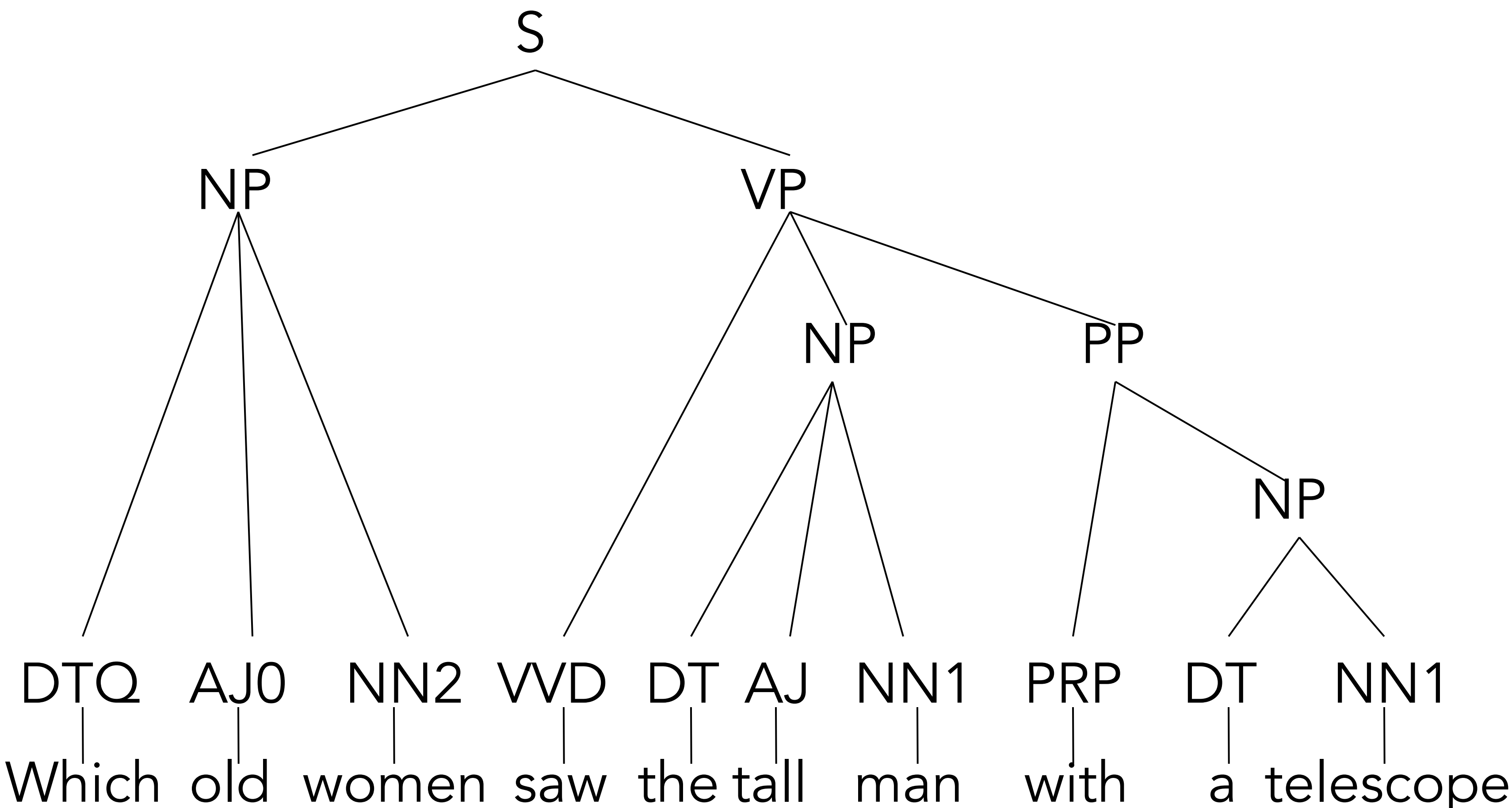
CKY Parsing

CS-585

Natural Language Processing

Derrick Higgins

Phrase Structure Trees



Parsing

- ➢ Recognize if a sentence is valid
- ➢ Figure out its syntactic structure

- Problem: Find all parse trees licensed by a given grammar, covering exactly the input words.
- Is it possible to parse a sentence deterministically as it is being read (e.g. from left-to-right)?

Computer Languages vs. Natural Languages

A parser for a computer language

- yields a unique tree for each string
- must be deterministic
- is allowed (basically) unrestricted memory

A natural language parser

- must allow for more than one parse
- should predict which parse will most likely be offered as the first choice by a native speaker
- human short-term memory is restricted

Properties of NL Parsing

- Highly ambiguous
- Different theoretical notions of *grammar*
- All solutions should be found (in principle)
- Analysis problem more complex
- Solutions based on saving partial parses

Context-Free Grammar

- Start symbol S
- Set of non-terminal symbols {NP, VP, …}
- Set of terminal symbols (words)
- Set of production rules, of the form

```
NT → a b c …
```

where NT is a non-terminal and a b c comprise a sequence of 1 or more terminals and non-terminals

Example

$$
\begin{array}{lll}
\texttt{S} & \rightarrow & \texttt{NP VP} \\
\texttt{NP} & \rightarrow & \texttt{Name} \\
\texttt{NP} & \rightarrow & \texttt{N} \\
\texttt{NP} & \rightarrow & \texttt{NP PP} \\
\texttt{VP} & \rightarrow & \texttt{V NP} \\
\texttt{VP} & \rightarrow & \texttt{V} \\
\texttt{VP} & \rightarrow & \texttt{VP PP} \\
\texttt{PP} & \rightarrow & \texttt{P NP}
\end{array}
\qquad
\begin{array}{lll}
\texttt{Name} & \rightarrow & \texttt{joe} \\
\texttt{N} & \rightarrow & \texttt{ice} \\
\texttt{N} & \rightarrow & \texttt{drinks} \\
\texttt{N} & \rightarrow & \texttt{water} \\
\texttt{V} & \rightarrow & \texttt{drinks} \\
\texttt{P} & \rightarrow & \texttt{with}
\end{array}
$$

“joe drinks water with ice”

Parser Properties

Soundness: A parser is sound if every parse returned is valid in the grammar.

Completeness: A parser is complete if for every grammar and sentence it returns all valid parses for that sentence.

- Soundness is key…
- …but completeness may be difficult or even undesirable, e.g. for highly ambiguous grammars…

Top-Down Parsing

- Start with list [S] of goals to achieve
- Iteratively rewrite the goal list by expanding rules until it matches the sentence
- Choices at each step:

1. Which rule to use if several apply to a given nonterminal
2. Which order to address the subgoals (depth-first, breadth-first, etc.)

Example

```
joe drinks water with ice
[S]
[NP,VP]
[Name,VP]
[joe,VP]
[joe,V,NP]
[joe,drinks,NP]
[joe,drinks,Name]
…
```

```
[joe,drinks,NP]
[joe,drinks,N]
[joe,drinks,ice]
…
[joe,drinks,NP]…
[joe,drinks,NP,PP]…
[joe,drinks,water,P,N]
[joe,drinks,water,with,ice]
```

Issues with Top-Down Parsing

- LR depth-first parser won't terminate if grammar has left-recursive rules
  - E.g., $\mathtt{NP} \rightarrow \mathtt{NP\ PP}$ will produce

    $\mathtt{NP} \rightarrow \mathtt{NP\ PP} \rightarrow \mathtt{NP\ PP\ PP} \rightarrow \mathtt{NP\ PP\ PP\ PP} \rightarrow \ldots$
  - Even combinations may be left-recursive:

    $\mathtt{NPos} \rightarrow \mathtt{NP}$ "'" "s" (john's book)

    $\mathtt{NP} \rightarrow \mathtt{NPos\ NP}$
- Many rules with same LHS may lead to a lot of backtracking…

Bottom-Up Parsing

- Goal list initialized as list of terminals in the string to be parsed
- If sequence of goals matches RHS of a rule, replace it with the LHS of the rule
- Parsing complete when producing S
- Choices:
  1. RHS of multiple rules may match
  2. Order of subgoals (depth-first, breadth-first)

NB: Inefficient when grammar has lexical ambiguity

Shift-Reduce Parsing

DoneGoals	String	Operation
	Nancy ate waffles today	`shift`
`NP`	ate waffles today	`shift`
`NP V`	waffles today	`shift`
`NP V NP`	today	`reduce`
`NP VP`	today	`shift`
`NP VP AV`		`reduce`
`NP VP`		`reduce`
`S`		

Effects of ambiguity

- Produces inefficiency due to backtracking
  - Multiple matching LHSs in top-down
  - Lexical ambiguity (multiple RHSs) in bottom-up
- How to produce all (most? many?) parses efficiently?
  - There may be an exponential number of them

Chart Parsing

- Remember intermediate results
- Explore all possible solutions in parallel

Sentence: w_1 w_2 w_3 w_4 …

Chart: Array whose entries show the set of categories that could generate words from n to n+m

Formally:

$$chart(m,n) = \{ A \mid A \rightarrow* \; w_n \ldots w_{n+m} \}$$

Example

The_0 man_1 drinks_2 water_3 with_4 ice_5

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4	5
0	Det	N,NP	V,VP,NP	N,NP	P	N,NP
1	NP	S	VP	{}	PP	
2	S	S	{}	NP		
3	S	{}	VP			
4	{}	S				
5	S					

Chomsky Normal Form

- Constraint on form of the grammar:
  - Each RHS is either 2 non-terminals or a terminal
- All CFGs can be written in CNF

S	$\longrightarrow$	NP VP	Det	$\longrightarrow$	the
NP	$\longrightarrow$	NP PP	NP	$\longrightarrow$	joe
NP	$\longrightarrow$	Det NP	NP	$\longrightarrow$	ice
VP	$\longrightarrow$	V NP	NP	$\longrightarrow$	drinks
VP	$\longrightarrow$	VP PP	NP	$\longrightarrow$	water
PP	$\longrightarrow$	P NP	V	$\longrightarrow$	drinks
			VP	$\longrightarrow$	drinks
			P	$\longrightarrow$	with

Cocke-Younger-Kasami (CYK)

Assume "Chomsky Normal Form" grammar

```
for n := 0 to N_w-1 do:
  chart[0, n] := {X | X → word_n }

for m := 1 to N_w-1 do:
   for n := 0 to N_w-m-1 do:
      chart[m, n] := {}
      for k := n+1 to n+m do
         for every rule A → B C do
            if B ∈ chart[k-n-1, n] and C ∈ chart[n+m-k, k] then
               chart[m, n] := chart[m, n] ∪ {A}

if S ∈ chart[N_w-1, 0] then accept else reject
```

CYK Example (in CNF)

$$\begin{aligned} S &\rightarrow NP\ VP \\ NP &\rightarrow NP\ PP \\ VP &\rightarrow V\ NP \\ VP &\rightarrow VP\ PP \\ PP &\rightarrow P\ NP \end{aligned}$$

$$\begin{aligned} NP &\rightarrow \texttt{joe} \\ NP &\rightarrow \texttt{ice} \\ NP &\rightarrow \texttt{drinks} \\ NP &\rightarrow \texttt{water} \\ V &\rightarrow \texttt{drinks} \\ VP &\rightarrow \texttt{drinks} \\ P &\rightarrow \texttt{with} \end{aligned}$$

"joe drinks water with ice"

Cocke-Younger-Kasami (CYK)

```
for n := 0 to N_w-1 do:
  chart[0, n] := {X | X → word_n }

for m := 1 to N_w-1 do:
  for n := 0 to N_w-m-1 do:
    chart[m, n] := {}
    for k [illegible]
      for every rule A → B C do
        if B ∈ chart[k-n-1, n] and C ∈ chart[n+m-k, k] then
          chart[m, n] := chart[m, n] ∪ {A}

if S ∈ chart[N_w-1, 0] then accept else reject
```

Initialize chart with terminal symbols

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

$N_w = 5$

m (constituent length -1)	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1					
2					
3					
4					

Cocke-Younger-Kasami (CYK)

Look for increasingly longer phrases

Start from the left-hand edge and stop if the constituent would run past the end of the sentence

Consider all ways you could divide the text span into two parts, and look for a rule that matches

```
for n := 0 to N_w-1 do:
  chart[0, n] := {X | X → word_n}

for m := 1 to N_w-1 do:
  for n := 0 to N_w-m-1 do:
    chart[m, n] := {}
    for k := n+1 to n+m do
      for every rule A → B C do
        if B ∈ chart[k-n-1, n] and C ∈ chart[n+m-k, k] then
          chart[m, n] := chart[m, n] ∪ {A}

if S ∈ chart[N_w-1, 0] then accept else reject
```

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S				
2					
3					
4					

$N_w = 5$

$k = n+1$

$S \longrightarrow NP\ VP$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP			
2					
3					
4					

$N_w = 5$

$k = n+1$

$VP \longrightarrow V\ NP$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}		
2					
3					
4					

$N_w = 5$

$k = n+1$

$\boxed{\emptyset}$

Example

Joe_0 $drinks_1$ $water_2$ $with_3$ ice_4

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2					
3					
4					

$N_w = 5$

$k = n+1$

PP $\longrightarrow$ P NP

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S				
3					
4					

$N_w = 5$

$k = n+1$

$S \longrightarrow NP\ VP$

Example

Joe_0 $drinks_1$ $water_2$ $with_3$ ice_4

n (constituent start index)

m (constituent length -1)	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S				
3					
4					

$N_w = 5$

$k = n+2$

$\boxed{\emptyset}$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}			
3					
4					

$N_w = 5$

$k = n+1$

$\boxed{\emptyset}$

Example

Joe_0 $drinks_1$ $water_2$ $with_3$ ice_4

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}			
3					
4					

$N_w = 5$

$k = n+2$

$\boxed{\emptyset}$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3					
4					

$N_w = 5$

$k = n+1$

$NP \longrightarrow NP\ PP$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3					
4					

$N_w = 5$
$k = n+2$

$\boxed{\emptyset}$

Example

Joe_0 drinks_1 water_2 with_3 ice_4

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}				
4					

$N_w = 5$

$k = n+1$

$\boxed{\emptyset}$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}				
4					

$N_w = 5$

$k = n+2$

$\boxed{\emptyset}$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}				
4					

$N_w = 5$

$k = n+3$

$\boxed{\emptyset}$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}	VP			
4					

$N_w = 5$

$k = n+1$

$$\boxed{\mathrm{VP} \longrightarrow \mathrm{V}\ \ \mathrm{NP}}$$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}	VP			
4					

$N_w = 5$

$k = n+2$

VP $\longrightarrow$ VP PP

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}	VP			
4					

$N_w = 5$

$k = n+3$

$\boxed{\emptyset}$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}	VP			
4	S				

$N_w = 5$

$k = n+1$

$S \longrightarrow NP\ VP$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}	VP			
4	S				

$N_w = 5$

$k = n+2$

$\boxed{\emptyset}$

Example

Joe_0 $drinks_1$ $water_2$ $with_3$ ice_4

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}	VP			
4	S				

$N_w = 5$

$k = n+3$

$\boxed{\emptyset}$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}	VP			
4	S				

$N_w = 5$

$k = n+4$

$\boxed{\emptyset}$

Example

Joe$_0$ drinks$_1$ water$_2$ with$_3$ ice$_4$

n (constituent start index)

$N_w = 5$

m (constituent length -1)

	0	1	2	3	4
0	NP	V,VP,NP	NP	P	NP
1	S	VP	{}	PP	
2	S	{}	NP		
3	{}	VP			
4	S				

Generalizing to non-CNF grammars

- Well-Formed Substring Table (WFST)
  - Same as CYK table, but allows RHS of rules to have many non-terminals

- Complexity: $O(n^{k+1})$ where k is largest number of non-terminals in a rule

WFST Example

$S \longrightarrow NP\ VP$

$NP \longrightarrow NP\ PP$

$VP \longrightarrow V_{trans}\ NP$

$VP \longrightarrow V_{ditrans}\ NP\ NP$

$VP \longrightarrow VP\ PP$

$PP \longrightarrow P\ NP$

$NP \longrightarrow$ joe

$NP \longrightarrow$ mary

$NP \longrightarrow$ ice

$NP \longrightarrow$ drinks

$NP \longrightarrow$ water

$V_{trans} \longrightarrow$ drinks

$V_{ditrans} \longrightarrow$ gives

$P \longrightarrow$ with

"joe gives mary water"

Active Charts

- WFST stores complete analyses....

  ...but this requires redoing partial results:

  $\mathtt{VP} \rightarrow \mathbf{V_{ditrans}}\ \mathbf{NP}\ \mathtt{PP_{to}}$

  $\mathtt{VP} \rightarrow \mathbf{V_{ditrans}}\ \mathbf{NP}\ \mathtt{NP}$

Idea: Store partial results as well!

- Active chart contains:
  - Passive items: complete results
  - Active items: partial results

Dotted Rules

- Partial results (in chart) must consider how much has been recognized so far:

Dotted rule	For a VP we still need...
$\mathtt{VP} \rightarrow \blacklozenge\mathtt{Vdit\ NP\ PP}_{\mathtt{to}}$	$\mathtt{Vdit}$, $\mathtt{NP}$, and $\mathtt{PP}_{\mathtt{to}}$
$\mathtt{VP} \rightarrow \mathtt{Vdit}\ \blacklozenge\mathtt{NP\ PP}_{\mathtt{to}}$	$\mathtt{NP}$ and $\mathtt{PP}_{\mathtt{to}}$
$\mathtt{VP} \rightarrow \mathtt{Vdit\ NP}\ \blacklozenge\mathtt{PP}_{\mathtt{to}}$	$\mathtt{PP}_{\mathtt{to}}$
$\mathtt{VP} \rightarrow \mathtt{Vdit\ NP\ PP}_{\mathtt{to}}\blacklozenge$	nothing