Natural Language Processing: CIS 410/510

Constituent Parsing

Instructor: Thien Huu Nguyen

Based on slides from: Ralph Grishman, David Bamman, Dan Jurasky, and others

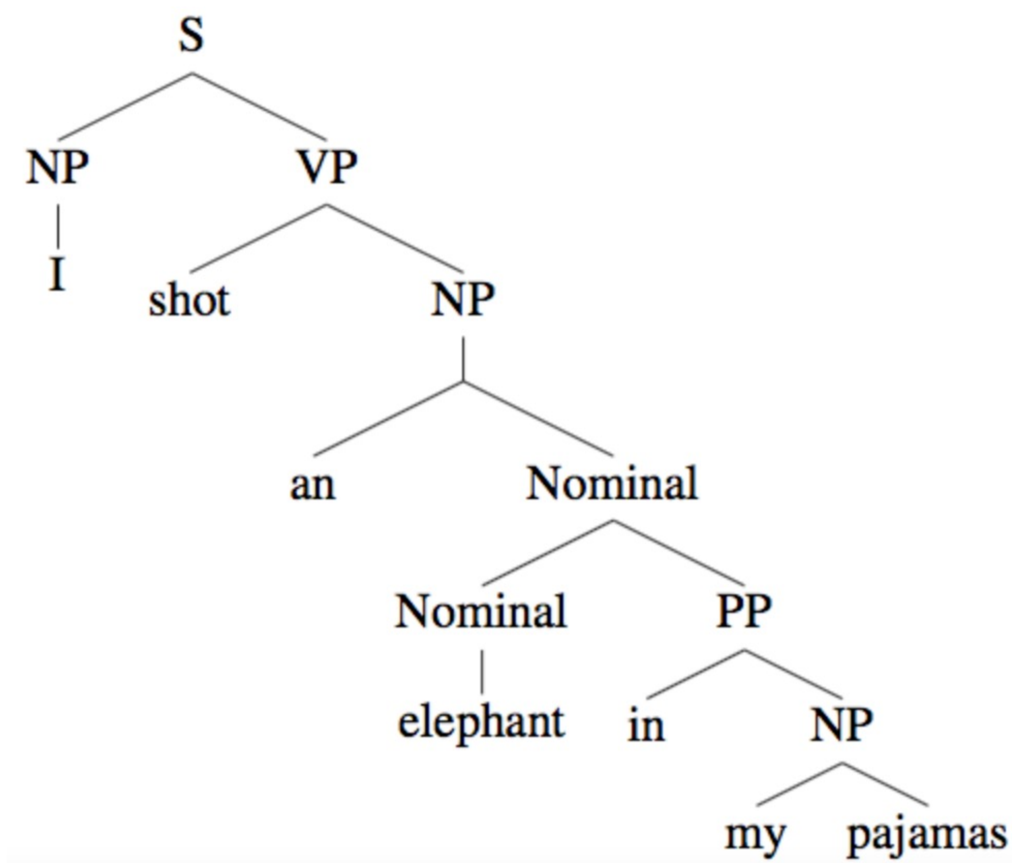
Syntax

- With syntax, we're moving from labels for discrete items - documents (sentiment analysis), tokens (POS tagging, NER) - to the structure between items.
- Syntax is fundamentally about the hierarchical structure of language and (in some theories) which sentences are grammatical in a language

  words → phrases → clauses → sentences

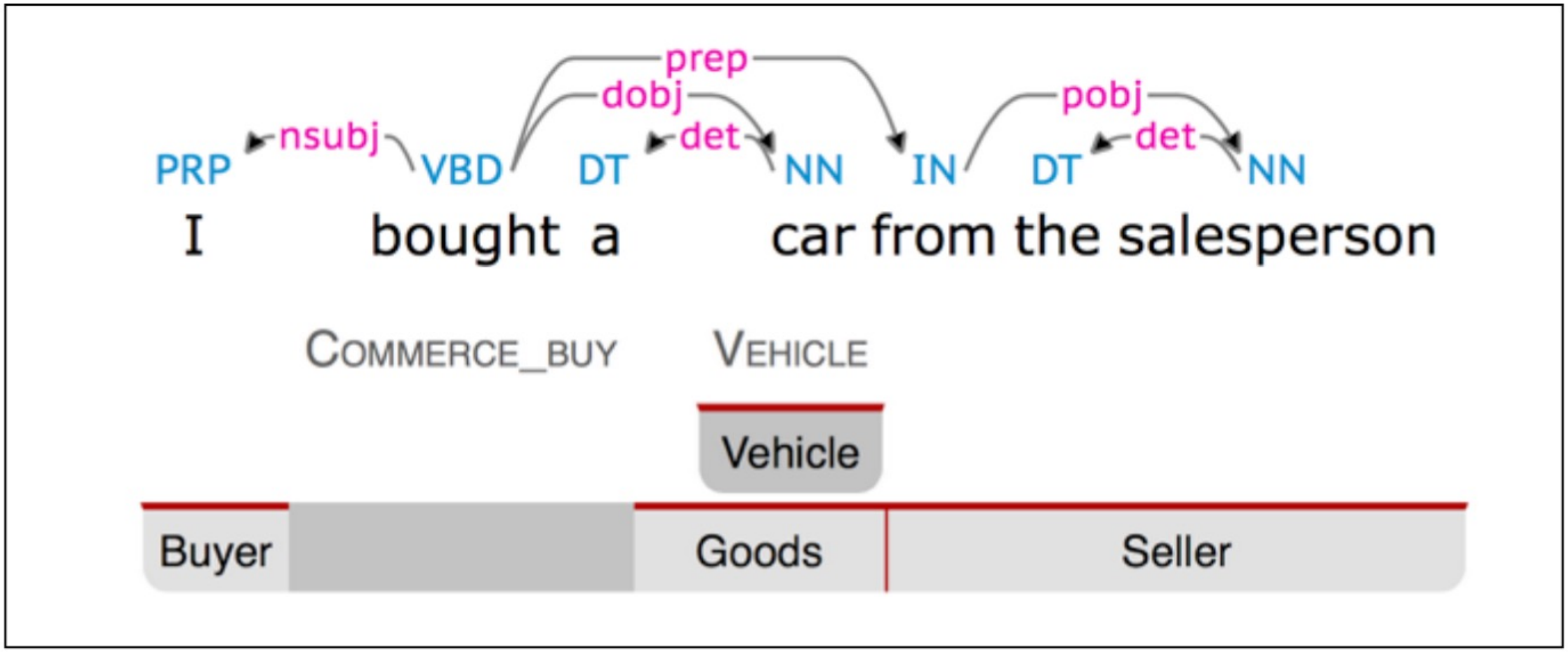
PRP VBD DT NN IN PRP$ NNS

I shot an elephant in my pajamas



Why is syntax important?

- Foundation for semantic analysis (on many levels of representation: semantic roles, compositional semantics, frame semantics)
- Humans communicate complex ideas by composing words together into bigger units to convey complex meanings



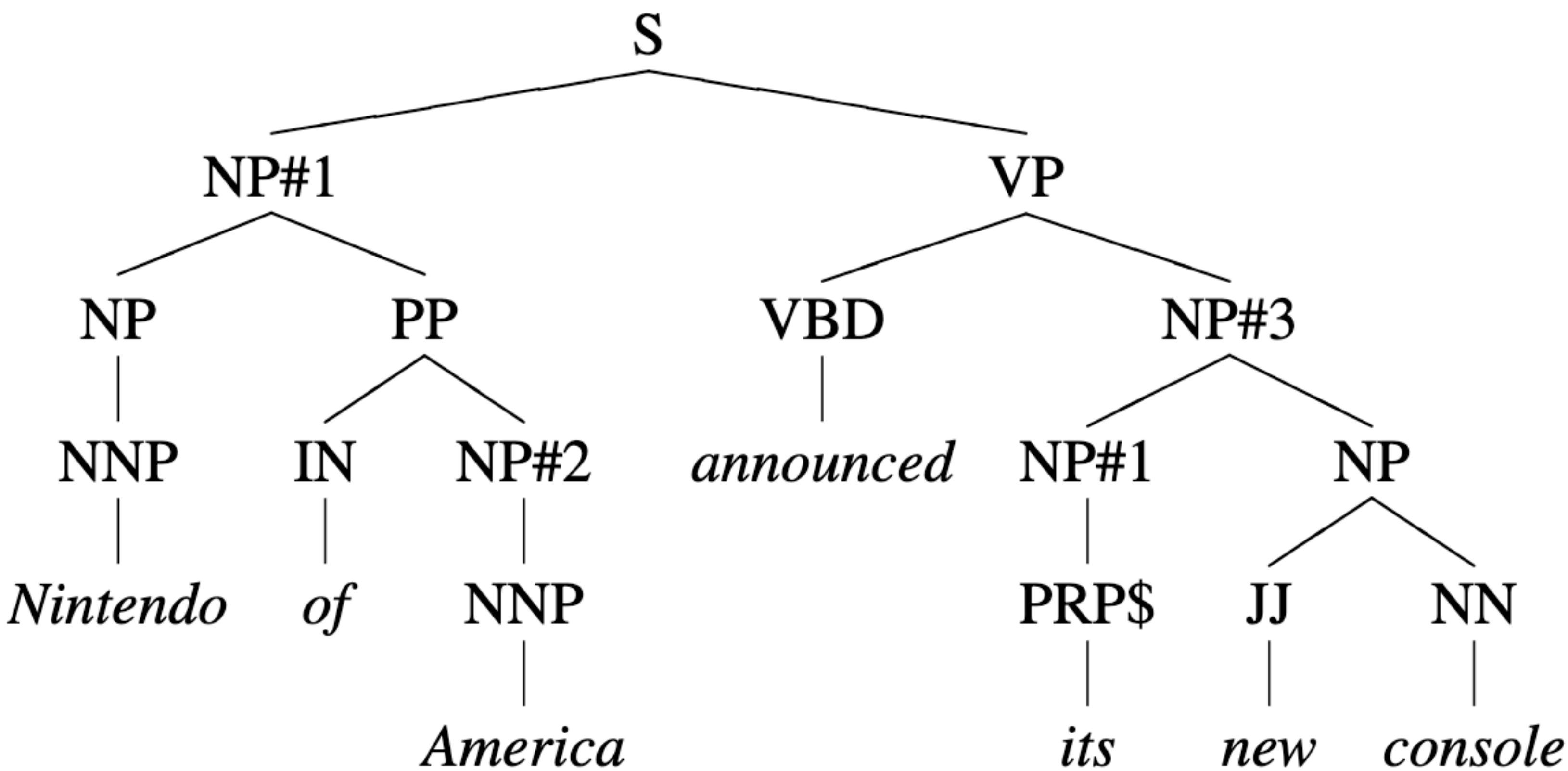
Why is syntax important?

- Linguistic typology; relative positions of subjects (S), objects (O) and verbs (V)

SVO	English, Mandarin	I grabbed the chair
SOV	Latin, Japanese	I the chair grabbed
VSO	Hawaiian	Grabbed I the chair
OSV	Yoda	Patience you must have
…	…	…

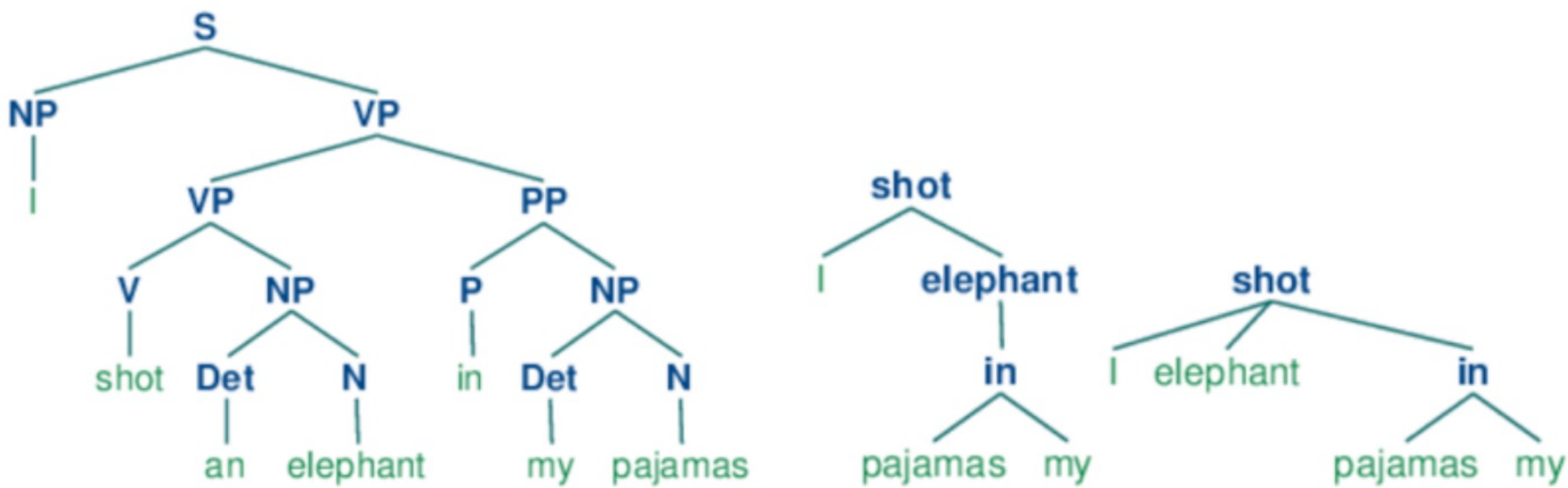
Why is syntax important?

- Strong representation for discourse analysis (e.g., coreference resolution)



https://en.wikipedia.org/wiki/Discourse_analysis

Formalisms



Phrase structure grammar
(Chomsky 1957)

Dependency grammar
(Mel'čuk 1988; Tesnière 1959; Pāṇini)

Constituency

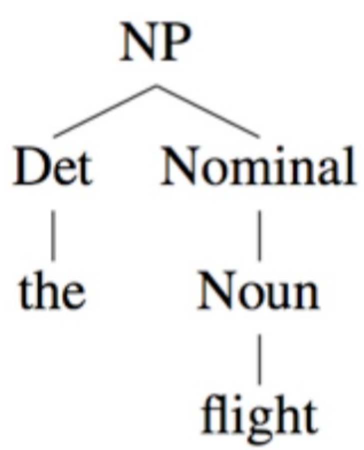
- Groups of words ("constituents") behave as single units
- "Behave" = show up in the same distributional environments as single units (e.g., the substitution test)

- Substitution test for POS: if a word is replaced by another word, does the sentence remain grammatical?

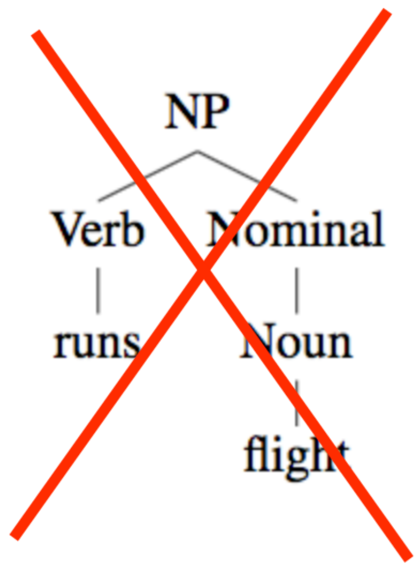
- Substitution test for Constituency: if a constituent is replaced by another constituent of the same type, does the sentence remain grammatical?

Context-free grammar (CFG)

- A CFG gives a formal way to define what meaningful constituents are and exactly how a constituent is formed out of other constituents (or words). It defines valid structure in a language (i.e., defining how symbols in a language combine to form valid structures)



NP → Det Nominal



NP → Verb Nominal

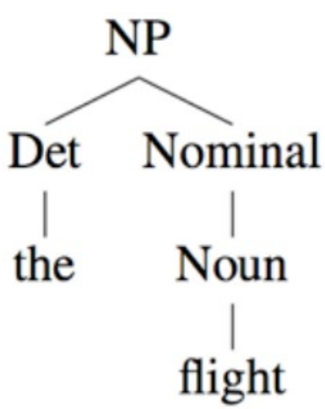
Context-free grammar (CFG)

N	Finite set of non-terminal symbols	NP, VP, S
Σ	Finite alphabet of terminal symbols	the, dog, eat
R	Set of production rules, each of the form $A \rightarrow \beta, \beta \in (\Sigma \cup N)*$	S → NP VP Noun → dog
S	A designed start symbol	

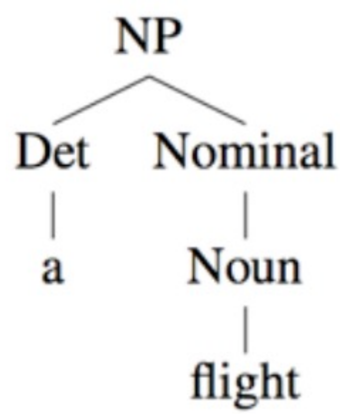
Derivation

- Given a CFG, a derivation is the sequence of productions used to generate a string of words/terminal symbols (e.g., a sentence), often visualized as a parse tree.

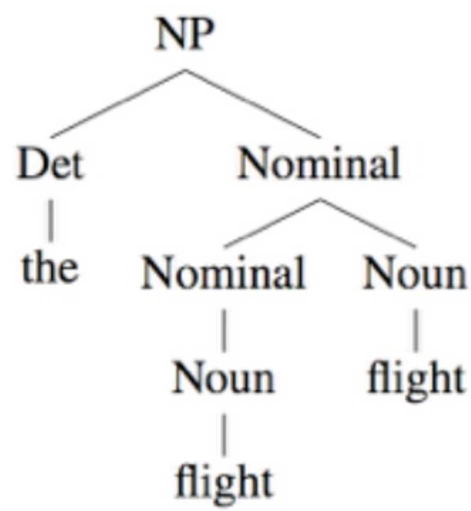
NP VP ➔ cats VP ➔ cats chase NP



the flight



a flight



the flight flight

Language

- The strings of words (e.g., sentences) are called as "derivable from the start symbol (S)"

- The formal language defined by a CFG is the set of strings derivable from S

S➔ NP VP ➔cats VP ➔ cats chase NP ➔ cats chase mice

Preterminals

- It is convenient to include a set of symbols called *preterminals* (corresponding to the parts of speech) which can be directly rewritten as terminals (words)
- This allows us to separate the productions into a set which generates sequences of preterminals (the "grammar") and those which rewrite the preterminals as terminals (the "dictionary")

Grouping Alternates

- To make the grammar more compact, we group productions with the same left-hand side:

S → NP VP

NP → N | ART N | ART ADJ N

VP → V | V NP

Example

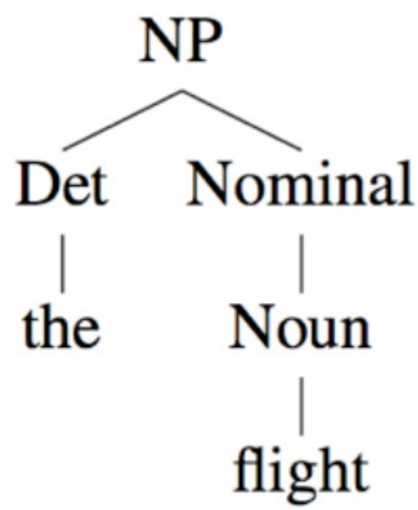
Noun →	*flights* | *breeze* | *trip* | *morning*
Verb →	*is* | *prefer* | *like* | *need* | *want* | *fly*
Adjective →	*cheapest* | *non-stop* | *first* | *latest*
	| *other* | *direct*
Pronoun →	*me* | *I* | *you* | *it*
Proper-Noun →	*Alaska* | *Baltimore* | *Los Angeles*
	| *Chicago* | *United* | *American*
Determiner →	*the* | *a* | *an* | *this* | *these* | *that*
Preposition →	*from* | *to* | *on* | *near*
Conjunction →	*and* | *or* | *but*

Figure 12.2 The lexicon for $\mathscr{L}_0$.

Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

Figure 12.3 The grammar for $\mathscr{L}_0$, with example phrases for each rule.

Bracketed notation

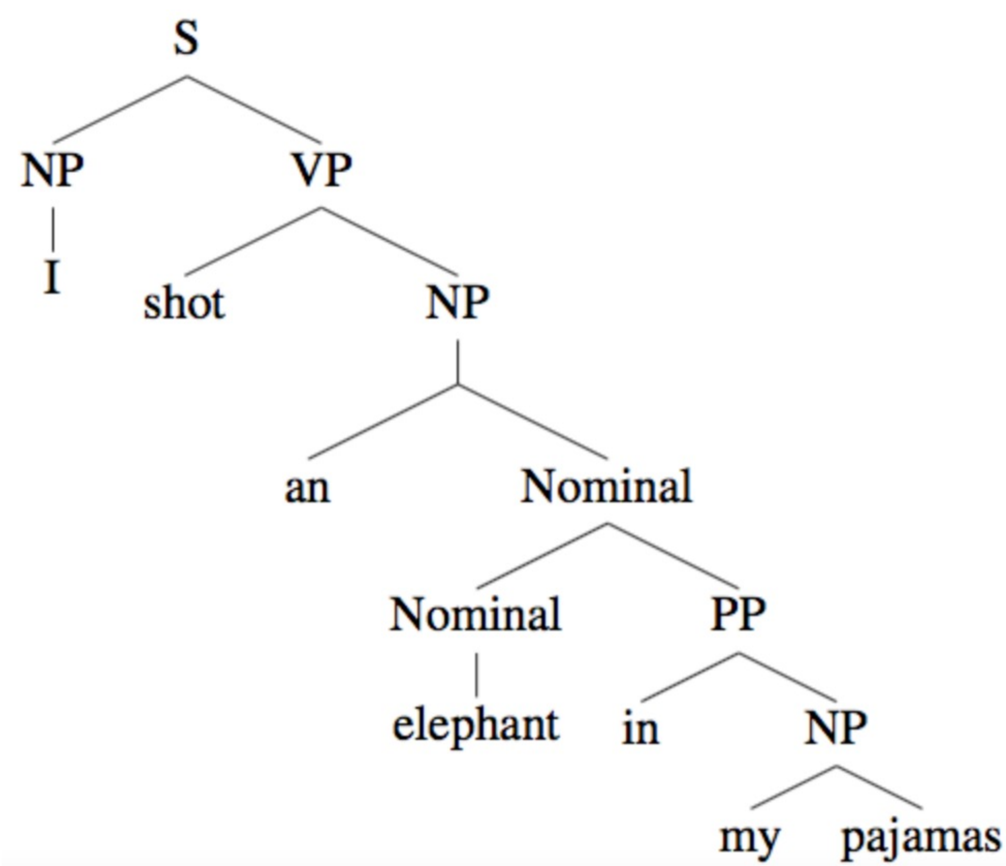


$[_{NP}\ [_{Det}\ \text{the}]\ [_{Nominal}\ [_{Noun}\ \text{flight}]]]$

Constituents

Every internal node is a phrase

- my pajamas
- in my pajamas
- elephant in my pajamas
- an elephant in my pajamas
- shot an elephant in my pajamas
- I shot an elephant in my pajamas



Each phrase could be replaced by another of the same type of constituent

Sentence

Rule	Description	Example
S → VP	Imperative	• Show me the right way
S → NP VP	Declarative	• The dog barks
S → Aux VP NP	Yes/no questions	• Will you show me the right way?

Noun Phrases

- NP → Pronoun | Proper-noun | Det Nominal
- Nominal → Nominal PP
  - An elephant [$_{PP}$ in my pajamas]
  - The cat [$_{PP}$ on the floor] [$_{PP}$ under the table] [$_{PP}$ next to the dog]
- Nominal → RelClause, RelClause → (who|that) VP : A relative pronoun (that, which) in a relative clause can be the subject or object of the embedded verb.
  - A flight [$_{RelClause}$ that serves breakfast]
  - A flight [$_{RelClause}$ that I got]

Verb Phrases

VP → Verb	disappear
VP → Verb NP	prefer a morning flight
VP → Verb NP PP	prefer a morning flight on Monday
VP → Verb PP	leave on Wednesday
VP → Verb S	I think [$_S$ I want a new flight]
VP → Verb VP	want [$_{VP}$ to fly today]

Not every verb can appear in each of these productions

Verb Phrases

VP → Verb	* I filled
VP → Verb NP	* I exist the morning flight
VP → Verb NP PP	* I exist the morning flight on Monday
VP → Verb PP	* I filled on Wednesday
VP → Verb S	* I exist [$_S$ I want a new flight]
VP → Verb VP	* I fill [$_{VP}$ to fly today]

Not every verb can appear in each of these productions

Subcategorization

- Verbs are compatible with different complements
  - Transitive verbs take direct object NP ("I filled the tank")
  - Intransitive verbs don't ("I exist")

- The set of possible complements of a verb is its subcategorization frame.

VP	→	Verb VP	* I fill [$_{VP}$ to fly today]
VP	→	Verb VP	I want [$_{VP}$ to fly today]

Coordination

NP → NP and NP	the dogs and the cats
Nominal → Nominal and Nominal	dogs and cats
VP → VP and VP	I came and saw and conquered
JJ → JJ and JJ	beautiful and red
S → S and S	I came and I saw and I conquered

Ambiguity

- Most sentences will have more than one parse
- Generally different parses will reflect different meanings …
  - Attachment ambiguity: a particular constituent can be attached to the parse tree at more than one place

    "I saw the man with a telescope."

    Can attach PP ("with a telescope") under NP or VP

  - Coordination ambiguity: different sets of phrases can be conjoined by a conjunction like "and":

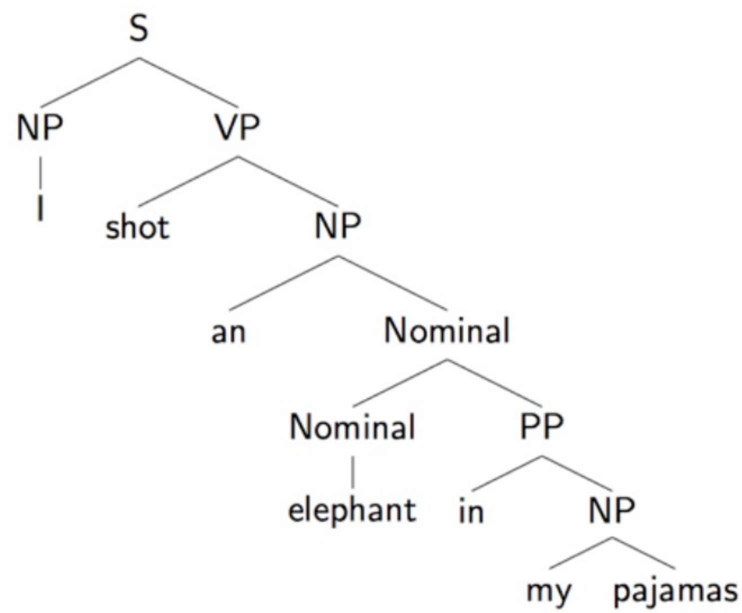
    "old man and woman" -> [old [men and women]] or [[old man] and [woman]]?

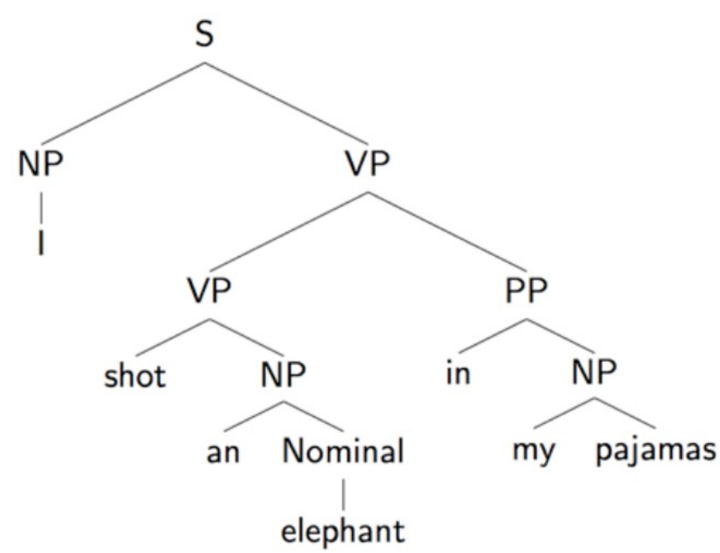
Example

S	→	NP VP
VP	→	Verb NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	Noun
Nominal	→	Pronoun
PP	→	Prep NP
NP	→	Det Nominal
NP	→	Nominal
NP	→	PossPronoun Nominal

Verb	→	shot
Det	→	an | my
Noun	→	pajamas | elephant
Pronoun	→	I
PossPronoun	→	my

I shot an elephant in my pajamas



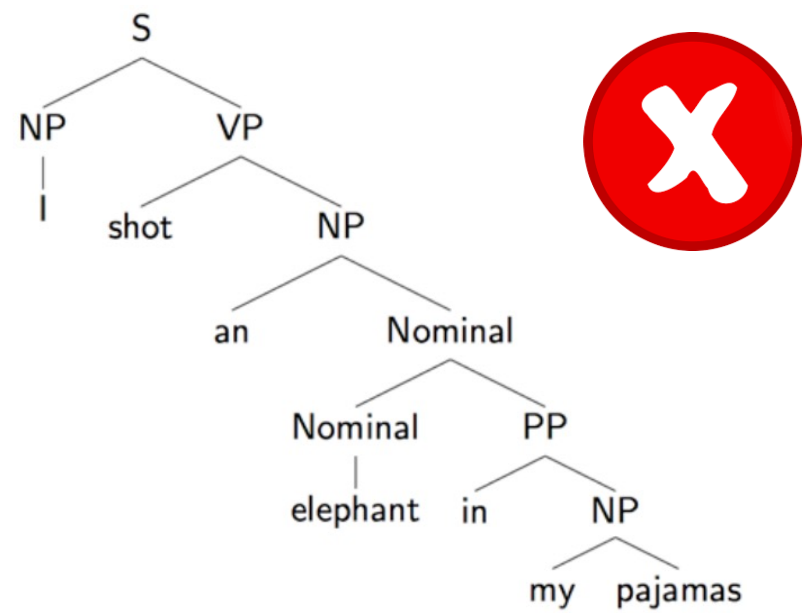


Evaluation

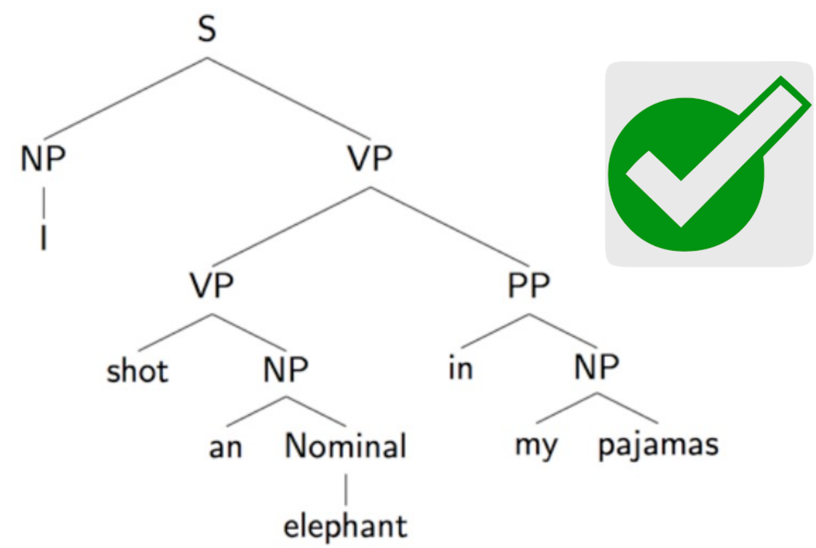
- Parseval (1991): represent each tree as a collection of tuples.
- Calculate precision, recall, F1 from these collections of tuples

$$< l_1, i_1, j_1 >, \ldots, < l_n, i_n, j_n >$$

  - l_k: label for the k-th phrase
  - i_k: index for the first word in the k-th phrase
  - j_k: index for the last word in the k-th phrase



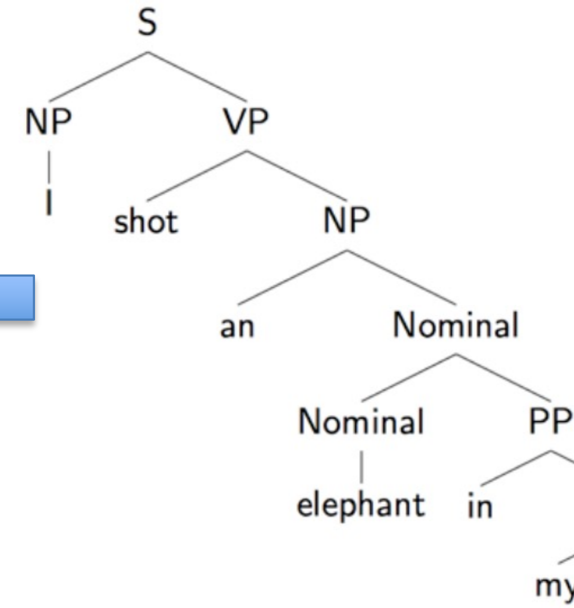
- <S, 1, 7>
- <NP, 1,1>
- <VP, 2, 7>
- <VP, 2, 4>
- <NP, 3, 4>
- <Nominal, 4, 4>
- <PP, 5, 7>
- <NP, 6, 7>



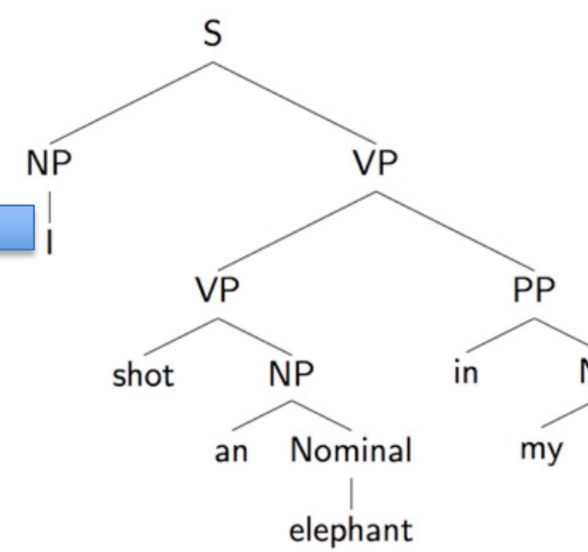
Evaluation

- Precision (P) = number of tuples in the predicted tree also in correct tree, divided by number of tuples in the predicted tree = 5/7
- Recall (R) = number of tuples in the predicted tree also in correct tree, divided by number of tuples in the correct tree = 5/7
- $F1 = \frac{2PR}{P+R}$

- <S, 1, 7>
- <NP, 1,1>
- <VP, 2, 7>
- <NP, 3, 7>
- <Nominal, 4, 7>
- <Nominal, 4, 4>
- <PP, 5, 7>
- <NP, 6, 7>



- <S, 1, 7>
- <NP, 1,1>
- <VP, 2, 7>
- <VP, 2, 4>
- <NP, 3, 4>
- <Nominal, 4, 4>
- <PP, 5, 7>
- <NP, 6, 7>



CFGs

- Building a CFG by hand is really hard
- To capture all (and only) grammatical sentences, need to exponentially increase the number of categories (e.g., detailed subcategorization info)

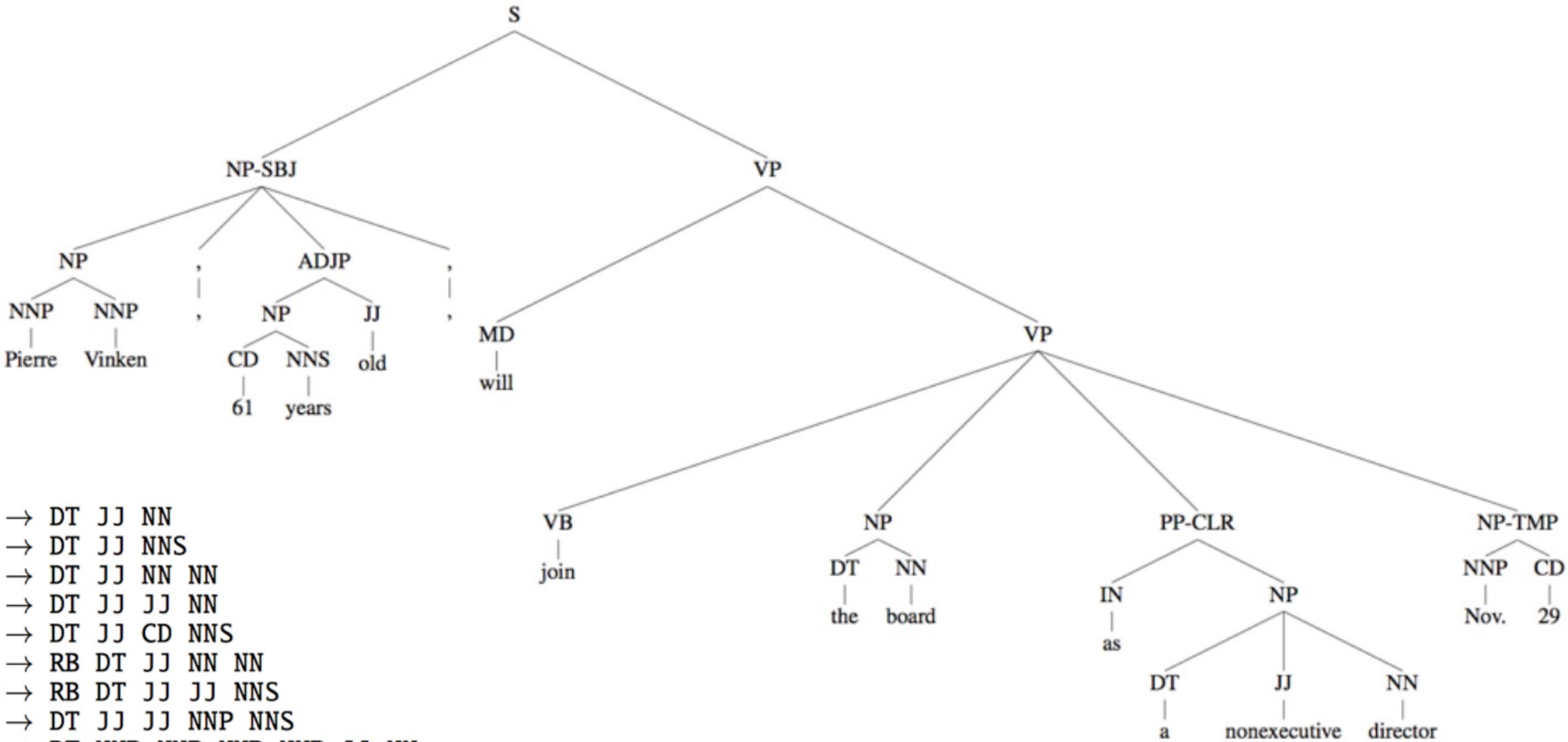
Verb-with-no-complement	→	disappear
Verb-with-S-complement	→	said
VP	→	Verb-with-no-complement
VP	→	Verb-with-S-complement S

Treebanks

- Rather than create the rules by hand, we can annotate sentences with their syntactic structure and then extract the rules from the annotations

- Treebanks: collections of sentences annotated with syntactic structure (e.g., Penn Treebank)

Penn Treebank



NP → DT JJ NN
NP → DT JJ NNS
NP → DT JJ NN NN
NP → DT JJ JJ NN
NP → DT JJ CD NNS
NP → RB DT JJ NN NN
NP → RB DT JJ JJ NNS
NP → DT JJ JJ NNP NNS
NP → DT NNP NNP NNP NNP JJ NN
NP → DT JJ NNP CC JJ JJ NN NNS
NP → RB DT JJS NN NN SBAR
NP → DT VBG JJ NNP NNP CC NNP
NP → DT JJ NNS , NNS CC NN NNS NN
NP → DT JJ JJ VBG NN NNP NNP FW NNP
NP → NP JJ , JJ ‘‘ SBAR ’’ NNS

NP	→	NNP NNP
NP-SBJ	→	NP , ADJP ,
S	→	NP-SBJ VP
VP	→	VB NP PP-CLR NP-TMP

Example rules extracted from this single annotation

How to parse?

- Given a CFG and a sentence, how can we obtain the parse tree(s) for the sentence?
  - Top-down parsing
    - repeat
      - expand leftmost non-terminal using first production (save any alternative productions on backtrack stack)
      - if we have matched entire sentence, quit (success)
      - if we have generated a terminal which doesn't match sentence, pop choice point from stack (if stack is empty, quit (failure))
  - Bottom-up parsing
  - Inefficiency: the top-down parsers waste effort to explore trees that are not consistent with the input while the bottom-up parsers waste effort to explore trees that cannot lead to the start symbol S. See SLP2 for details

→ Dynamic programming parsing, i.e., CYK parsing (Cocke-Kasami-Younger)

Chomsky Normal Form (CNF)

N	Finite set of non-terminal symbols	NP, VP, S
Σ	Finite alphabet of terminal symbols	the, dog, eat
R	Set of production rules, each of the form $A \rightarrow \beta, \beta \in (\Sigma \cup N)*$ where β = a single terminal in Σ or two non-terminals in N	S → NP VP Noun → dog
S	A designed start symbol	

Chomsky Normal Form (CNF)

- Any CFG can be converted into a weakly equivalent CNF (recognizing the same set of sentences as existing in the grammar but differing in their derivation).

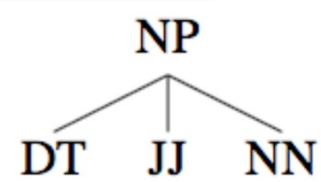
Case 1: mix of terminals and non-terminals

INF-VP → to VP

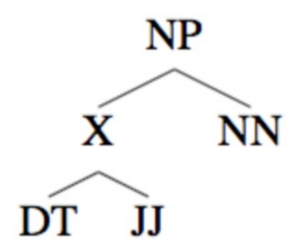
INF-VP → TO VP
TO → to

Case 2: more than 2 non-terminals

NP → DT JJ NN



NP → X NN
X → DT JJ



CNF conversion

Case 3: single non-terminal

$A \rightarrow^{*} B$
$B \rightarrow \gamma$

$A \rightarrow \gamma$

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	NN
Nominal	→	NNS
Nominal	→	PRP
PP	→	IN NP
NP	→	DT NN
NP	→	Nominal
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
NN	→	elephant
NNS	→	pajamas
PRP	→	I
PRP$	→	my
IN	→	in

I shot an elephant in my pajamas

CNF conversion

Case 3: single non-terminal

$A \rightarrow^* B$
$B \rightarrow \gamma$

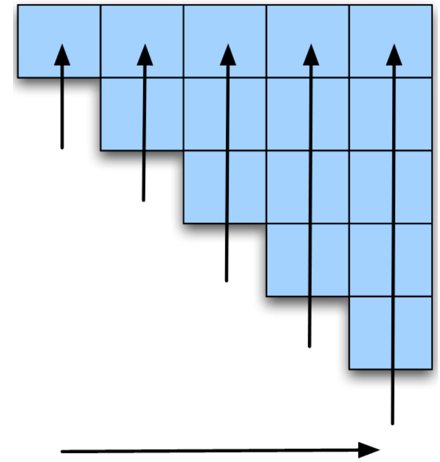
$A \rightarrow \gamma$

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

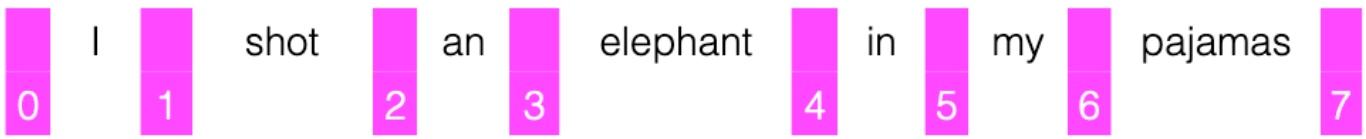
VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I shot an elephant in my pajamas

CYK parsing



- For parsing from a grammar expressed in CNF
- Bottom-up dynamic programming



function CKY-PARSE(*words, grammar*) **returns** *table*

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{A \mid A \rightarrow words[j] \in grammar\}$
    $table[j-1, j] \leftarrow table[j-1, j] \cup A$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
        $table[i,j] \leftarrow table[i,j] \cup A$

Figure 13.5 The CKY algorithm.

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]						
	VBD [1,2]					
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

Each cell i,j keeps track of all phrase types that can be formed from *all* words from position i through position j

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]						
	VBD [1,2]					
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

What phrases can be formed from "shot an elephant in"

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]						
	VBD [1,2]					
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

What phrases can be formed from “I shot an elephant in my pajamas”

CNF

- In CNF, each non-terminal generates two non-terminals

  S → NP VP

  [$_S$ [$_{NP}$ I] [$_{VP}$ shot an elephant in my pajamas]]

- If the left-side non-terminal spans tokens $i - j$, the right side must also span $i - j$, and there must be a single position k that separates them.

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]						
	VBD [1,2]					
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

Does any rule generate PRP VBD?

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅					
	VBD [1,2]					
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

Does any rule generate VBD DT?

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅					
	VBD [1,2]					
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

Does any rule generate VBD DT?

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅					
	VBD [1,2]	∅				
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

Two possible places look for that split k

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅					
	VBD [1,2]	∅				
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

Two possible places look for that split k

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅				
		DT [2,3]				
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

Does any rule generate DT NN?

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅				
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

Two possible places look for that split k

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅				
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

Two possible places look for that split k

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅	VP [1,4]			
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

Three possible places look for that split k

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅	VP [1,4]			
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

Three possible places look for that split k

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅	VP [1,4]			
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

Three possible places look for that split k

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅				
	VBD [1,2]	∅	VP [1,4]			
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

Three possible places look for that split k

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]			
	VBD [1,2]	∅	VP [1,4]			
		DT [2,3]	NP [2,4]			
			NP, NN [3,4]			
				IN [4,5]		
					PRP$ [5,6]	
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	
			NP, NN [3,4]	∅	∅	
				IN [4,5]	∅	
					PRP$ [5,6]	
						NNS [6,7]

*elephant in
*an elephant in
*shot an elephant in
*I shot an elephant in

*in my
*elephant in my
*an elephant in my
*shot an elephant in my
*I shot an elephant in my

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

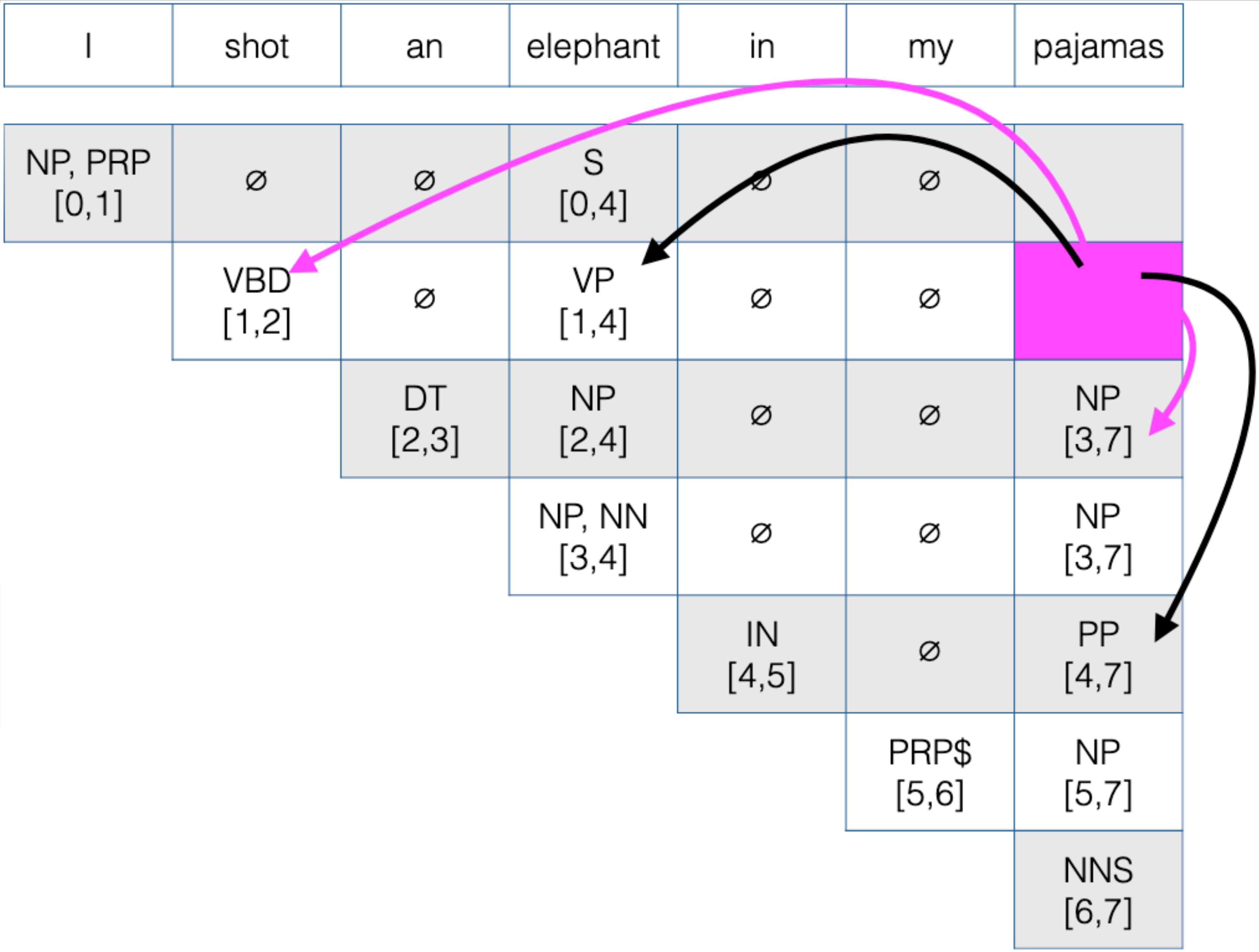
S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	
		DT [2,3]	NP [2,4]	∅	∅	NP [3,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in



I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	VP_1, VP_2 [1,7]
		DT [2,3]	NP [2,4]	∅	∅	NP [2,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	
	VBD [1,2]	∅	VP [1,4]	∅	∅	VP_1, VP_2 [1,7]
		DT [2,3]	NP [2,4]	∅	∅	NP [2,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

Possibilities:

S_1 → NP VP_1
S_2 → NP VP_2
? → S PP
? → PRP VP_1
? → PRP VP_2

I	shot	an	elephant	in	my	pajamas
NP, PRP [0,1]	∅	∅	S [0,4]	∅	∅	S_1, S_2 [0,7]
	VBD [1,2]	∅	VP [1,4]	∅	∅	VP_1, VP_2 [1,7]
		DT [2,3]	NP [2,4]	∅	∅	NP [2,7]
			NP, NN [3,4]	∅	∅	NP [3,7]
				IN [4,5]	∅	PP [4,7]
					PRP$ [5,6]	NP [5,7]
						NNS [6,7]

S	→	NP VP
VP	→	VBD NP
VP	→	VP PP
Nominal	→	Nominal PP
Nominal	→	pajamas | elephant | I
PP	→	IN NP
NP	→	DT NN
NP	→	pajamas | elephant | I
NP	→	PRP$ Nominal

VBD	→	shot
DT	→	an | my
PRP	→	I
PRP$	→	my
IN	→	in

Success! We've recognized a total of two valid parses

Complexity?

CFG

- CYK allows us to:
  - check whether a sentence in grammatical in the language defined by the CFG
  - enumerate all possible parses for a sentence CFG
- But it doesn't tell us on which one of those possible parses is most likely
  - might help to to disambiguate

-> Probabilistic context-free grammar

Probabilistic context-free grammar (PCFG)

- Probabilistic context-free grammar: each production is also associated with a probability.

N	Finite set of non-terminal symbols	NP, VP, S
Σ	Finite alphabet of terminal symbols	the, dog, eat
R	Set of production rules, each of the form $A \rightarrow \beta[p], \beta \in (\Sigma \cup N)*$ $p = P(\beta \mid A)$	S → NP VP Noun → dog
S	A designed start symbol	

Probabilistic context-free grammar (PCFG)

- We can then calculate the probability of a parse for a given sentence
- For a given parse tree T for sentence S comprised of n rules from R (each $A \rightarrow \beta$):

$$P(T) = \prod_{i=1}^{n} P(\beta|A)$$

- In practice, we often want to find the single best parse with the highest probability for a given tree S:

$$T^*(S) = argmax_T P(T|S) = argmax_T \frac{P(S|T)P(T)}{P(S)}$$

$$= argmax_T P(S|T)P(T) = argmax_T P(T)$$

- We calculate the max probability parse using CKY by storing the max probability of each phrase within each cell as we build it up.

Probabilistic CYK for PCFG

function PROBABILISTIC-CKY(*words,grammar*) **returns** most probable parse and its probability

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{ A \mid A \rightarrow words[j] \in grammar\}$
      $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
      for $k \leftarrow i+1$ **to** $j-1$ **do**
        for all $\{ A \mid A \rightarrow BC \in grammar,$ **and** $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$
          if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**
            $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
            $back[i,j,A] \leftarrow \{k, B, C\}$
    return BUILD_TREE(*back*[1, LENGTH(*words*), *S*]), *table*[1, LENGTH(*words*), *S*]

Estimate the probabilities

- Using the treebank to count the statistics

$$P(\beta|A) = \frac{Count(A \rightarrow \beta)}{\sum_{\gamma} Count(A \rightarrow \gamma)} = \frac{Count(A \rightarrow \beta)}{Count(A)}$$

- We can also estimate the probabilities using a (non-probabilistic) parser
  - Parse the corpus, compute the statistics, and normalize the probabilities
  - Might need to use the inside-outside algorithm for ambiguous sentences (see SLP2,3)

A		β	P(β | NP)
NP	→	NP PP	0.092
NP	→	DT NN	0.087
NP	→	NN	0.047
NP	→	NNS	0.042
NP	→	DT JJ NN	0.035
NP	→	NNP	0.034
NP	→	NNP NNP	0.029
NP	→	JJ NNS	0.027
NP	→	QP -NONE-	0.018
NP	→	NP SBAR	0.017
NP	→	NP PP-LOC	0.017
NP	→	JJ NN	0.015
NP	→	DT NNS	0.014
NP	→	CD	0.014
NP	→	NN NNS	0.013
NP	→	DT NN NN	0.013
NP	→	NP CC NP	0.013

I	shot	an	elephant	in	my	pajamas
PRP:0.04 [0,1]						
	VBD:0.04 [1,2]					
		DT:0.05 [2,3]				
			NN:0.03 [3,4]			
				IN:0.10 [4,5]		
					PRP$: 0.12 [5,6]	
						NNS:0.01 [6,7]

Probaiblity of a terminal (word) given its tag

$$P(A \rightarrow \beta)$$

I	shot	an	elephant	in	my	pajamas
PRP:0.04 [0,1]	∅	∅				
	VBD:0.04 [1,2]	∅				
		DT:0.05 [2,3]	NP: 0.00015 [2,4]			
			NN:0.03 [3,4]			
				IN:0.10 [4,5]		
					PRP$:0.12 [5,6]	
						NNS:0.01 [6,7]

$$table(2,4,NP) = P(\text{NP} \rightarrow \text{DT NN}) \times table(2,3,DT) \times table(3,4,NN)$$

I	shot	an	elephant	in	my	pajamas
PRP:0.04 [0,1]	∅	∅				
	VBD:0.04 [1,2]	∅	VP: 0.0000006 [1,4]			
		DT:0.05 [2,3]	NP: 0.00015 [2,4]			
			NN:0.03 [3,4]			
				IN:0.10 [4,5]		
					PRP$:0.12 [5,6]	
						NNS:0.01 [6,7]

We just calculated this value and can use it now

$$table(1,4,VP) = P(\text{VP} \rightarrow \text{VBD NP}) \times table(1,2,VBD) \times table(2,4,NP)$$

I	shot	an	elephant	in	my	pajamas
PRP: -3.21 [0,1]	∅	∅	S: -19.2 [0,4]			
	VBD: -3.21 [1,2]	∅	VP: -14.3 [1,4]			
		DT: -3.0 [2,3]	NP: -8.8 [2,4]			
			NN: -3.5 [3,4]			
				IN: -2.3 [4,5]		
					PRP$: -2.12 [5,6]	
						NNS: -4.6 [6,7]

Note these values are getting very small! Better to add in log space

I	shot	an	elephant	in	my	pajamas
PRP: -3.21 [0,1]	∅	∅	S: -19.2 [0,4]	∅	∅	
	VBD: -3.21 [1,2]	∅	VP: -14.3 [1,4]	∅	∅	VP_1, VP_2 [1,7]
		DT: -3.0 [2,3]	NP: -8.8 [2,4]	∅	∅	NP: -24.7 [2,7]
			NN: -3.5 [3,4]	∅	∅	NP: -19.4 [3,7]
				IN: -2.3 [4,5]	∅	PP: -13.6 [4,7]
					PRP$: -2.12 [5,6]	NP: -9.0 [5,7]
						NNS: -4.6 [6,7]

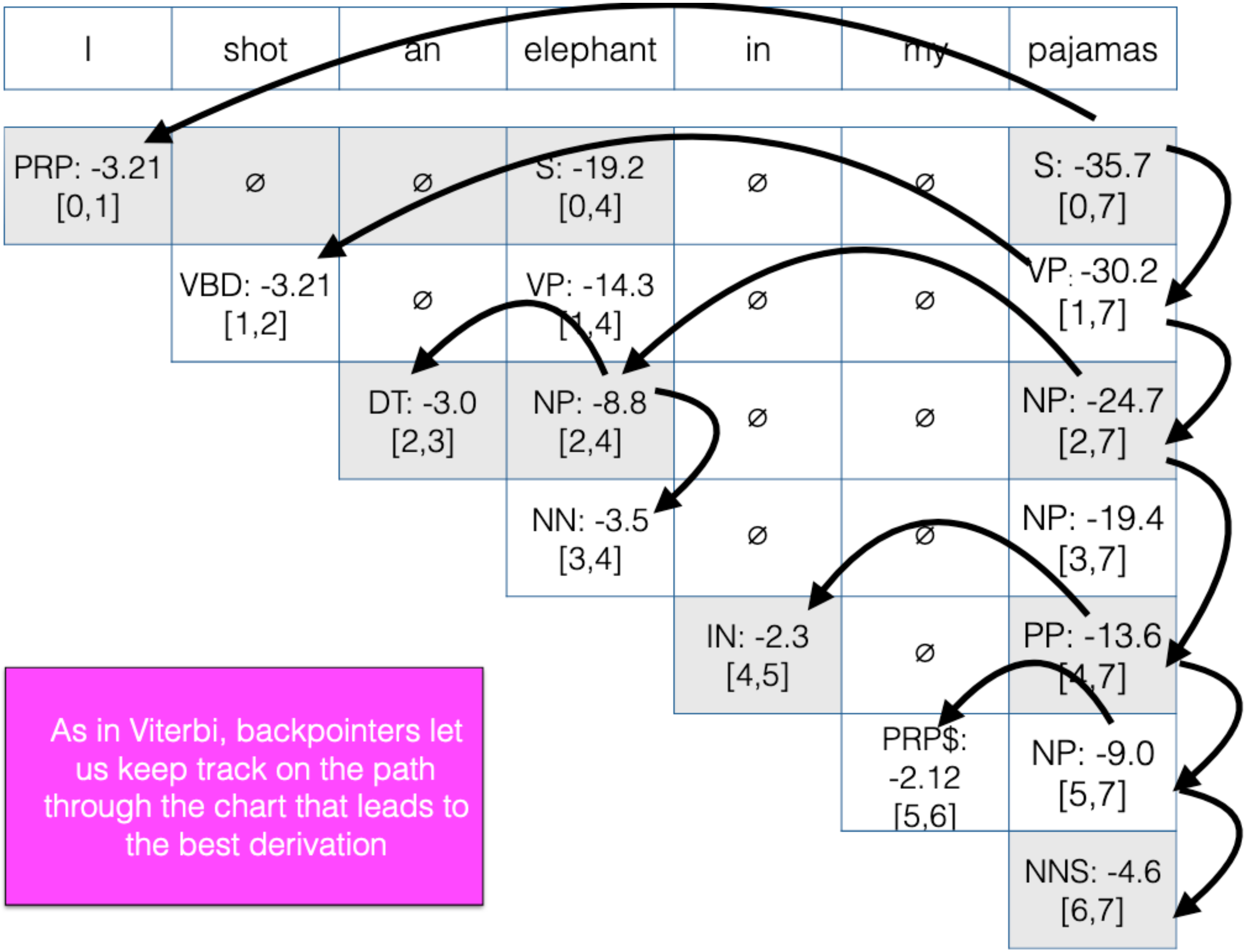
For any phrase type spanning [i,j], we only need to keep the max probability given the assumptions of a PCFG

I	shot	an	elephant	in	my	pajamas
PRP: -3.21 [0,1]	∅	∅	S: -19.2 [0,4]	∅	∅	
	VBD: -3.21 [1,2]	∅	VP: -14.3 [1,4]	∅	∅	VP: -30.2 [1,7]
		DT: -3.0 [2,3]	NP: -8.8 [2,4]	∅	∅	NP: -24.7 [2,7]
			NN: -3.5 [3,4]	∅	∅	NP: -19.4 [3,7]
				IN: -2.3 [4,5]	∅	PP: -13.6 [4,7]
					PRP$: -2.12 [5,6]	NP: -9.0 [5,7]
						NNS: -4.6 [6,7]

For any phrase type spanning [i,j], we only need to keep the max probability given the assumptions of a PCFG

I	shot	an	elephant	in	my	pajamas
PRP: -3.21 [0,1]	∅	∅	S: -19.2 [0,4]	∅	∅	S: -35.7 [0,7]
	VBD: -3.21 [1,2]	∅	VP: -14.3 [1,4]	∅	∅	VP: -30.2 [1,7]
		DT: -3.0 [2,3]	NP: -8.8 [2,4]	∅	∅	NP: -24.7 [2,7]
			NN: -3.5 [3,4]	∅	∅	NP: -19.4 [3,7]
				IN: -2.3 [4,5]	∅	PP: -13.6 [4,7]
					PRP$: -2.12 [5,6]	NP: -9.0 [5,7]
						NNS: -4.6 [6,7]

For any phrase type spanning [i,j], we only need to keep the max probability given the assumptions of a PCFG



As in Viterbi, backpointers let us keep track on the path through the chart that leads to the best derivation

Problems with PCFG

- $P(T) = \prod_{i=1}^{n} P(\beta|A)$
- Strong independence assumptions:
  - Each production (e.g., NP → DT NN) is independent of the rest of tree.
  - In real use, productions are strongly dependent on their place in the tree.

	NP → PRP Pronoun	NP → DT NN Non-Pronoun
Subject	91%	9%
Object	34%	66%

Problems with PCFG

- $P(T) = \prod_{i=1}^{n} P(\beta|A)$
- Strong independence assumptions:

	NP → PRP Pronoun	NP → DT NN Non-Pronoun
Subject	91%	9%
Object	34%	66%

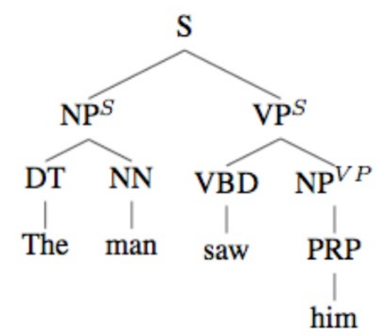
  - With maximum likelihood estimator on Swithboard dataset:

$$P(NP \rightarrow DT\ NN) = 0.28$$
$$P(NP \rightarrow PRP) = 0.25$$

Splitting non-terminals/ Parent annotation

- Rather than having a single rule for each non-terminal P(NP → DT NN), we can condition on some context (Johnson 1998)
  - $P_{subject}$(NP → DT NN)
  - P_{object}(NP → DT NN)

- More generally, we can encode context by annotating each node in a tree with its parent (parent annotation)
  - This lets us to learn different probabilities for:
    - NP^{S} (subject)
    - NP_{VP} (object)



- This dramatically increases the size of the grammar → less training data for each production (data sparsity)
- Modern approaches search for best splits that maximize the training data likelihood (Petrov et al 2006)

Problems with PCFGs

- Lack of lexical dependency: Lexical information in a PCFG has little influence on the overall parse structure
  - The identity of the verbs, nouns, and prepositions might be crucial to disambiguate the parses.

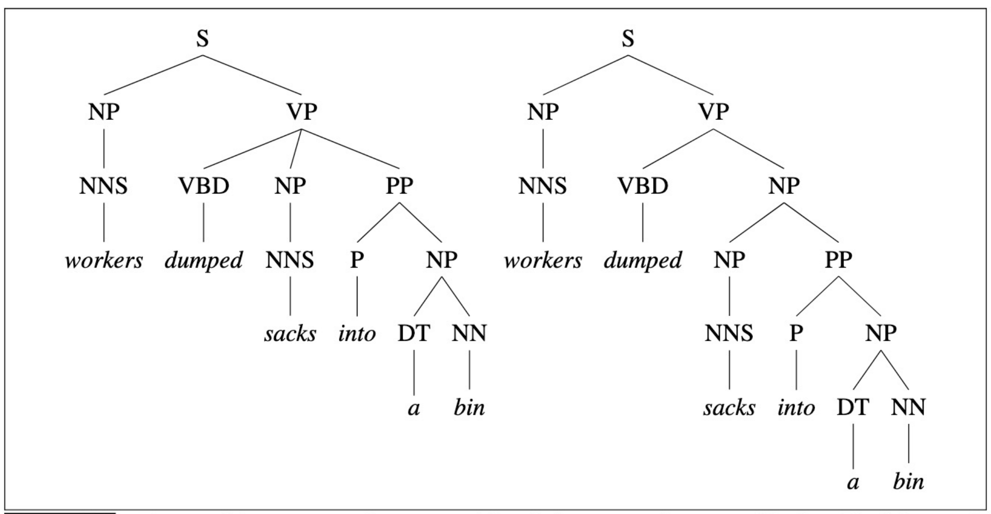


Figure 14.5 Two possible parse trees for a **prepositional phrase attachment ambiguity**. The left parse is the sensible one, in which "into a bin" describes the resulting location of the sacks. In the right incorrect parse, the sacks to be dumped are the ones which are already "into a bin", whatever that might mean.

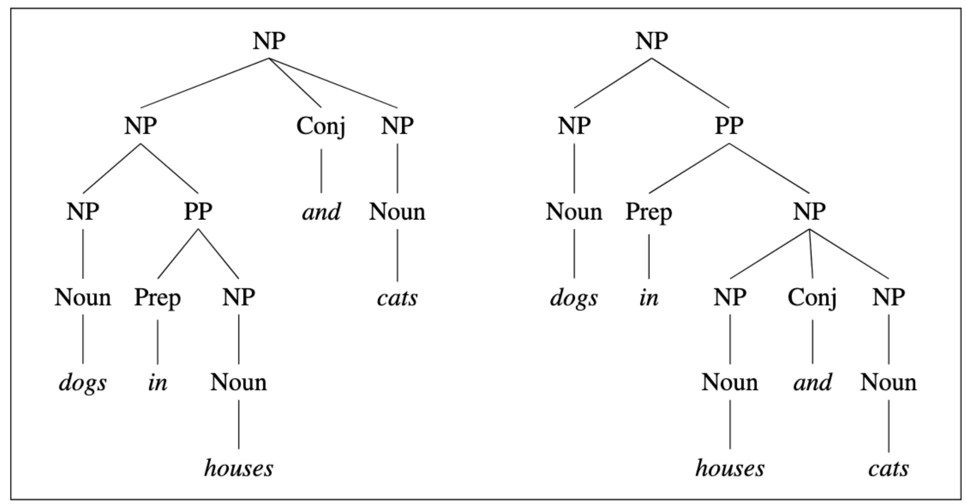
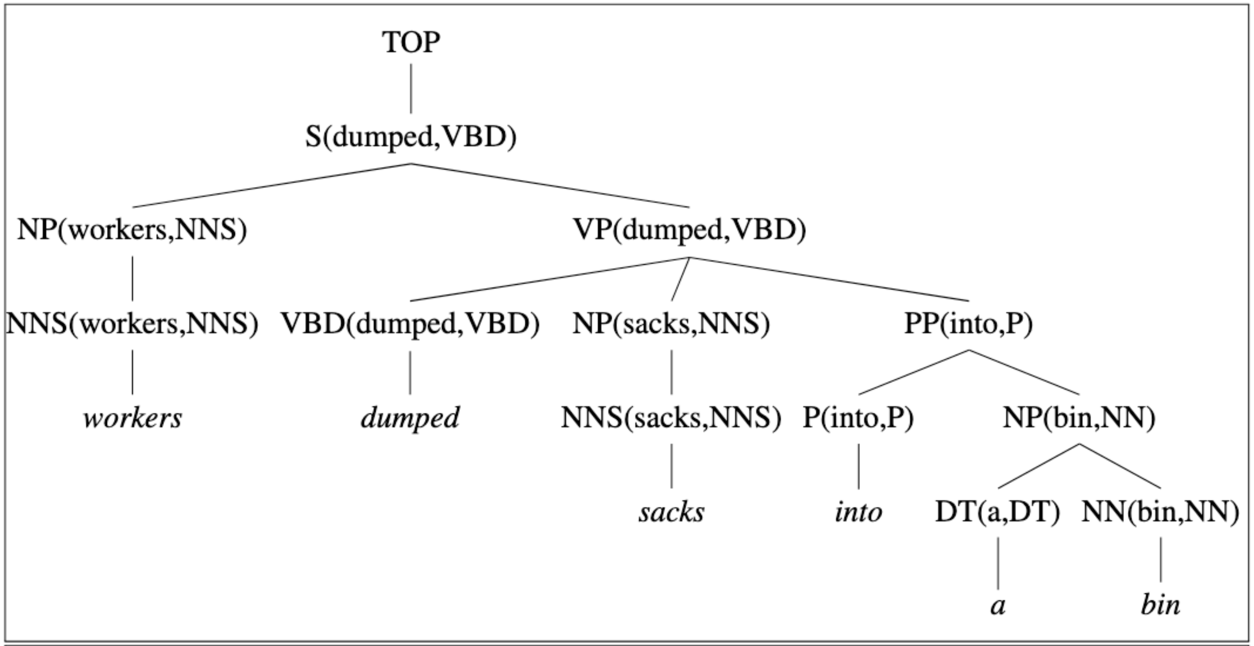


Figure 14.7 An instance of coordination ambiguity. Although the left structure is intuitively the correct one, a PCFG will assign them identical probabilities since both structures use exactly the same set of rules. After Collins (1999).

Lexicalized PCFG

- Annotate each node with its head + POS tag of head



Internal Rules					**Lexical Rules**		
TOP	→	S(dumped,VBD)			NNS(workers,NNS)	→	workers
S(dumped,VBD)	→	NP(workers,NNS)	VP(dumped,VBD)		VBD(dumped,VBD)	→	dumped
NP(workers,NNS)	→	NNS(workers,NNS)			NNS(sacks,NNS)	→	sacks
VP(dumped,VBD)	→	VBD(dumped, VBD)	NP(sacks,NNS)	PP(into,P)	P(into,P)	→	into
PP(into,P)	→	P(into,P)	NP(bin,NN)		DT(a,DT)	→	a
NP(bin,NN)	→	DT(a,DT)	NN(bin,NN)		NN(bin,NN)	→	bin

Figure 14.10 A lexicalized tree, including head tags, for a WSJ sentence, adapted from Collins (1999). Below we show the PCFG rules needed for this parse tree, internal rules on the left, and lexical rules on the right.

Lexicalized PCFG

- Annotate each node with its head + POS tag of head
- We can't estimate probabilities for such fine-grained productions well:

$$\frac{Count(VP(dumped,VBD) \rightarrow VBD(dumped,\ VBD)\ NP(sacks,NNS)\ PP(into,P))}{Count(VP(dumped,VBD))}$$

- Different models make different independent assumptions to make this quantity tractable (Collins 1999, Charniak 1997)