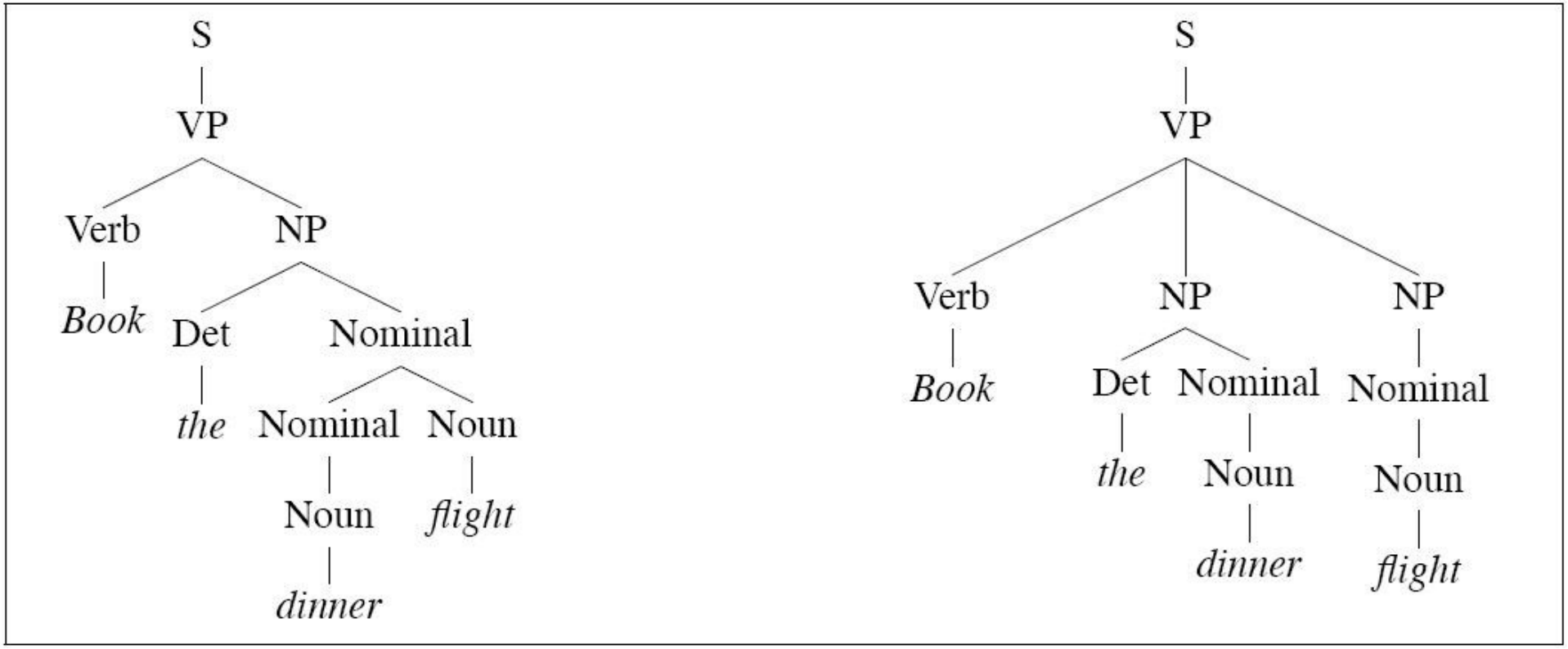
Natural Language Processing

Lecture 13:
More on CFG Parsing

Probabilistic/Weighted Parsing

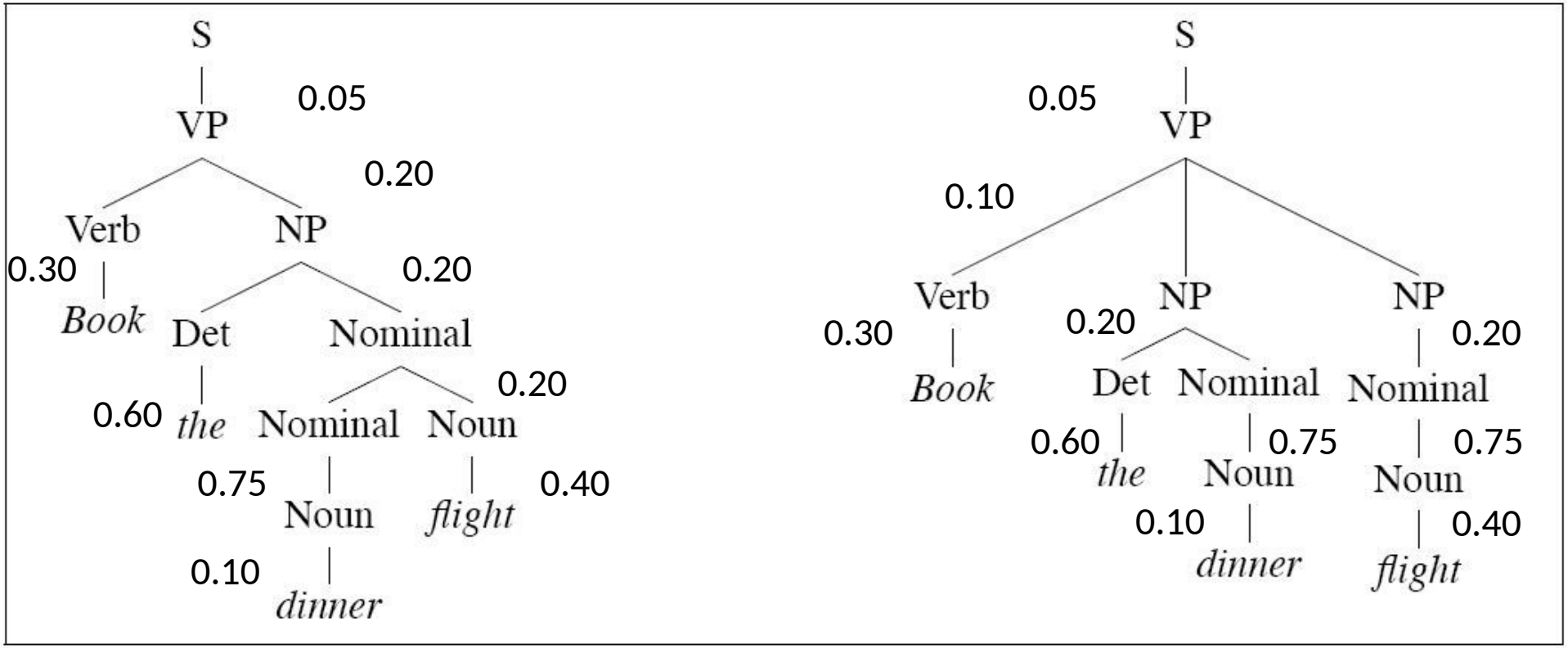
Example: ambiguous parse



Probabilistic CFG

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] | $flight$ [.30]
$S \rightarrow VP$	[.05]	| $meal$ [.15] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flights$ [.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]	

Ambiguous parse w/probabilities



P(left) = 2.2 *10^-6

P(right) = 6.1 *10^-7

Review: Context-Free Grammars

- Vocabulary of terminal symbols, Σ
- Set of nonterminal symbols (a.k.a. variables), N
- Special start symbol $S \in N$
- Production rules of the form $X \rightarrow \alpha$

where

$X \in N$

$\alpha \in (N \cup \Sigma)^*$ (in CNF: $\alpha \in N^2 \cup \Sigma$)

Probabilistic Context-Free Grammars

- Vocabulary of terminal symbols, Σ
- Set of nonterminal symbols (a.k.a. variables), N
- Special start symbol $S \in N$
- Production rules of the form $X \rightarrow \alpha$, each with a positive weight $p(X \rightarrow \alpha)$, where

  $X \in N$

  $\alpha \in (N \cup \Sigma)^*$ (in CNF: $\alpha \in N^2 \cup \Sigma$)

  $\forall X \in N, \ \sum_\alpha p(X \rightarrow \alpha) = 1$

CKY Algorithm: Review

for i = 1 … n

  C[i-1, i] = { V | V → w_i }

for ℓ = 2 … n // width

  for i = 0 … n - ℓ // left boundary

  k = i + ℓ // right boundary

  for j = i + 1 … k – 1 // midpoint

  C[i, k] = C[i, k] ∪

  { V | V → YZ, Y ∈ C[i, j], Z ∈ C[j, k] }

return true if S ∈ C[0, n]

Weighted CKY Algorithm

for $i = 1 \ldots n$, $V \in N$

  $C[V, i\text{-}1, i] = p(V \rightarrow w_i)$

for $\ell = 2 \ldots n$ // width of span

  for $i = 0 \ldots n - \ell$ // left boundary

  $k = i + \ell$ // right boundary

  for $j = i + 1 \ldots k - 1$ // midpoint

  for each binary rule $V \rightarrow YZ$

  $C[V, i, k] = \max\{ C[V, i, k],$
    $C[Y, i, j] \times C[Z, j, k]$
   $\times\ p(V \rightarrow YZ)\ \}$

return true if $S \in C[\cdot, 0, n]$

CKY Algorithm: Review

$$C[V, i-1, i] = \begin{cases} \text{TRUE} & \text{if } V \to w_i \\ \text{FALSE} & \text{otherwise} \end{cases}$$

$$C[V, i, j] = \begin{cases} \text{TRUE} & \text{if } \exists j, Y, Z \text{ such that} \\ & V \to YZ \\ & \quad \text{and } C[Y, i, k] \\ & \quad \text{and } C[Z, k, j] \\ & \quad \text{and } i < k < j \\ \text{FALSE} & \text{otherwise} \end{cases}$$

$$\text{goal} = C[S, 0, n]$$

Weighted CKY Algorithm

base case:

$$C[X, i-1, i] = p(X \rightarrow w_i)$$

induction:

$$C[X, i, k] = \max_{j,Y,Z} p(X \rightarrow Y\ Z) \times C[Y, i, j] \times C(Z, j, k)$$

goal:

$$C[S, 0, n] \text{ where } n = |\boldsymbol{w}|$$

$$p(\tau^*, w_1, w_2, \ldots, w_n) = C[S, 0, n]$$

P-CKY algorithm from book

```
function PROBABILISTIC-CKY(words,grammar) returns most probable parse
                                                  and its probability
  for j ← from 1 to LENGTH(words) do
     for all { A | A → words[j] ∈ grammar}
        table[j − 1, j, A] ← P(A → words[j])
     for i ← from j − 2 downto 0 do
        for k ← i + 1 to j − 1 do
              for all { A | A → BC ∈ grammar,
                              and table[i,k,B] > 0 and table[k, j,C] > 0 }
                    if (table[i,j,A] < P(A → BC) × table[i,k,B] × table[k,j,C]) then
                        table[i,j,A] ← P(A → BC) × table[i,k,B] × table[k,j,C]
                        back[i,j,A] ← {k,B,C}
     return BUILD_TREE(back[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Det: .40 [0,1]	NP: .30 *.40 *.02 = .0024 [0,2]	[0,3]	[0,4]	[0,5]
	N: .02 [1,2]	[1,3]	[1,4]	[1,5]
		V: .05 [2,3]	[2,4]	[3,5]
			[3,4]	[3,5]
				[4,5]

Parsing as (Weighted) Deduction

Antecedents

Side conditions

$$\frac{A : u \quad B : v}{C : w} \quad \phi$$

Consequent

"If A and B are true with weights u and v, and phi is also true, then C is true with weight w."

Earley’s Algorithm

Jay Earley, PhD

My Self-Therapy Journey Story

Self-Therapy Journey (STJ) is my brain-child and my life purpose. It brings together the two professional sides of me that have been separate up till now—the computer scientist and the psychologist. It is very exciting to integrate these two disparate sides of me in this amazing project.

Example Grammar (same for CKY)

Grammar	Lexicon
$S \rightarrow NP\ VP$	$Det \rightarrow that \mid this \mid a$
$S \rightarrow Aux\ NP\ VP$	$Noun \rightarrow book \mid flight \mid meal \mid money$
$S \rightarrow VP$	$Verb \rightarrow book \mid include \mid prefer$
$NP \rightarrow Pronoun$	$Pronoun \rightarrow I \mid she \mid me$
$NP \rightarrow Proper\text{-}Noun$	$Proper\text{-}Noun \rightarrow Houston \mid NWA$
$NP \rightarrow Det\ Nominal$	$Aux \rightarrow does$
$Nominal \rightarrow Noun$	$Preposition \rightarrow from \mid to \mid on \mid near \mid through$
$Nominal \rightarrow Nominal\ Noun$	
$Nominal \rightarrow Nominal\ PP$	
$VP \rightarrow Verb$	
$VP \rightarrow Verb\ NP$	
$VP \rightarrow Verb\ NP\ PP$	
$VP \rightarrow Verb\ PP$	
$VP \rightarrow VP\ PP$	
$PP \rightarrow Preposition\ NP$	

Earley Parsing

- Allows arbitrary CFGs
- Top-down control
- Fills a table (or *chart*) in a single sweep over the input
  - Table is length N+1; N is number of words
  - Table entries represent
    - Completed constituents and their locations
    - In-progress constituents
    - Predicted constituents

States

- The table-entries are called states and are represented with dotted-rules.

S . VP	A VP is predicted
NP Det . Nominal	An NP is in progress
VP V NP .	A VP has been found

States/Locations

- S . VP [0,0]

A VP is predicted at the start of the sentence

- NP Det .Nominal [1,2]

An NP is in progress; the Det goes from 1 to 2

- VP V NP . [0,3]

A VP has been found starting at 0 and ending at 3

Earley top-level

- As with most dynamic programming approaches, the answer is found by looking in the table in the right place.
- In this case, there should be an S state in the final column that spans from 0 to N and is complete. That is,

S α . [0,N]

- If that's the case, you're done.

Earley top-level (2)

- So sweep through the table from 0 to N…
  - New predicted states are created by starting top-down from S
  - New incomplete states are created by advancing existing states as new constituents are discovered
  - New complete states are created in the same way.

Earley top-level (3)

- More specifically…
  1. *Predict* all the states you can upfront
  2. Read a word
     1. Extend states based on matches
     2. Generate new predictions
     3. Go to step 2
  3. When you're out of words, look at the chart to see if you have a winner

Earley code: top-level

```
function EARLEY-PARSE(words, grammar) returns chart

  ENQUEUE((γ → • S, [0,0]), chart[0])
  for i ← from 0 to LENGTH(words) do
   for each state in chart[i] do
     if INCOMPLETE?(state) and
              NEXT-CAT(state) is not a part of speech then
        PREDICTOR(state)
     elseif INCOMPLETE?(state) and
              NEXT-CAT(state) is a part of speech then
        SCANNER(state)
     else
        COMPLETER(state)
   end
  end
  return(chart)
```

Earley code: 3 main functions

```
procedure PREDICTOR((A → α • B β, [i, j]))
    for each (B → γ) in GRAMMAR-RULES-FOR(B, grammar) do
        ENQUEUE((B → • γ, [j, j]), chart[j])
    end

procedure SCANNER((A → α • B β, [i, j]))
    if B ⊂ PARTS-OF-SPEECH(word[j]) then
        ENQUEUE((B → word[j], [j, j+1]), chart[j+1])

procedure COMPLETER((B → γ •, [j,k]))
    for each (A → α • B β, [i, j]) in chart[j] do
        ENQUEUE((A → α B • β, [i,k]), chart[k])
    end

procedure ENQUEUE(state, chart-entry)
    if state is not already in chart-entry then
        PUSH(state, chart-entry)
    end
```

Extended Earley Example

- *Book that flight*
- We should find:

an S from 0 to 3 that is a completed state

Chart[0]	S0	$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
	S1	$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
	S2	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
	S3	$S \rightarrow \bullet VP$	[0,0]	Predictor
	S4	$NP \rightarrow \bullet Pronoun$	[0,0]	Predictor
	S5	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
	S6	$NP \rightarrow \bullet Det\ Nominal$	[0,0]	Predictor
	S7	$VP \rightarrow \bullet Verb$	[0,0]	Predictor
	S8	$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor
	S9	$VP \rightarrow \bullet Verb\ NP\ PP$	[0,0]	Predictor
	S10	$VP \rightarrow \bullet Verb\ PP$	[0,0]	Predictor
	S11	$VP \rightarrow \bullet VP\ PP$	[0,0]	Predictor
Chart[1]	S12	$Verb \rightarrow book \bullet$	[0,1]	Scanner
	S13	$VP \rightarrow Verb \bullet$	[0,1]	Completer
	S14	$VP \rightarrow Verb \bullet NP$	[0,1]	Completer
	S15	$VP \rightarrow Verb \bullet NP\ PP$	[0,1]	Completer
	S16	$VP \rightarrow Verb \bullet PP$	[0,1]	Completer
	S17	$S \rightarrow VP \bullet$	[0,1]	Completer
	S18	$VP \rightarrow VP \bullet PP$	[0,1]	Completer
	S19	$NP \rightarrow \bullet Pronoun$	[1,1]	Predictor
	S20	$NP \rightarrow \bullet Proper\text{-}Noun$	[1,1]	Predictor
	S21	$NP \rightarrow \bullet Det\ Nominal$	[1,1]	Predictor
	S22	$PP \rightarrow \bullet Prep\ NP$	[1,1]	Predictor

Chart[0]	S0	$\gamma \rightarrow \bullet S$	[0,0]	Dummy start state
	S1	$S \rightarrow \bullet NP\ VP$	[0,0]	Predictor
	S2	$S \rightarrow \bullet Aux\ NP\ VP$	[0,0]	Predictor
	S3	$S \rightarrow \bullet VP$	[0,0]	Predictor
	S4	$NP \rightarrow \bullet Pronoun$	[0,0]	Predictor
	S5	$NP \rightarrow \bullet Proper\text{-}Noun$	[0,0]	Predictor
	S6	$NP \rightarrow \bullet Det\ Nominal$	[0,0]	Predictor
	S7	$VP \rightarrow \bullet Verb$	[0,0]	Predictor
	S8	$VP \rightarrow \bullet Verb\ NP$	[0,0]	Predictor
	S9	$VP \rightarrow \bullet Verb\ NP\ PP$	[0,0]	Predictor
	S10	$VP \rightarrow \bullet Verb\ PP$	[0,0]	Predictor
	S11	$VP \rightarrow \bullet VP\ PP$	[0,0]	Predictor
Chart[1]	S12	$Verb \rightarrow book \bullet$	[0,1]	Scanner
	S13	$VP \rightarrow Verb \bullet$	[0,1]	Completer
	S14	$VP \rightarrow Verb \bullet NP$	[0,1]	Completer
	S15	$VP \rightarrow Verb \bullet NP\ PP$	[0,1]	Completer
	S16	$VP \rightarrow Verb \bullet PP$	[0,1]	Completer
	S17	$S \rightarrow VP \bullet$	[0,1]	Completer
	S18	$VP \rightarrow VP \bullet PP$	[0,1]	Completer
	S19	$NP \rightarrow \bullet Pronoun$	[1,1]	Predictor
	S20	$NP \rightarrow \bullet Proper\text{-}Noun$	[1,1]	Predictor
	S21	$NP \rightarrow \bullet Det\ Nominal$	[1,1]	Predictor
	S22	$PP \rightarrow \bullet Prep\ NP$	[1,1]	Predictor

Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
	S24	*NP* → *Det* • *Nominal*	[1,2]	Completer
	S25	*Nominal* → • *Noun*	[2,2]	Predictor
	S26	*Nominal* → • *Nominal Noun*	[2,2]	Predictor
	S27	*Nominal* → • *Nominal PP*	[2,2]	Predictor
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	Completer
	S30	*NP* → *Det Nominal* •	[1,3]	Completer
	S31	*Nominal* → *Nominal* • *Noun*	[2,3]	Completer
	S32	*Nominal* → *Nominal* • *PP*	[2,3]	Completer
	S33	*VP* → *Verb NP* •	[0,3]	Completer
	S34	*VP* → *Verb NP* • *PP*	[0,3]	Completer
	S35	*PP* → • *Prep NP*	[3,3]	Predictor
	S36	*S* → *VP* •	[0,3]	Completer
	S37	*VP* → *VP* • *PP*	[0,3]	Completer

Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
	S24	*NP* → *Det* • *Nominal*	[1,2]	Completer
	S25	*Nominal* → • *Noun*	[2,2]	Predictor
	S26	*Nominal* → • *Nominal Noun*	[2,2]	Predictor
	S27	*Nominal* → • *Nominal PP*	[2,2]	Predictor
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	Completer
	S30	*NP* → *Det Nominal* •	[1,3]	Completer
	S31	*Nominal* → *Nominal* • *Noun*	[2,3]	Completer
	S32	*Nominal* → *Nominal* • *PP*	[2,3]	Completer
	S33	*VP* → *Verb NP* •	[0,3]	Completer
	S34	*VP* → *Verb NP* • *PP*	[0,3]	Completer
	S35	*PP* → • *Prep NP*	[3,3]	Predictor
	S36	*S* → *VP* •	[0,3]	Completer
	S37	*VP* → *VP* • *PP*	[0,3]	Completer

Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
	S24	*NP* → *Det* • *Nominal*	[1,2]	Completer
	S25	*Nominal* → • *Noun*	[2,2]	Predictor
	S26	*Nominal* → • *Nominal Noun*	[2,2]	Predictor
	S27	*Nominal* → • *Nominal PP*	[2,2]	Predictor
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	Completer
	S30	*NP* → *Det Nominal* •	[1,3]	Completer
	S31	*Nominal* → *Nominal* • *Noun*	[2,3]	Completer
	S32	*Nominal* → *Nominal* • *PP*	[2,3]	Completer
	S33	*VP* → *Verb NP* •	[0,3]	Completer
	S34	*VP* → *Verb NP* • *PP*	[0,3]	Completer
	S35	*PP* → • *Prep NP*	[3,3]	Predictor
	S36	*S* → *VP* •	[0,3]	Completer
	S37	*VP* → *VP* • *PP*	[0,3]	Completer

Earley's Algorithm in equations

- We can look at this from the declarative programming point of view too.

ROOT → • S [0,0]					goal: ROOT → S• [0,n]
book	the	flight	through	Chicago	

Earley's Algorithm: PREDICT

Given $V \rightarrow \alpha \bullet X\beta$ [i, j]
and the rule $X \rightarrow \gamma$,
create $X \rightarrow \bullet\gamma$ [j, j]

ROOT → • S [0,0]
S→ VP
S → • VP [0,0]

ROOT → • S [0,0] S → • VP [0,0] S → • NP VP [0,0] … VP → • V NP [0,0] … NP → • DT N [0,0] …					
book	the	flight	through	Chicago	

Earley's Algorithm: SCAN

Given $V \rightarrow \alpha \bullet T\beta$ [i, j]
and the rule $T \rightarrow w_{j+1}$,
create $T \rightarrow w_{j+1} \bullet$ [j, j+1]

VP → • V NP [0,0]
V → book
V → book • [0,1]

ROOT → • S [0,0] S → • VP [0,0] S → • NP VP [0,0] … VP → • V NP [0,0] … NP → • DT N [0,0] …	V → book• [0, 1]				
book	the	flight	through	Chicago	

Earley's Algorithm: COMPLETE

Given $V \rightarrow \alpha \bullet X \beta$ [i, j]
and $X \rightarrow \gamma \bullet$ [j, k],
create $V \rightarrow \alpha X \bullet \beta$ [i, k]

VP → • V NP [0,0]
V → book • [0,1]
VP → V • NP [0,1]

ROOT → • S [0,0] S → • VP [0,0] S → • NP VP [0,0] … VP → • V NP [0,0] … NP → • DT N [0,0] …	V → book• [0, 1] VP → V • NP [0,1]				
book	the	flight	through	Chicago	

Earley's Algorithm

Goal: $[\text{ROOT} \rightarrow S\bullet, 0, n]$ **Items:** $[X \rightarrow \alpha \bullet \beta, i, j]$

Axiom: $\overline{[\text{ROOT} \rightarrow \bullet S, 0, 0]}$

Inference rules:

$$\text{PREDICT} \quad \frac{[V \rightarrow \alpha \bullet X\beta, i, j]}{[X \rightarrow \bullet\gamma, j, j] : p} \quad X \xrightarrow{p} \gamma \in R$$

$$\text{SCAN} \quad \frac{[V \rightarrow \alpha \bullet w_{j+1}\beta, i, j] : p}{[V \rightarrow \alpha w_{j+1} \bullet \beta, i, j] : p]}$$

$$\text{COMPLETE} \quad \frac{[X \rightarrow \gamma\bullet, j, k] : p \quad [V \rightarrow \alpha \bullet X\beta, i, j] : q}{[V \rightarrow \alpha X \bullet \beta, i, k] : pq}$$

Thought Questions

- Runtime?
  - $O(n^3)$
- Memory?
  - $O(n^2)$
- Can we make it faster?
- Recovering trees?

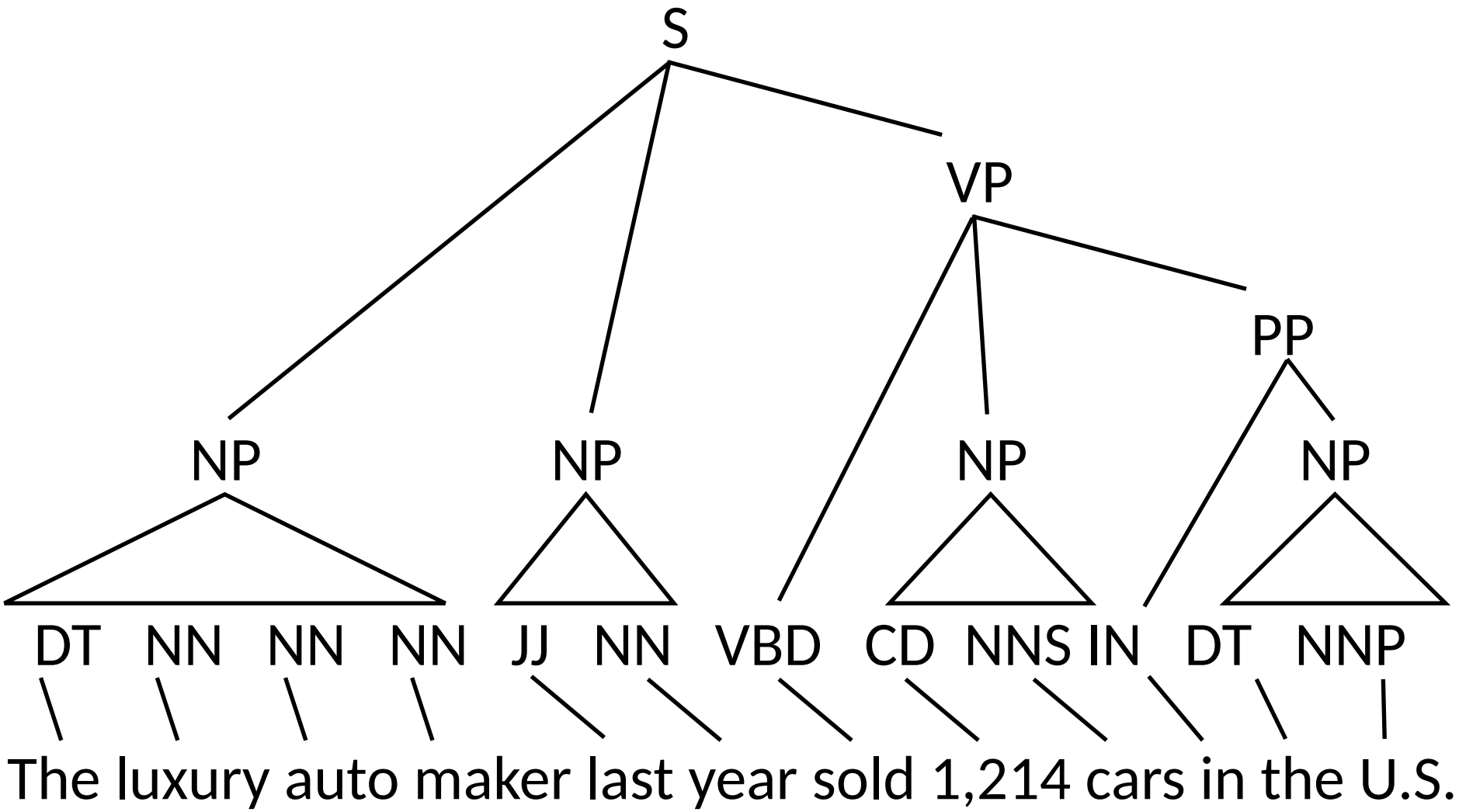
Make it an Earley Parser

Chart[1]	S12	*Verb* → *book* •	[0,1]	Scanner
Chart[2]	S23	*Det* → *that* •	[1,2]	Scanner
Chart[3]	S28	*Noun* → *flight* •	[2,3]	Scanner
	S29	*Nominal* → *Noun* •	[2,3]	(S28)
	S30	*NP* → *Det Nominal* •	[1,3]	(S23, S29)
	S33	*VP* → *Verb NP* •	[0,3]	(S12, S30)
	S36	*S* → *VP* •	[0,3]	(S33)

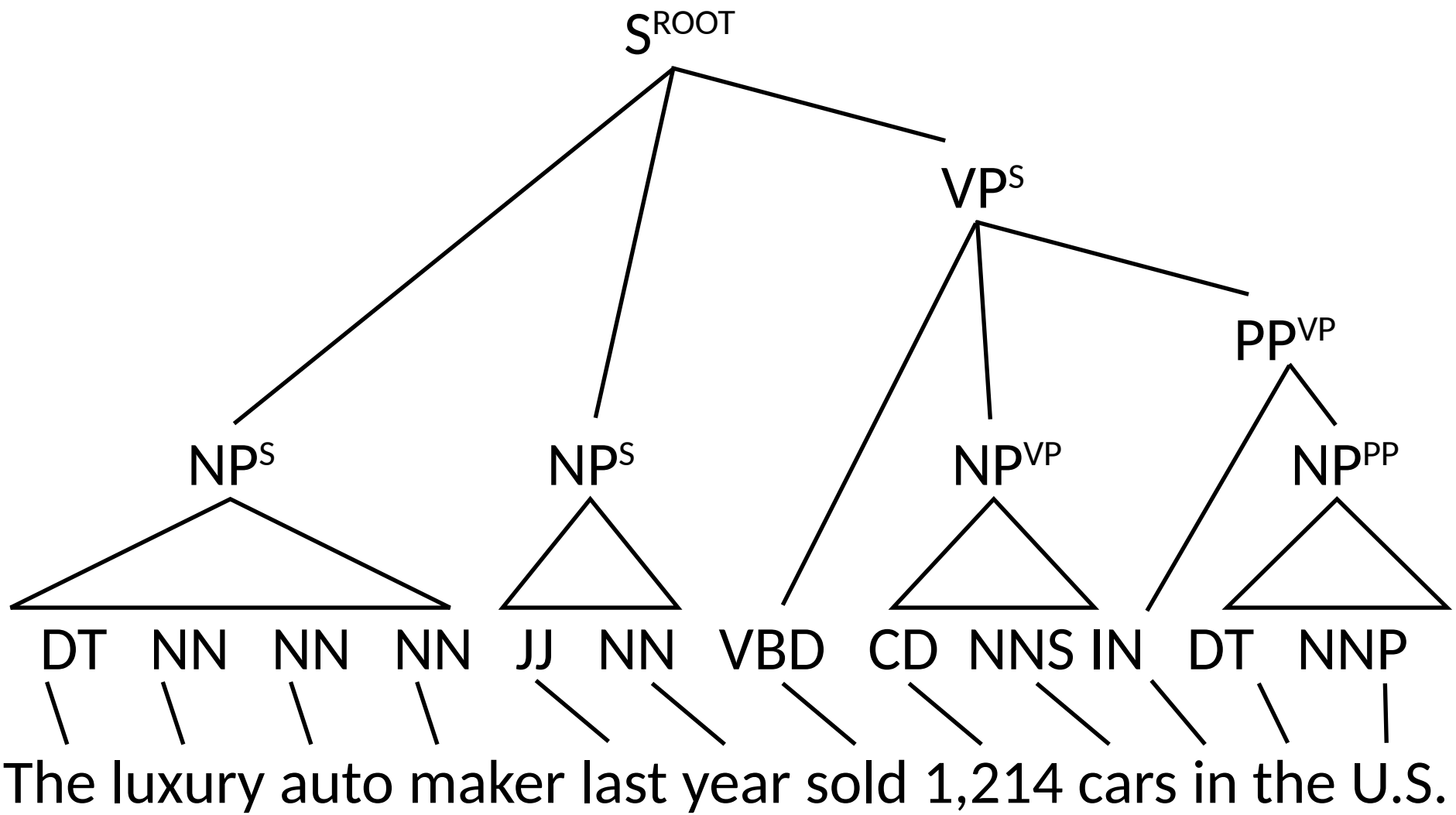
- Record which sub rules we used to complete edges

Heads in CFGs

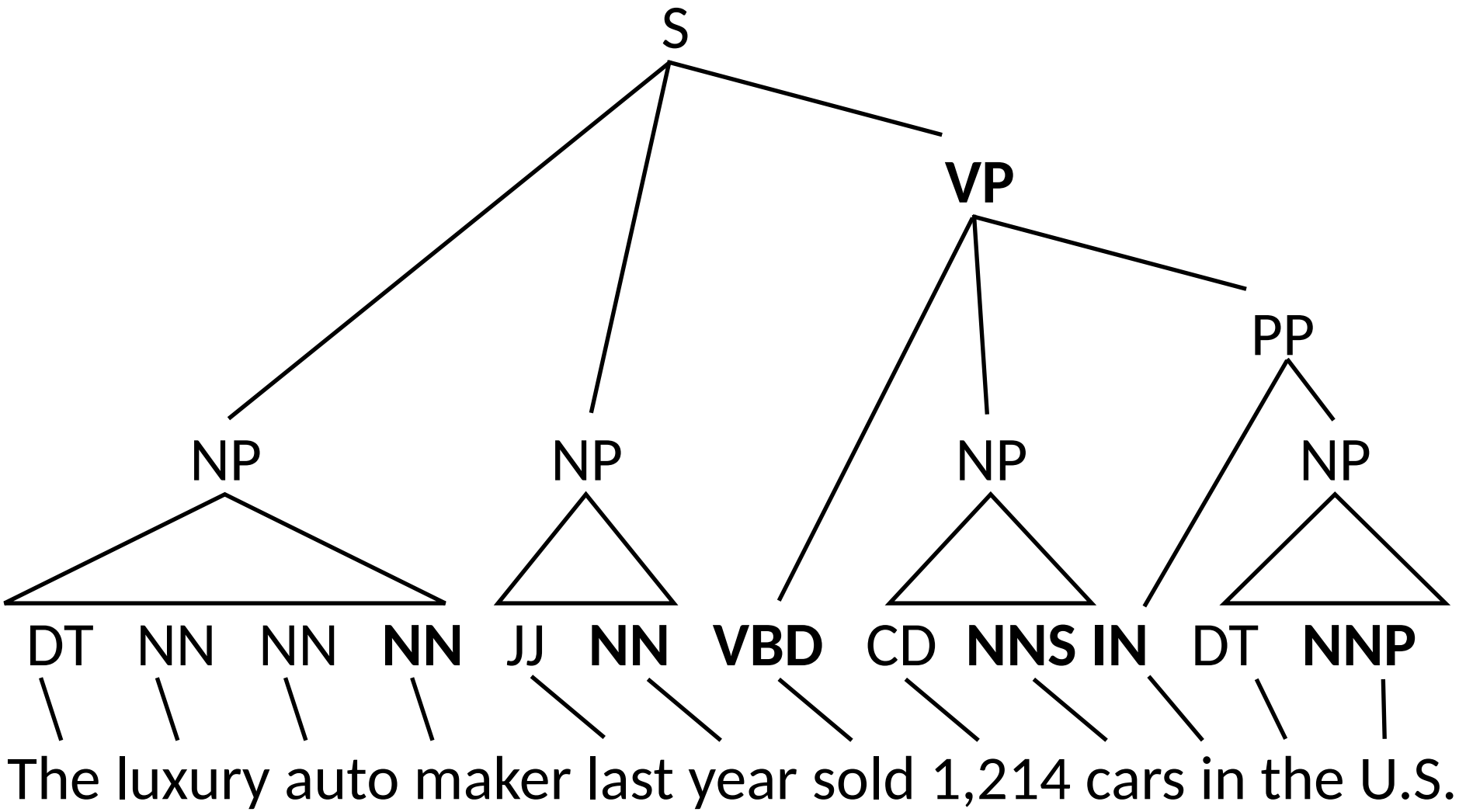
Treebank Tree



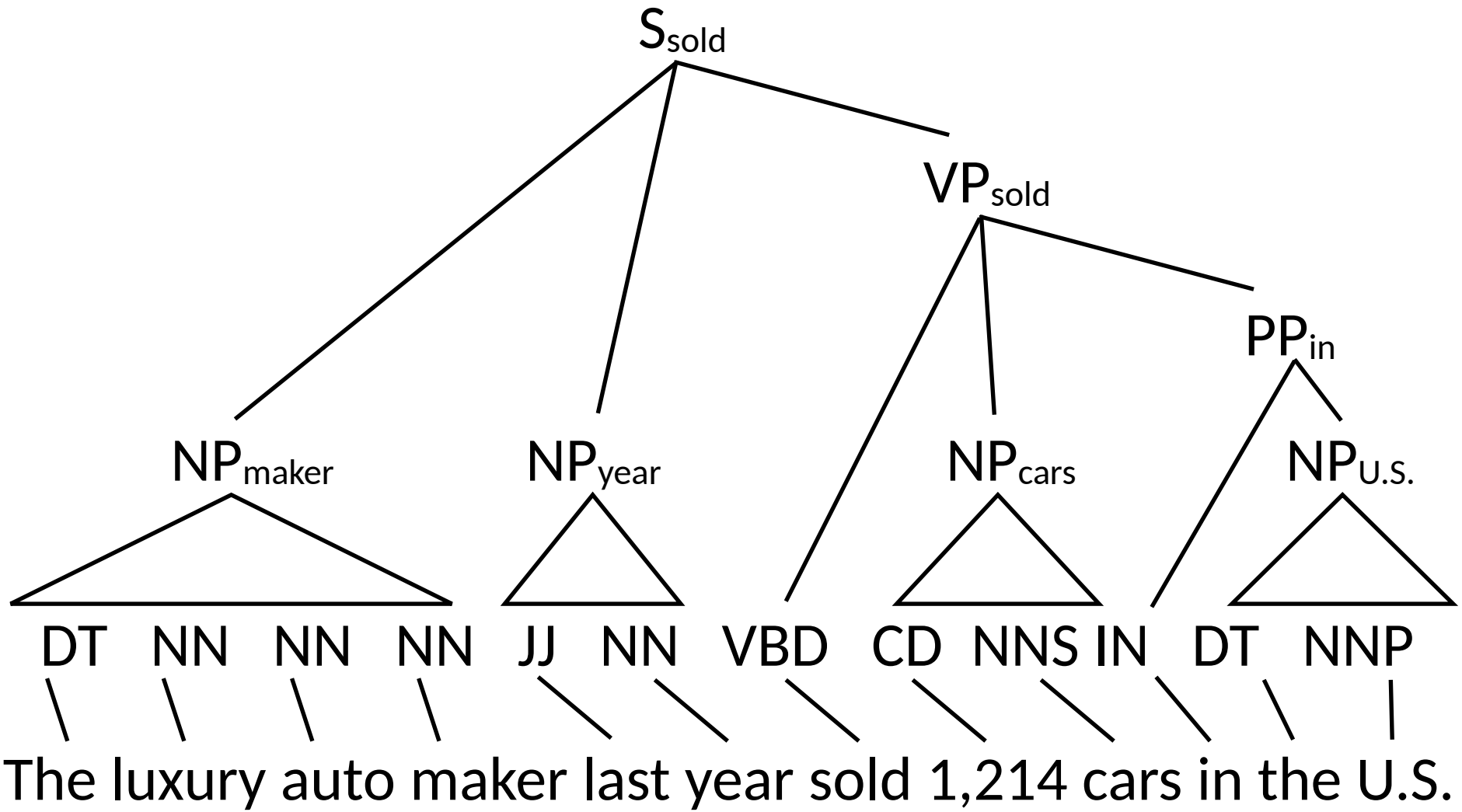
Parent-Annotated Tree



Headed Tree


S
VP
PP
NP
NP
NP
NP
DT NN NN NN JJ NN VBD CD NNS IN DT NNP
The luxury auto maker last year sold 1,214 cars in the U.S.

Lexicalized Tree



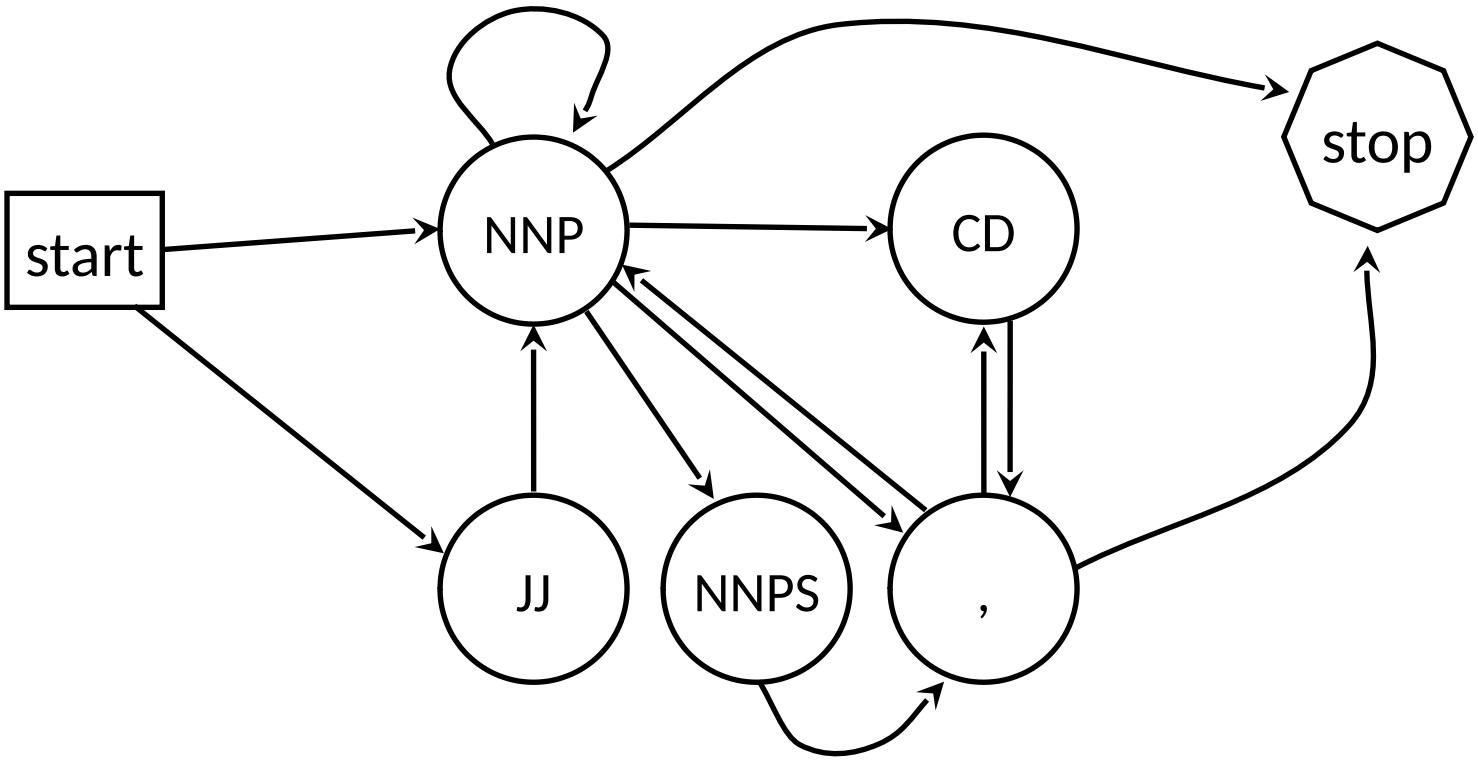
Random PCFG Text (5 ancestors, lex.)

- it can remember one million truly inspiring teachers from Rainbow Technologies .
- I have been able *-1 to force *-2 to be more receptive to therapy , and to keep the committee informed *-2 , usually in advance , of covert actions : ; the victims are large and costly machines .
- As their varied strategies suggest , Another suggestion would predict they will pay off .
- the two-day trip reportedly has said it would be done *-1 .
- Others have soared to the car market well .
- A spokesman for * paying the bill declined *-1 to pay taxes , but the fact that *T*-84 adjusted payouts on behalf of preventative medicine in terms of 29 years could be distributed *-1 .
- P&G , in the space of Orrick , Herrington & Sutcliffe , rarely rolls forward on a modest 1.1 million shares on the block .
- In the eight months last Friday , bond prices closed yesterday at $ 30.2 million , down 25 cents .
- Still , Honda says *T*-1 is calling for slight declines when there was posted *-1 within its pre-1967 borders .
- Moreover , Allianz 's Mr. Jarrett also sees only a `` internal erosion '' of about 35 of St. Petersburg , Fla. due 1994 .
- it *EXP*-1 is predicting negative third : - and fourth-quarter growth .
- Grace said luxury-car sales increased 1.4 % to 221.61 billion yen -LRB- $ 188.2 -RRB- , from $ 234.4 million a share , or $ 9.6 million , a year earlier .
- But AGIP already has been group vice president for such a gizmo at Texas Air .
- And when other rules are safeguarded *-232 by the Appropriations Committee *T*-1 , the White House passed a $ 1.5765 billion loan market-revision bill providing the first construction funds for the economy 's ambitious radio station in fiscal 1990 and incorporating far-reaching provisions affecting the erratic copper market .
- The urging also has yet opened in September in September .
- But Mr. Lorenzo is *-1 to elaborate on the latest reports of the line .

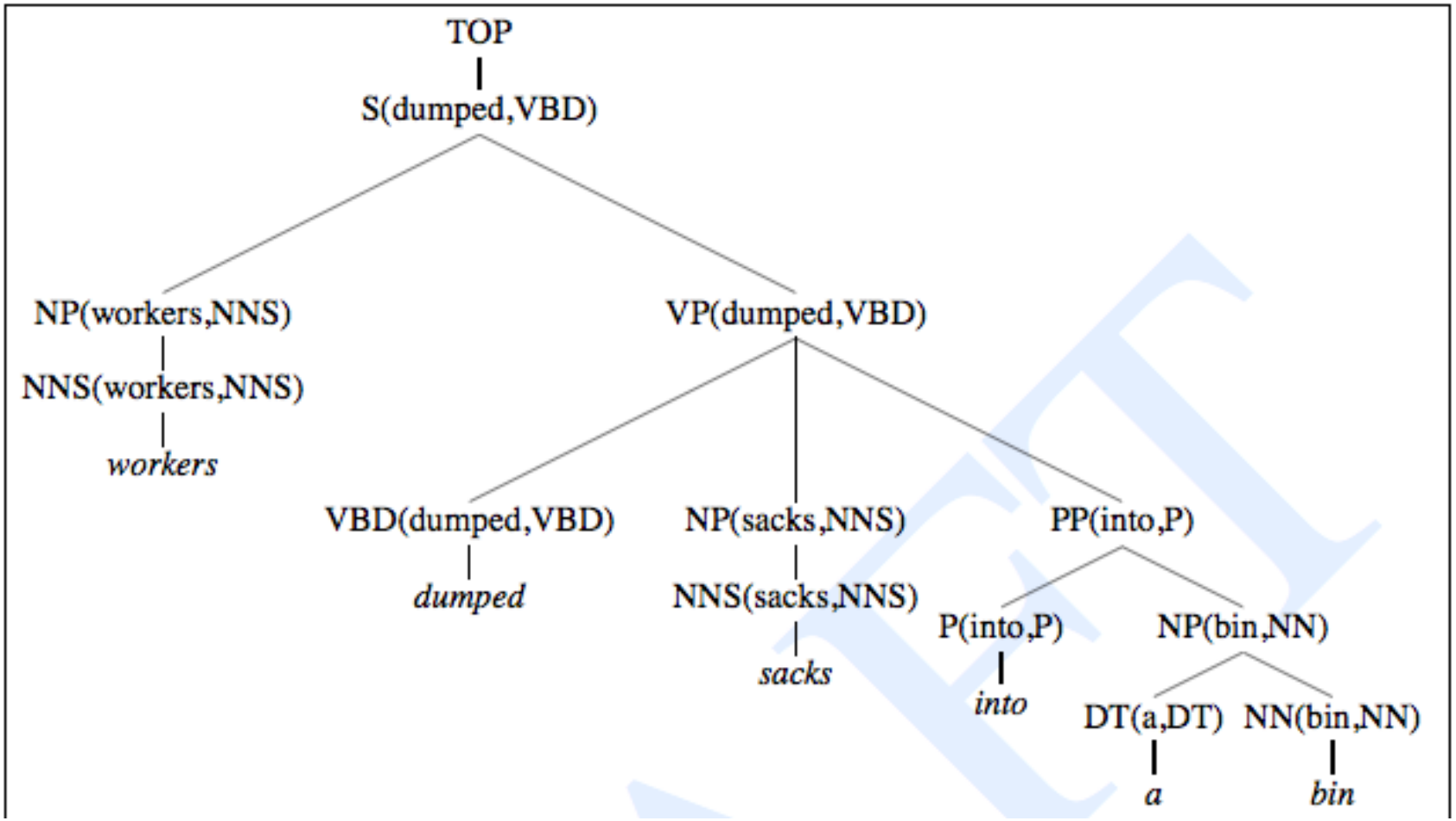
Some Related Rules

- NAC → NNP , NNP NNP 0.002463
- NAC → JJ NNP , NNP , 0.002463
- NAC → NNP , NNP NNP , 0.002463
- NAC → NNP CD , CD , 0.002463
- NAC → NNP NNP NNP , NNP , 0.002463
- NAC → NNP NNP , NNP 0.004926
- NAC → NNP NNPS , NNP , 0.007389
- NAC → NNP , NNP 0.019704
- NAC → NNP , NNP CD , CD , 0.024631
- NAC → NNP NNP , NNP , 0.125616
- NAC → NNP , NNP , 0.374384

Bigram Model for NAC



Lexicalized Rules



Markovizing Lexicalized Rules

VP+dumped+VBD → **VBD+dumped+VBD**

p(Heir = VBD+dumped+VBD | Parent = VP+dumped+VBD)

VP+dumped+VBD → **^** VBD+dumped+VBD

p(left-stop | Parent = VP+dumped+VBD, Heir = VBD+dumped+VBD)

VP+dumped+VBD → ^ VBD+dumped+VBD **NP+sacks+NNS**

p(RightChild = NP+sacks+NNS | Parent = VP+dumped+VBD, Heir = VBD+dumped+VBD)

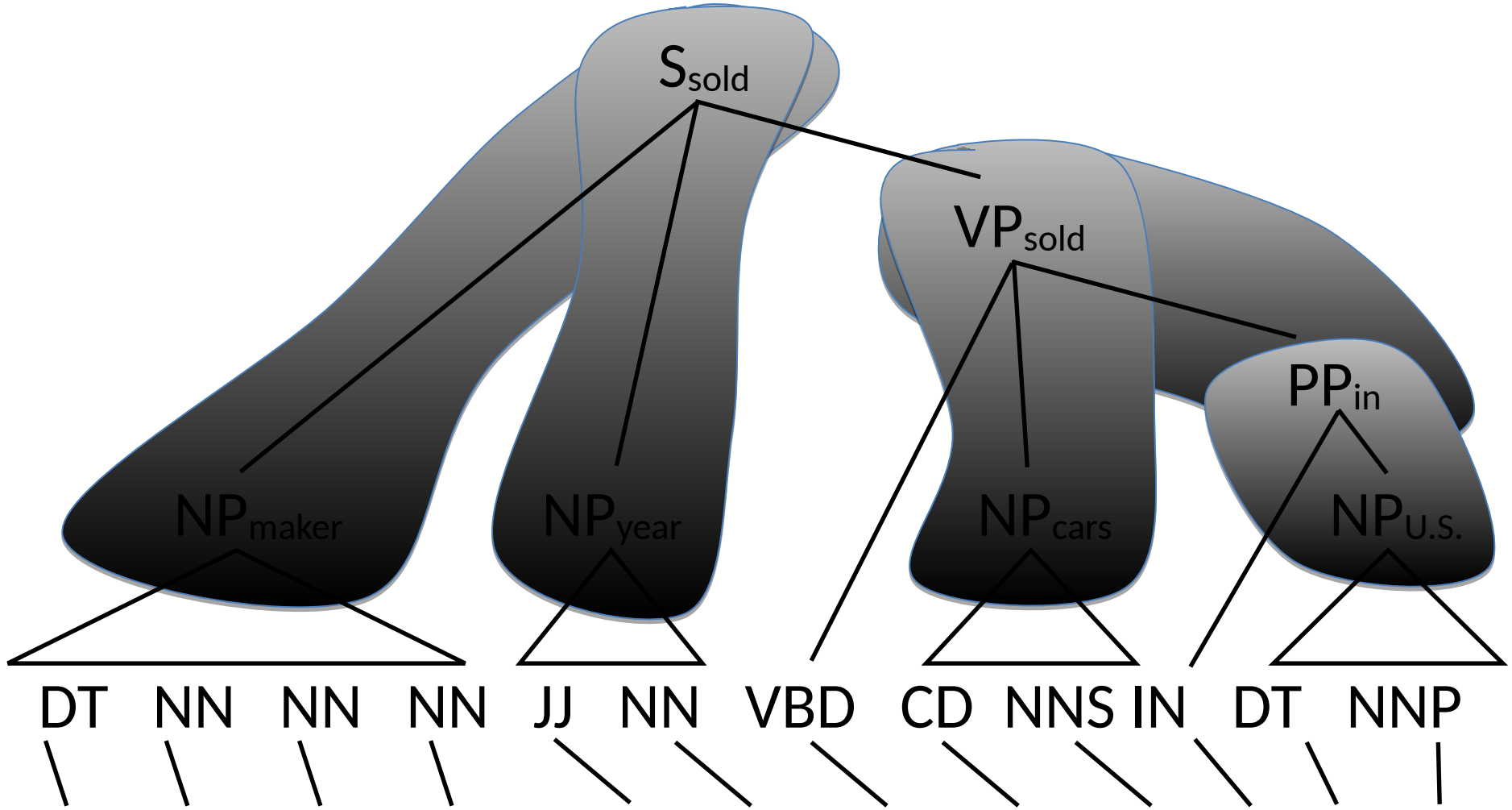
VP+dumped+VBD → ^ VBD+dumped+VBD NP+sacks+NNS **PP+into+P**

p(RightChild = PP+into+P | Parent = VP+dumped+VBD, Heir = VBD+dumped+VBD)

VP+dumped+VBD → ^ VBD+dumped+VBD NP+sacks+NNS PP+into+P **$**

p(right-stop | Parent = VP+dumped+VBD, Heir = VBD+dumped+VBD)

Dependencies in the Lexicalized Tree



Dependencies

maker → The

maker → luxury

maker → auto

sold → maker

year→ last

sold → year

$ → sold

cars→ 1,214

sold → cars

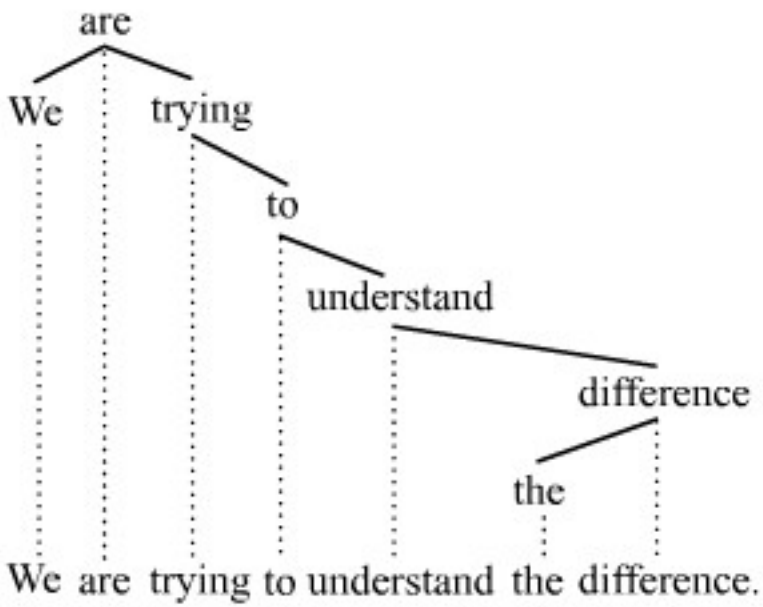
sold → in

U.S. → the

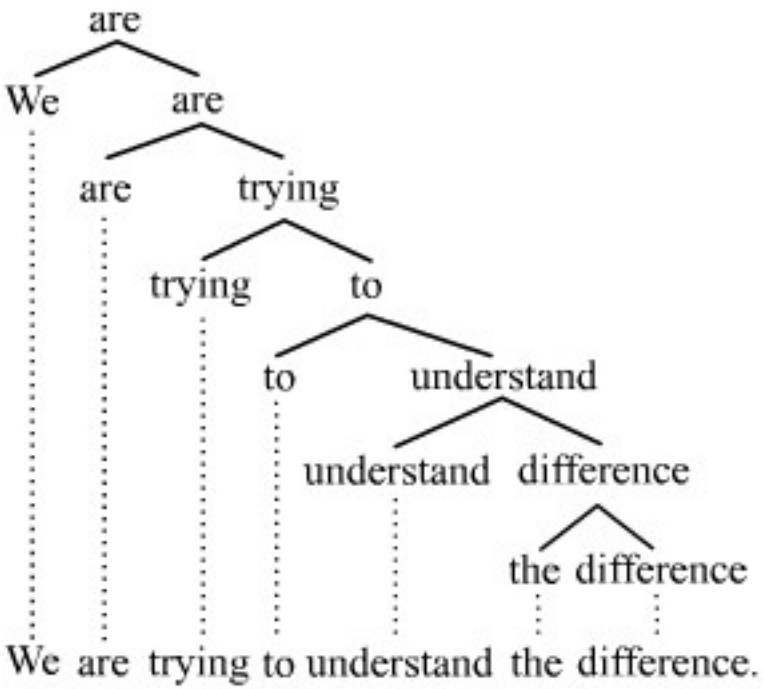
in → U.S.

The luxury auto maker last year sold 1,214 cars in the U.S.

Dependency vs Constituent



Dependency



Constituency (BPS)

By Tjo3ya – Own Work – CC by SA 3.0 via Wikipedia

Dependency Trees

- Links between heads and there dependents
  - Head is a Linguistic notion
  - Sort of “most important part”
- Only one head, acyclic
- Why?
  - Can be simpler to parse
  - Can be simpler for later ML processes

Dependency Trees

- Links between heads and there dependents
  - Head is a Linguistic notion
  - Sort of “most important part”
- Only one head, acyclic
- Why?
  - Can be simpler to parse
  - Can be simpler for later ML processes
- CT -> DT easier than DT -> CT

What is the head?

- Auxiliaries or main verbs?
  - I have written a letter.
- Prepositions or nouns?
  - A picture of my son
- Clause-initial elements? (Complementizers)
  - Who yawned?
  - I wonder which people yawned.
  - The student who yawned.
  - I think that the student yawned.
- Parts, kinds, and quantities?
  - I drank a cup of tea.
  - I drank a kind of tea.
  - I talked to a number of people.

Which word is the head?

- Lexical words

- the book
- at school
- has yawned

- Open class: you can make up new nouns and verbs

- Function words

- the book
- at school
- has yawned

- Closed class: you cannot make up new determiners, prepositions, or auxiliary verbs (although new ones can develop over time)

Stanford Dependency Parser provides two versions: lexical heads or functional heads

What you see most often in dependency treebanks

- the book
- at school
- The student has yawned
- The student has yawned
- very tall
- that the student yawned
- that the student yawned
  - As in “I think that the student yawned”

So what is the definition of “head”?

- The word that provides the main meaning:
  - “this smart student of linguistics with long hair” is a student, not a smart or a hair or a long, etc. So “student” is the head.
- The word that provides the most important inflectional features
  - Inflection includes things like tense, number, and gender

Which noun phrases are plural?

Singular

- The teacher
- The short teacher
- The teacher of the class
- The teacher of the classes
- The children's teacher
- The child's teacher

Plural

- The teachers
- The short teachers
- The teachers of the class
- The teachers of the classes
- The children's teachers
- The child's teachers

Only the head "teacher/teachers" determines whether the noun phrase is singular or plural. The other nouns "class/classes" and "child/children" do not make the noun phrase singular or plural.

Dependency Parsing

- Standard CFG (with Heads) plus CKY
  - But more computationally expensive
- Graph Algorithms
  - e.g. McDonald's MSTParse (Maximum Spanning Tree)
- Constraint satisfaction
  - Create all links and remove them (Karlsson 1990)
- Or actual parse the dependencies
  - Nivre et al 2008: MaltParser
- Neural dependency parses (Chen & Manning 2014)

Dependency Parsing

- Parse left to right
  - Make decisions about linking and shifting
- Us ML classifier to decide what to do
  - Condition on
  - Some lexical word links are more common [chair -> the]
  - Dependency distance: mostly short links
  - Intervening material: rarely span over verbs, punc
  - Valency of heads: number of expect dependents of a head

Dependency Tree

A Dependency Tree (Dutch)

1	Ze	ze	Pron	Pron	per|3|evofmv|nom	2	su
2	hadden	heb	V	V	trans|ovt|1of2of3|mv	0	ROOT
3	languit	languit	Adv	Adv	gew|geenfunc|stell|onverv	11	mod
4	naast	naast	Prep	Prep	voor	11	mod
5	elkaar	elkaar	Pron	Pron	rec|neut	4	obj1
6	op	op	Prep	Prep	voor	11	ld
7	de	de	Art	Art	bep|zijdofmv|neut	8	det
8	strandstoelen	strandstoel	N	N	soort|mv|neut	6	obj1
9	kunnen	kan	V	V	hulp|inf	2	vc
10	gaan	ga	V	V	hulp|inf	9	vc
11	liggen	lig	V	V	intrans|inf	10	vc
12	.	.	Punc	Punc	punt	11	punct

Ze hadden languit naast elkaar op de strandstoelen kunnen gaan liggen .

1 2 3 4 5 6 7 8 9 10 11 12

Other Grammar Formalisms

Unification-Based Grammars

- S → NP VP
  [NP NUMBER] = [VP NUMBER]
- Det → these
  [Det NUMBER] = plural
- MD → does
  [MD NUMBER] = singular
  [MD PERSON] = third

Categorial Grammar (CCG)

- 5 rules
- A/B + B = A
- B + A\B = A
- A/B + B/C = A/C
- A CONJ A’ = A
- A = X/(X\A)
- But the lexical items become more complex

Categorial Grammar (CCG)

```
John = np
Mary = np
likes = (s\np)/np

Forward application
 X/Y Y => X
Backward application
 Y X\Y => X

Thus
    John    likes    Mary
     np   (s\np)/np)  np
             ---------- Forward
                s\np
      ----------- Backward
            s
```

Categorial Grammar (CCG)

```
a,the np/n
old n/n
in  (np\np)/np
man,ball,park n
kicked (s\np)/np

the  old man kicked    a    ball    in     the   park
np/n n/n n  (s\np)/np np/n   n  (np\np)/np np/n  n
     -----             ------                 -----
       n                 np                     np
  ------                            -----------
    np                                np\np
                        -----------------
                                np
                -----------------
                     s\np
    -------------------
             s
```

Advanced Grammars

- Standard CFG
  - CKY vs Earley
- Lexicalized Grammars
- Other formalisms
  - TAG, HPSG, LFG
  - Unification Grammars
  - Categorial Grammars