- We talked about RNN's problem: vanishing gradient.
- RNN:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b)$$

current input embedding | previous hidden state | bias

- LSTM:

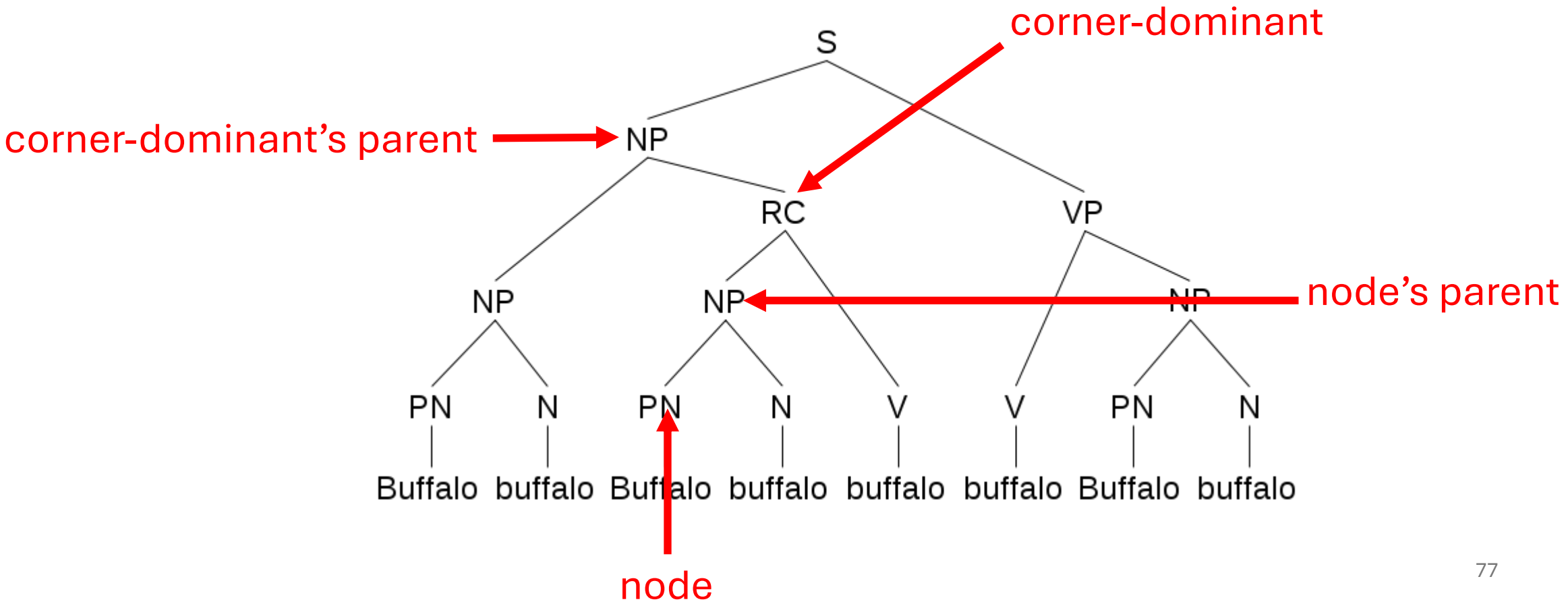


PRPN: parse-read-predict

- 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.

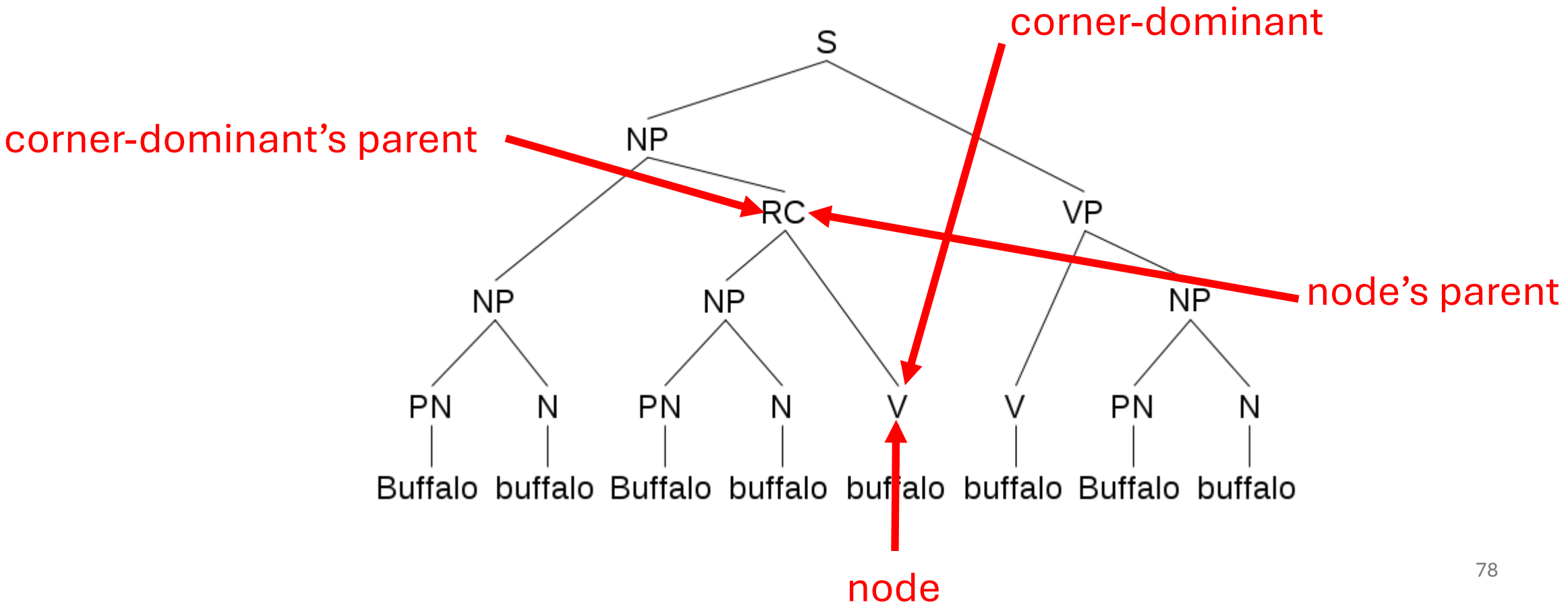
Some terminology

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



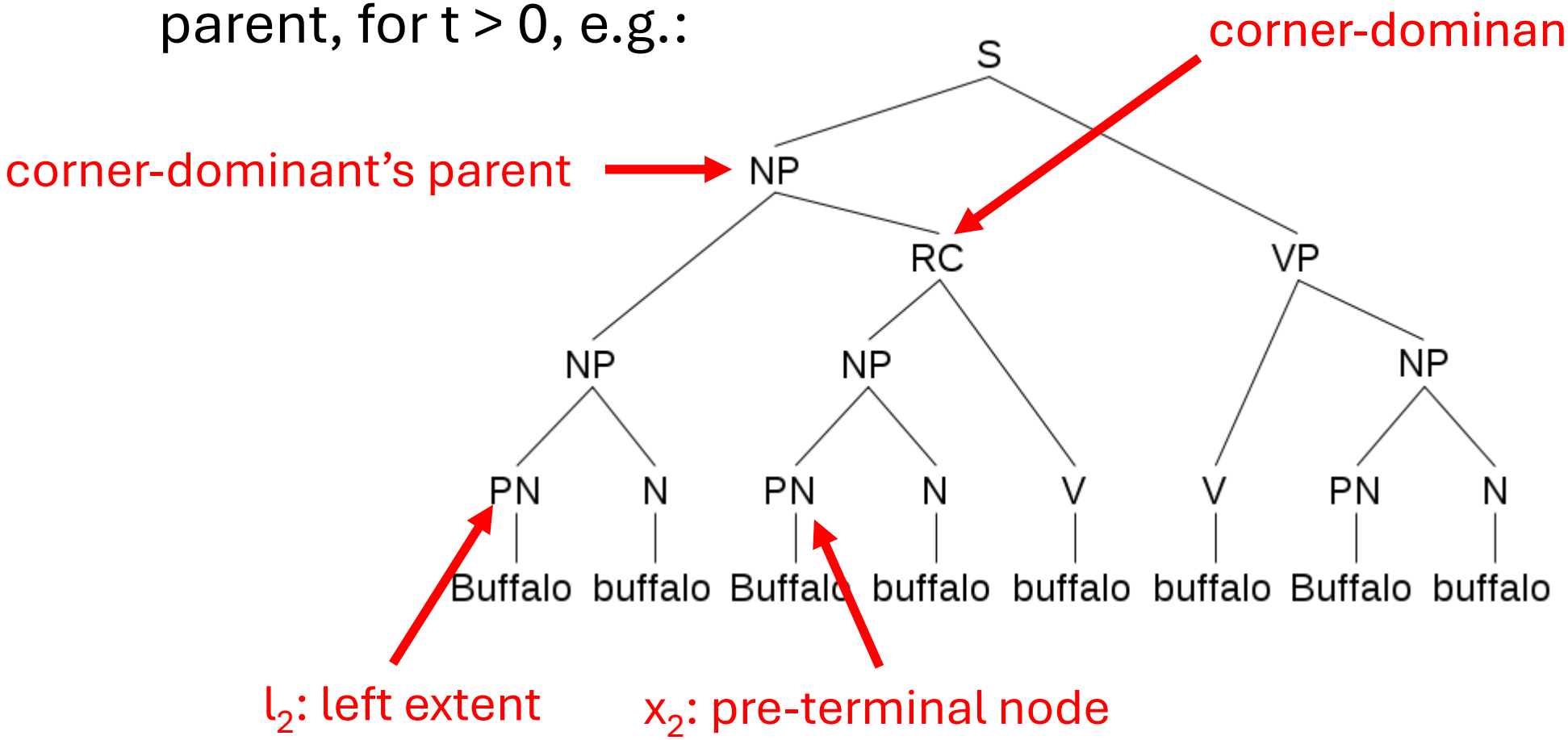
Some terminology

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



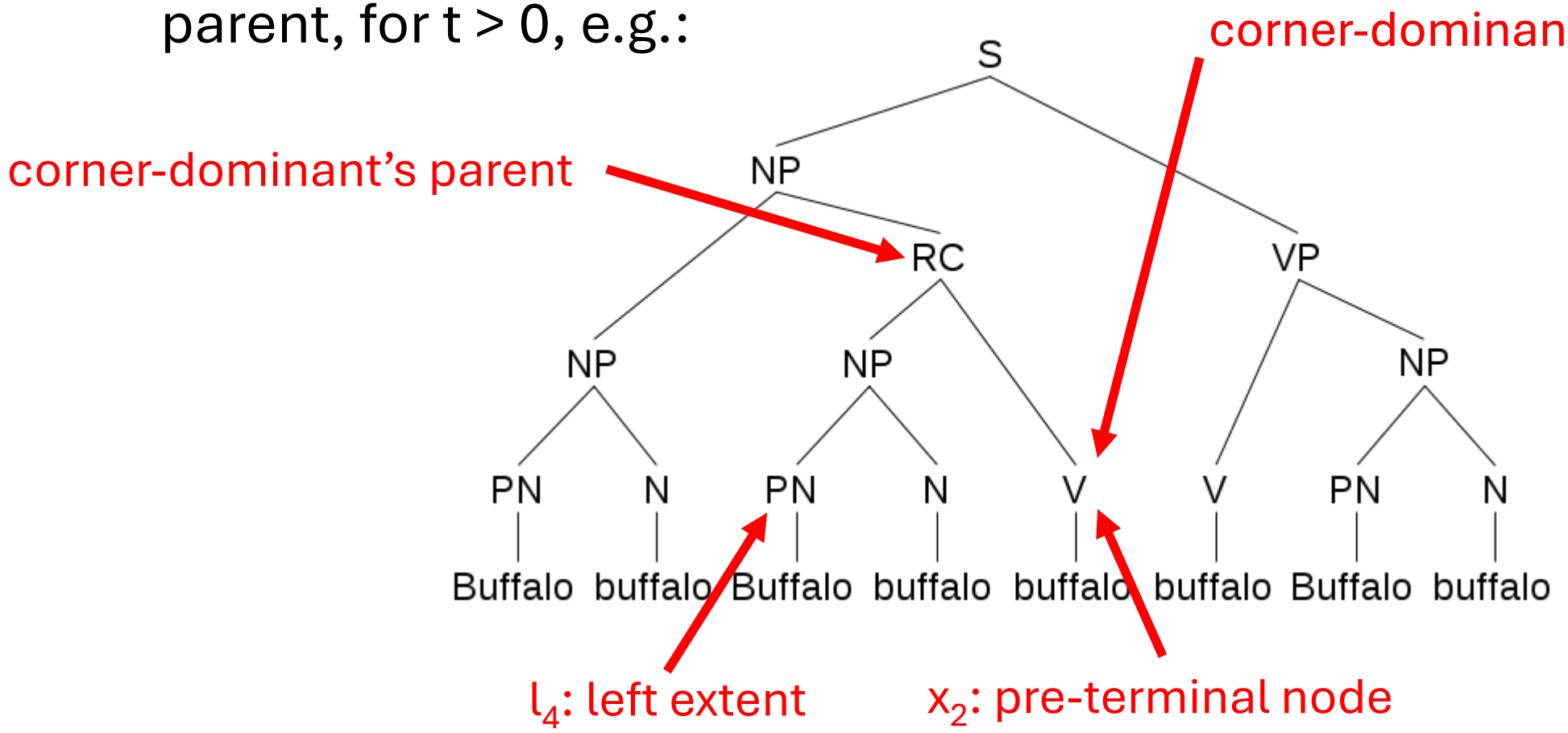
Some terminology

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



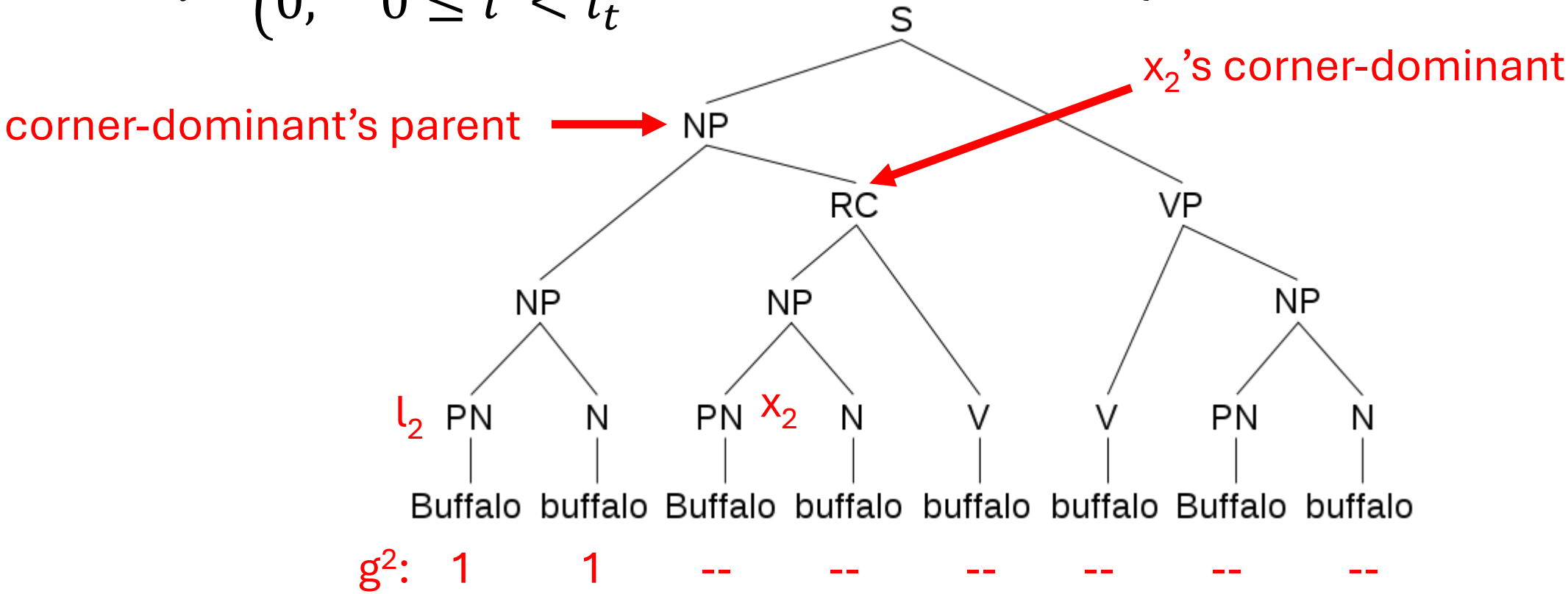
Some terminology

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



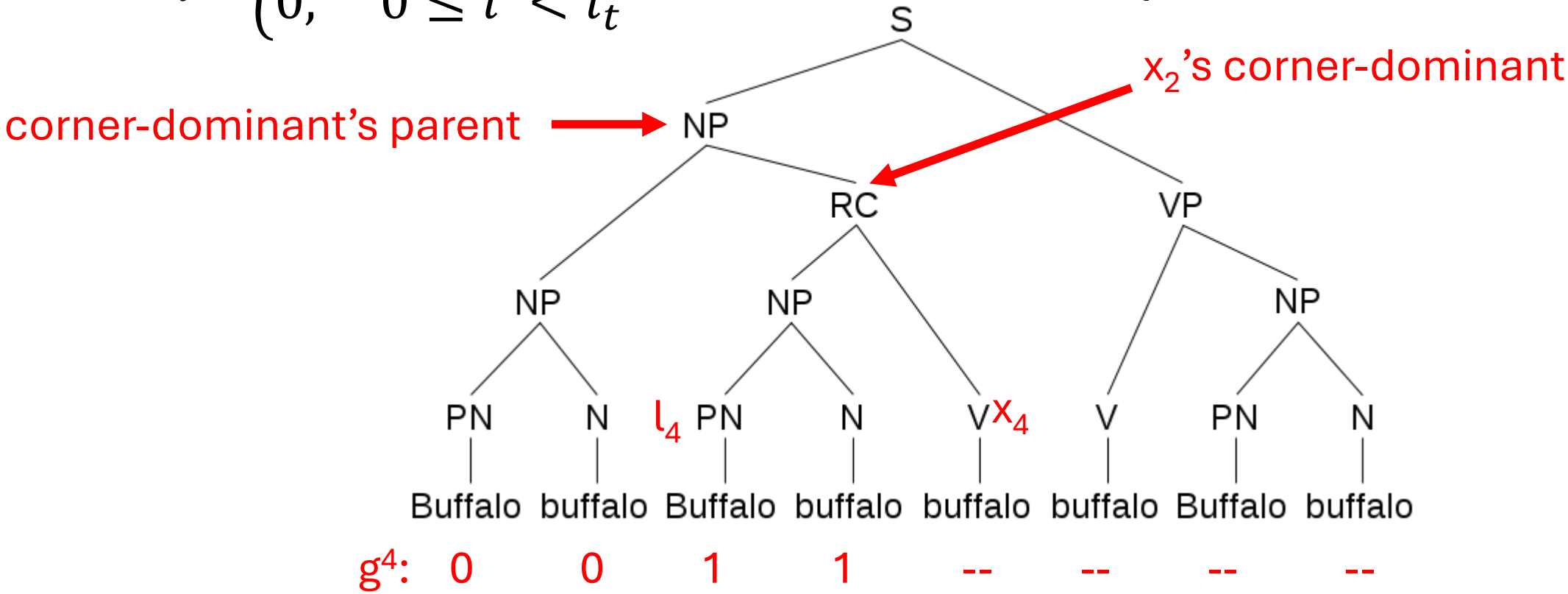
Some terminology

- "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.:



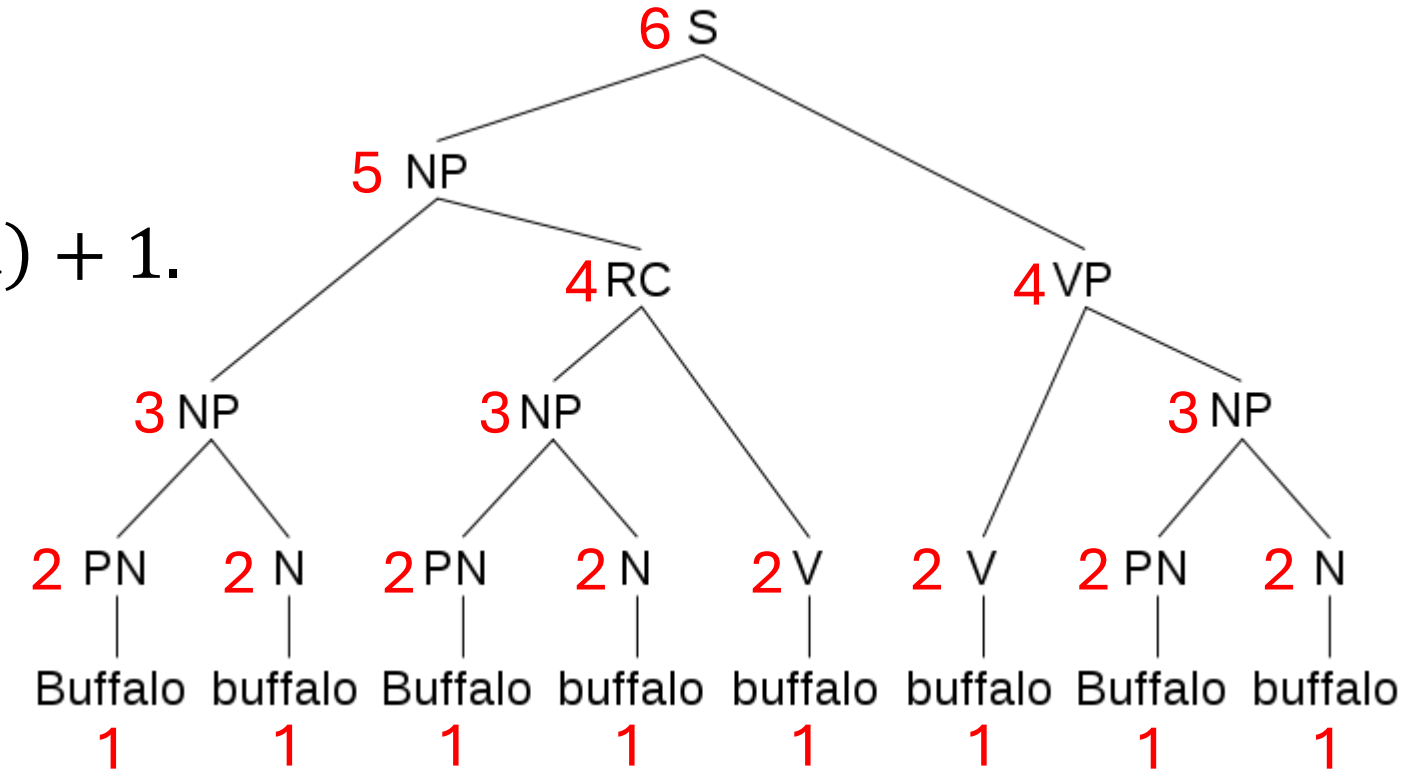
Some terminology

- "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.:



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.

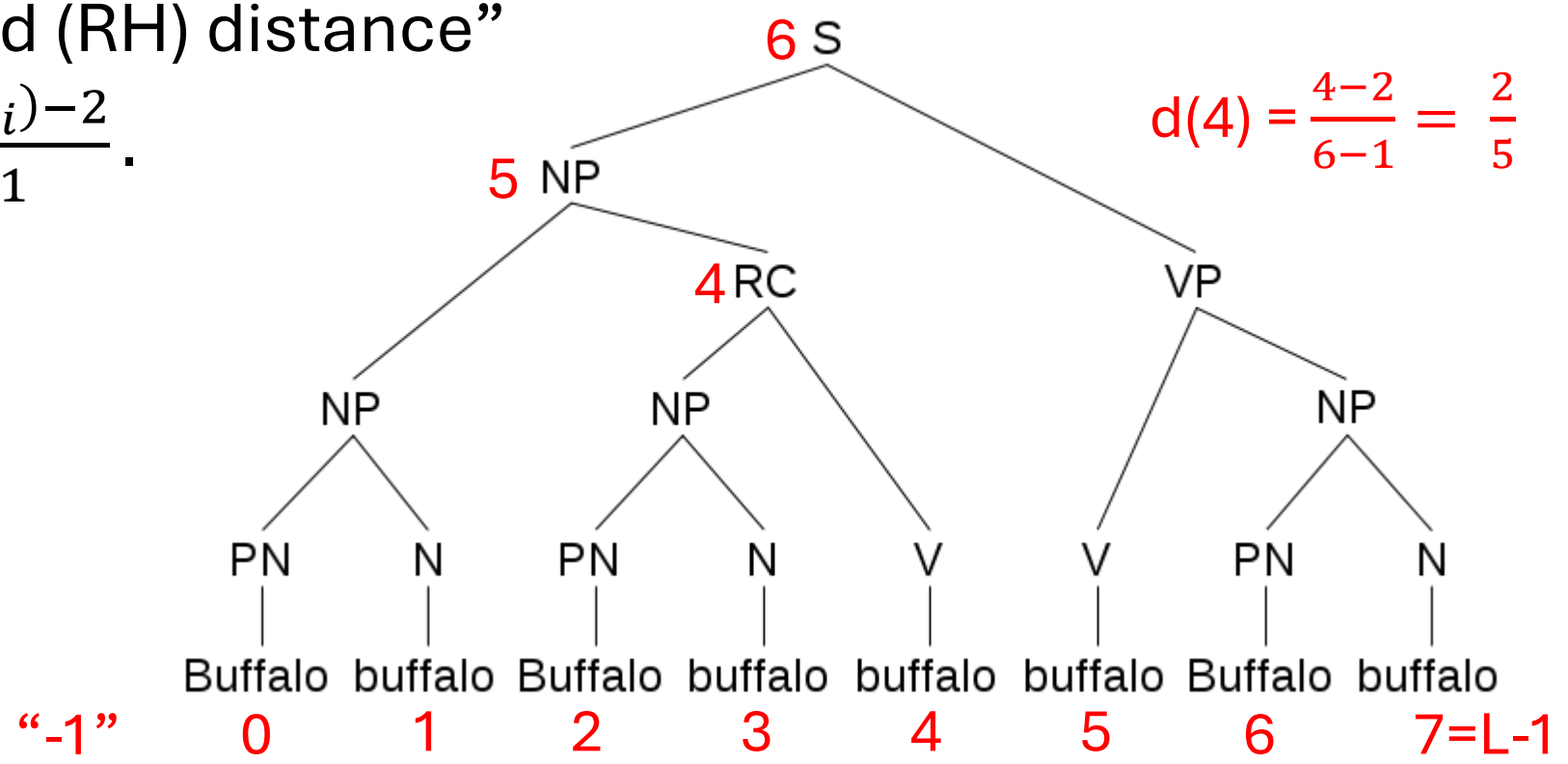
Some terminology

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

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

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

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

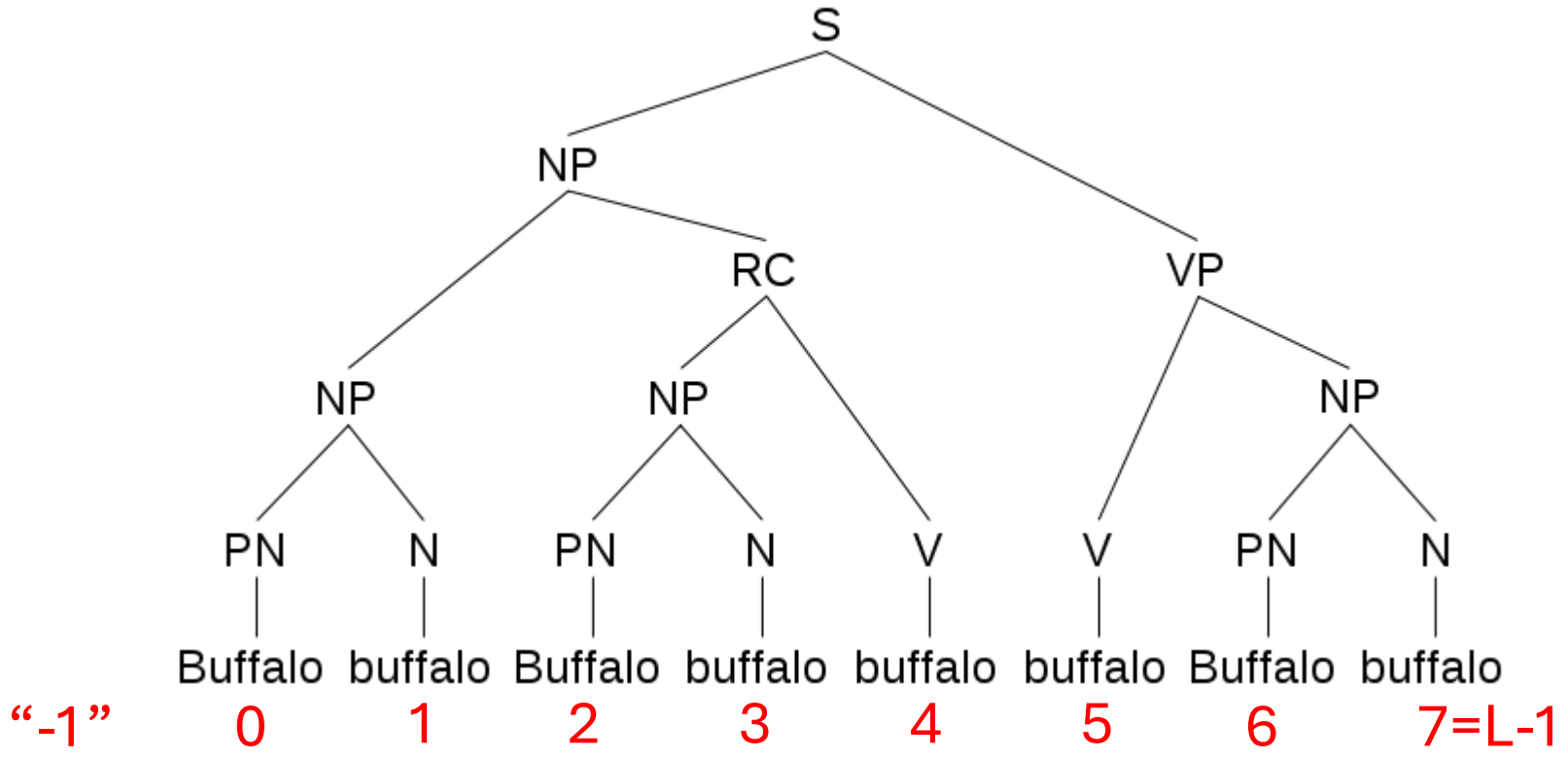


where $h(w_{-1},w_0) = h(w_{L-1},w_L) = h(r)+1$,

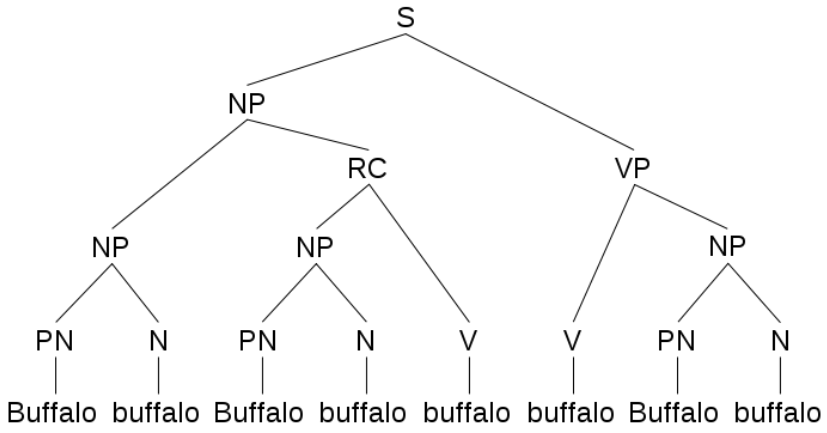
$h(u,v) = h(u \sqcup v)$ everywhere else (trees are CNF).

Quiz

- What's d(5)?

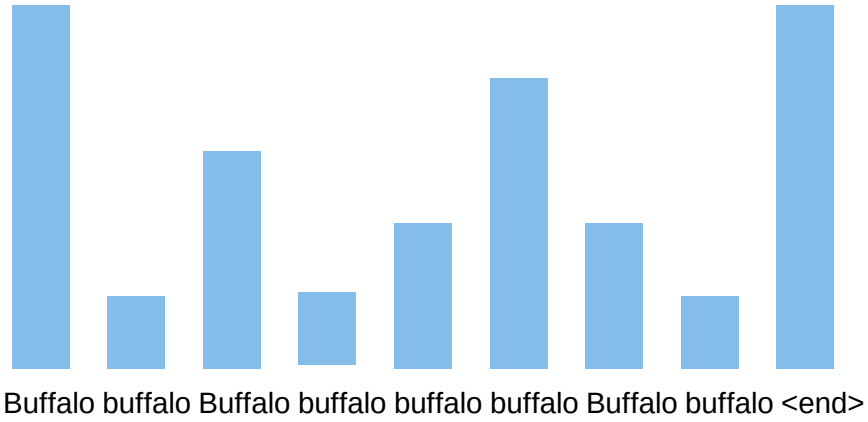


Roark-Hollingshead Conjecture



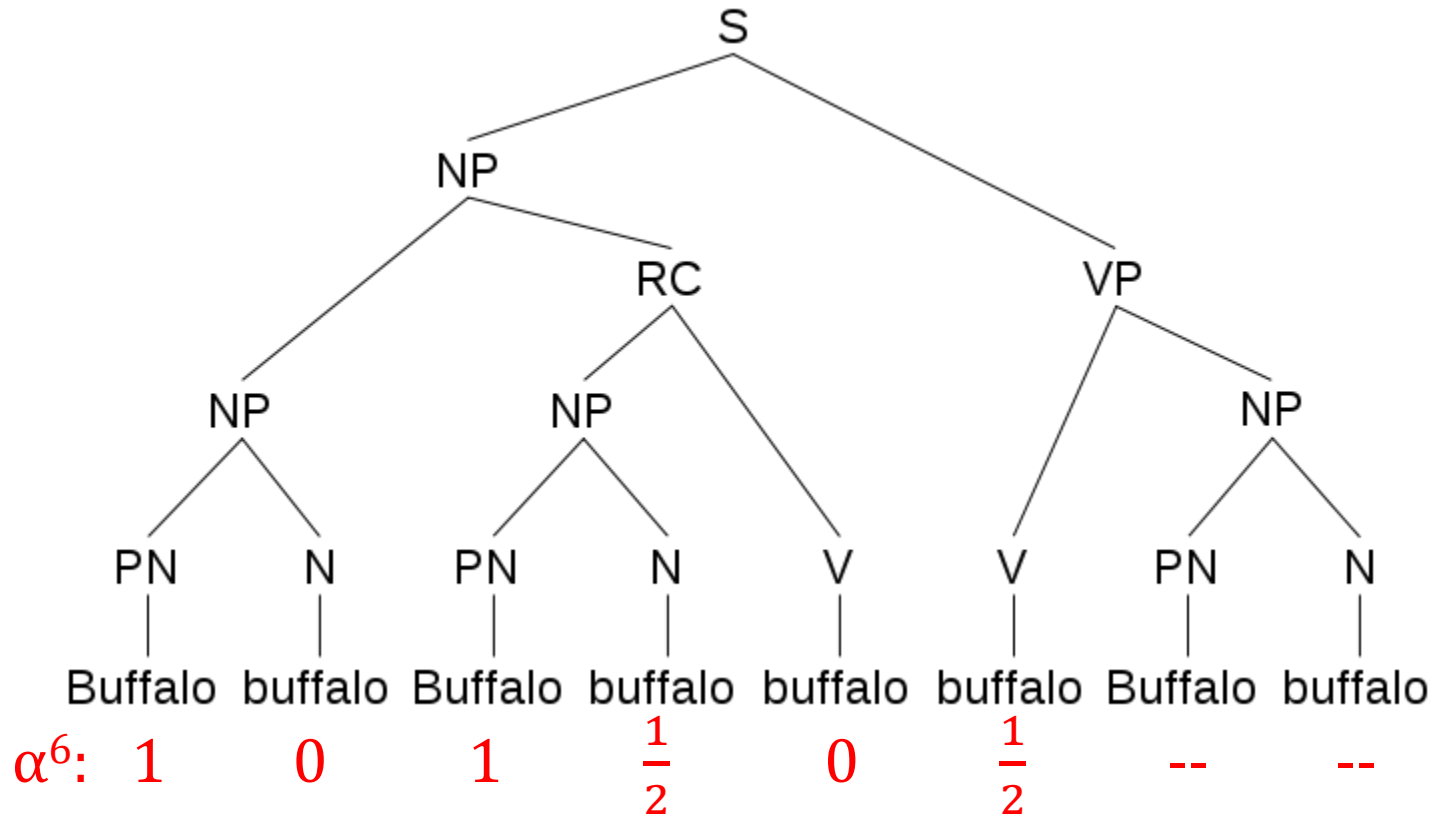
Q: How much of this does this preserve?

A: All of it (except labels)! Very cool, because this is a "local" linear statistic for tree structures.



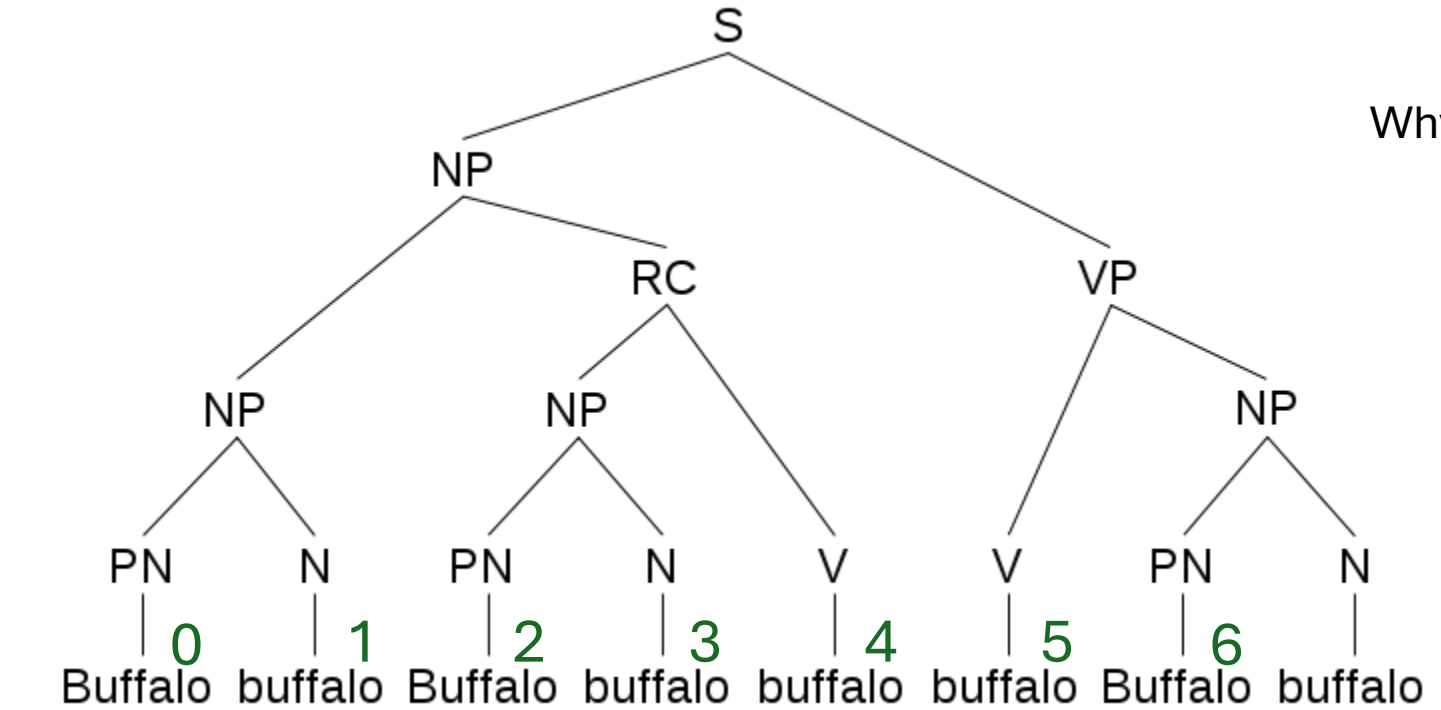
Some more terminology

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



PRPN's big idea

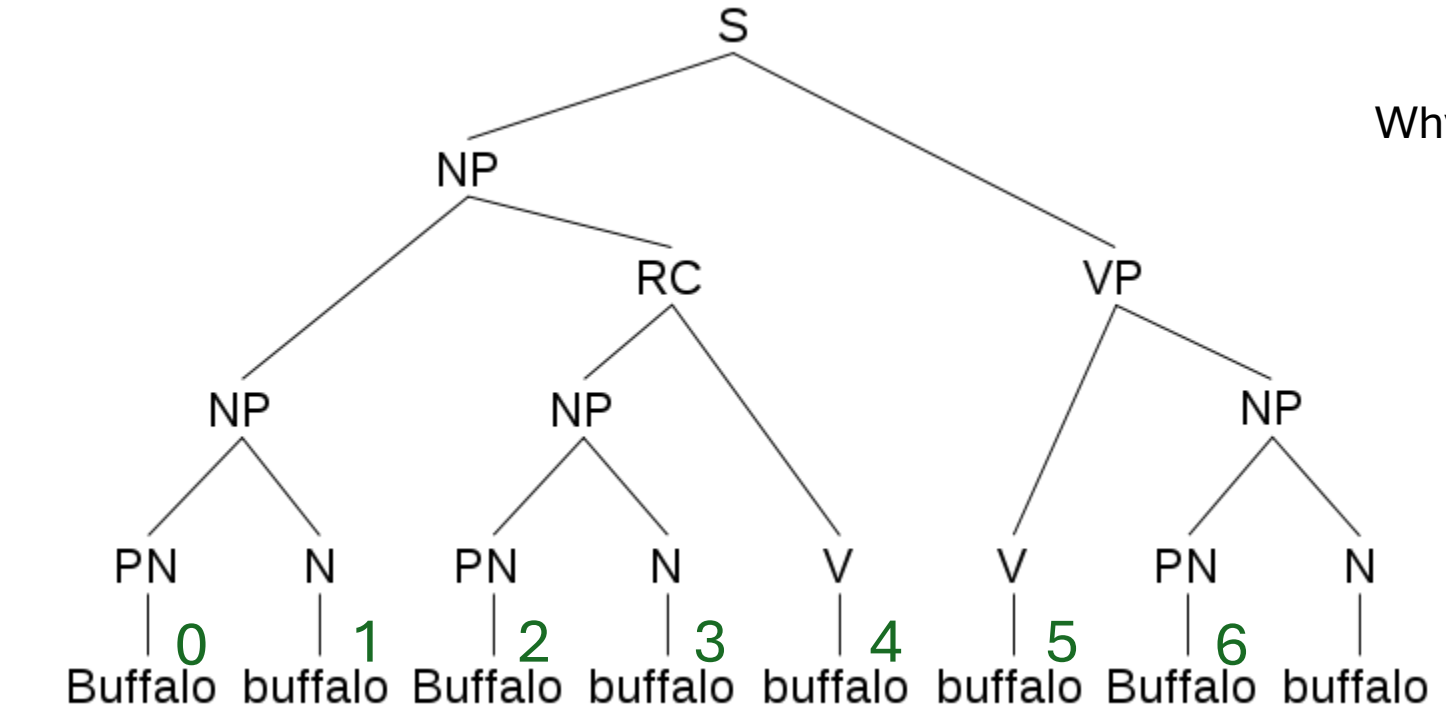
$$g_i^t = P(l_t \leq i) \approx \prod_{j=i+1}^{t-1} \alpha_j^t .$$



Why? See Shen et al. (2018)

PRPN's big idea

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



α^5: 1 1 1 1 $\frac{1}{2}$ -- -- --

g^5: 1 1 1 1 1 -- -- --

Why? See Shen et al. (2018)

PRPN: Parse

- Soften up "Dependency range:"
  - $\alpha_i^t = \frac{sign(d_t - d_{i+1}) + 1}{2}$, 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:
  - $q_t = ReLU\left(W_q \begin{bmatrix} e_{t-L} \\ e_{t-L+1} \\ \cdots \\ e_t \end{bmatrix} + b_q \right)$,
  - $d_t = ReLU(W_d q_t + b_d)$.
  - But we're not going to supervise this with d_t from actual trees…

Word vectors for w_{i-L}, w_{i-L+1}, … w_i

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

Recurrent update

PRPN: Predict

- Task: predict the probability distribution of next word x_{t+1}.
- Now, we know $m_0,\ldots,m_t$ and $e_0,\ldots,e_t$, we need to predict e_{t+1}

  - $k_t = W_m m_{t-1} + W_e e_t,$
  - $\bar{s}_i^t = softmax\left(\frac{m_i k_t^T}{\sqrt{\dim(k)}}\right),$
  - $r_i^t = \frac{g_i^{t+1}}{\sum_j g_j^{t+1}} \bar{s}_i^t,$
  - $\bar{l}_t = \sum_{i=l_{t+1}}^{t-1} r_i^t \cdot m_i$
  - Estimate $d_{t+1} \approx ReLU\left(\tilde{W}_d m_t + \tilde{b}_d\right),$
  - then estimate $\tilde{e}_{t+1} = \tanh(W_f \begin{bmatrix} \bar{l}_t \\ m_t \end{bmatrix} + b_f)$

Depends on d_{t+1}

PRPN Summary

Idea:

- Each word depends on its parent and its left siblings.
- Use g_i^t to control the LM process
- Can't directly model g_i^t because l_t is an unobserved latent variable.
- Use α_j^t to approximate g_i^t based on RH distance d_t.
- Use d_t to reconstruct the trees.
- How good is PRPN?

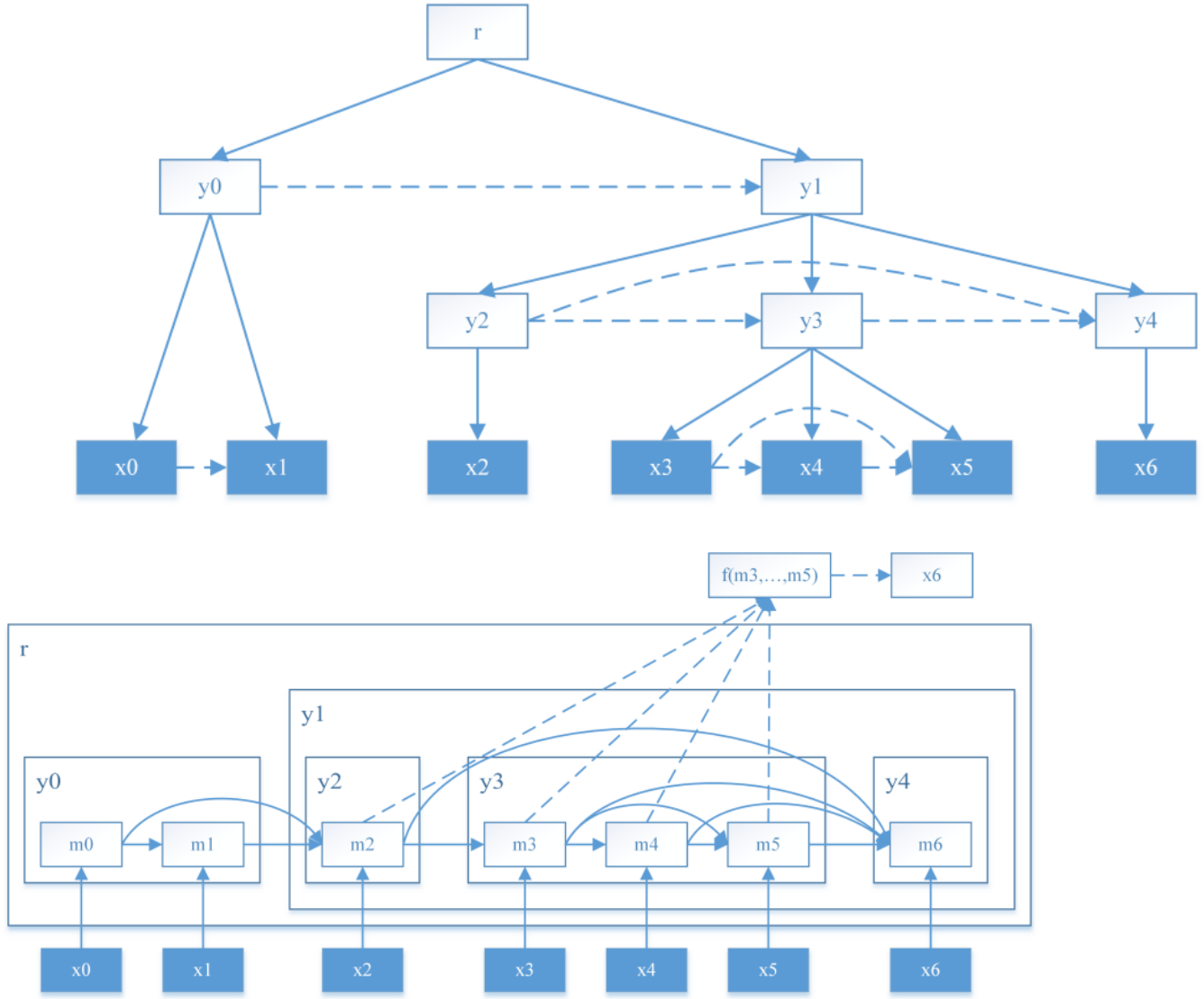


Figure 2: Proposed model architecture, hard line indicate valid connection in Reading Network, dash line indicate valid connection in Predict Network.

DOP: Data-Oriented Parsing

- DOP1 – Supervised DOP:
  - Given an annotated corpus, use all subtrees, regardless of size, to parse new sentences.

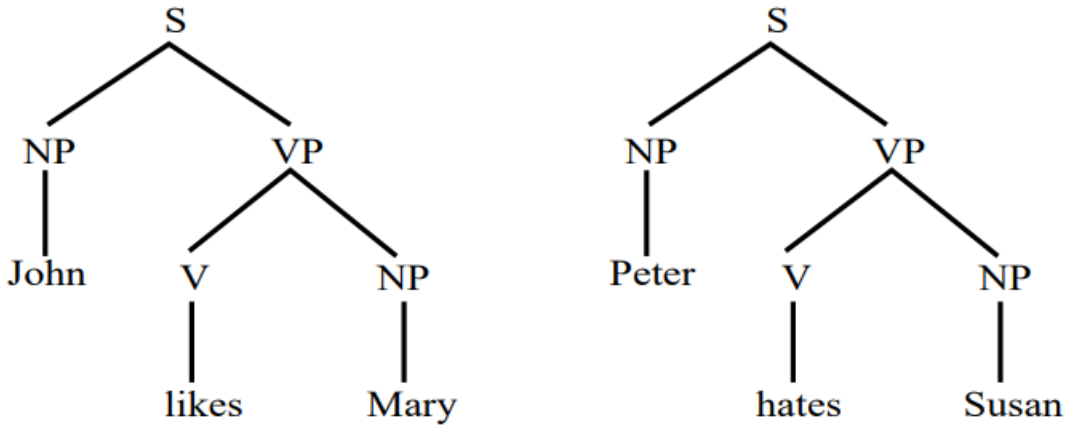


Figure 1. A corpus of two trees

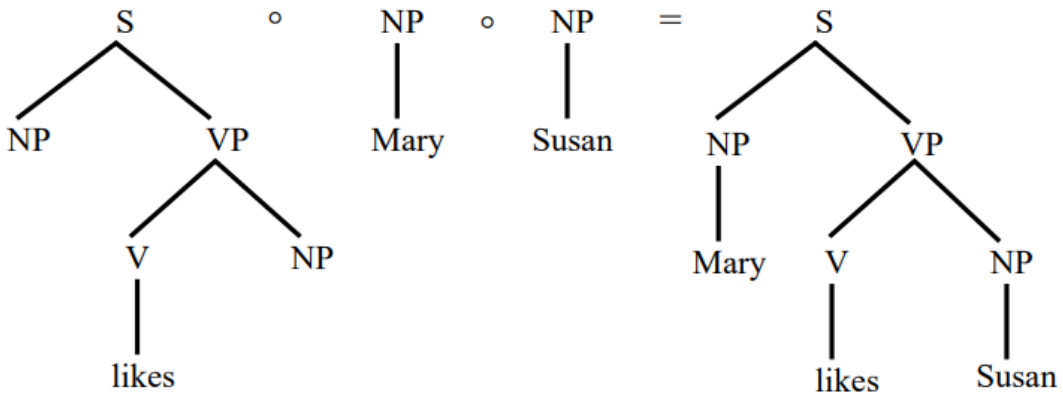
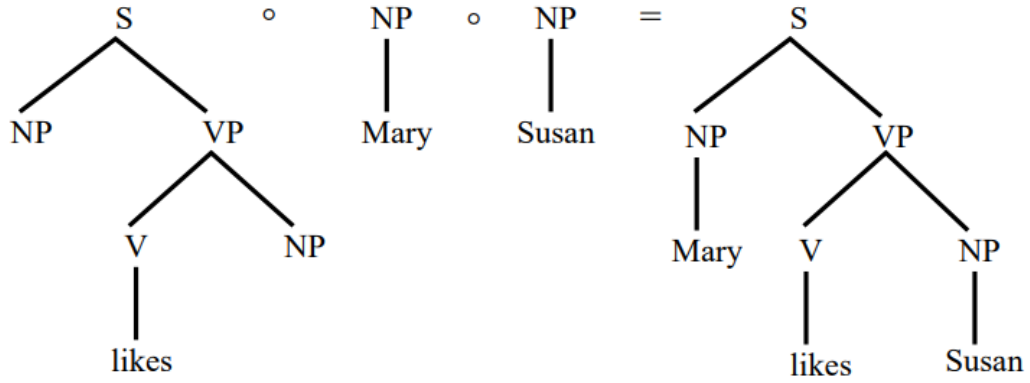


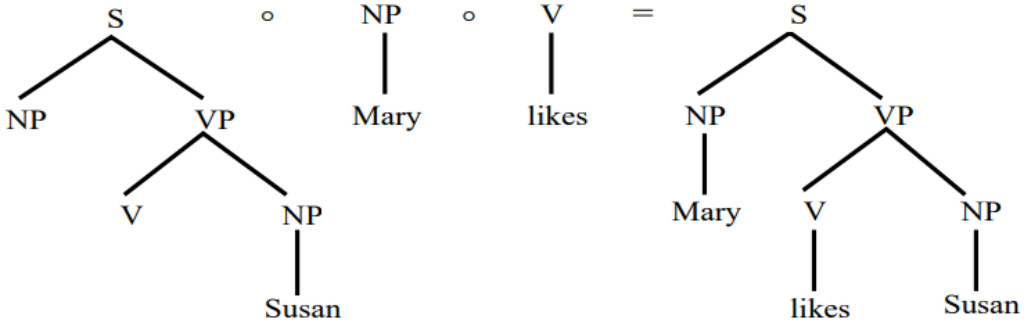
Figure 2. A derivation for *Mary likes Susan*

° means substitute

DOP: Data-Oriented Parsing

- Multiple ways to substitute.
- The probability of a subtree *t*:
  - The number of occurrences of t in the corpus,
  - Divided by the total number of occurrences of all subtrees *t′* with the same root label as *t*.
- $P(t_1 \circ \ldots \circ t_n) = \prod_i P(ti)$





DOP: Data-Oriented Parsing

- ML-DOP – maximum likelihood DOP:
  - Use an EM algorithm to estimate P(t) in DOP.
- U-DOP – Unsupervised DOP:
  - Simply use all possible subtrees as the “corpus.”
- UML-DOP
  - Use randomly sampled possible subtrees as training data.
  - Do ML-DOP.

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	