CS 4248
Natural Language Processing

Professor NG Hwee Tou

Department of Computer Science
School of Computing
National University of Singapore
nght@comp.nus.edu.sg

Syntactic Parsing

- Syntactic parsing: The task of recognizing a sentence and assigning a syntactic structure to it
- Useful for information extraction, semantic analysis, etc.

A Sample CFG

S $\rightarrow$ NP VP

S $\rightarrow$ Aux NP VP

S $\rightarrow$ VP

NP $\rightarrow$ Pronoun

NP $\rightarrow$ Proper-Noun

NP $\rightarrow$ Det Nominal

Nominal $\rightarrow$ Noun

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 $\rightarrow$ Prep NP

A Sample Lexicon

Det $\rightarrow$ that | this | a

Noun $\rightarrow$ book | flight | meal | money

Verb $\rightarrow$ book | include | prefer

Pronoun $\rightarrow$ I | she | me

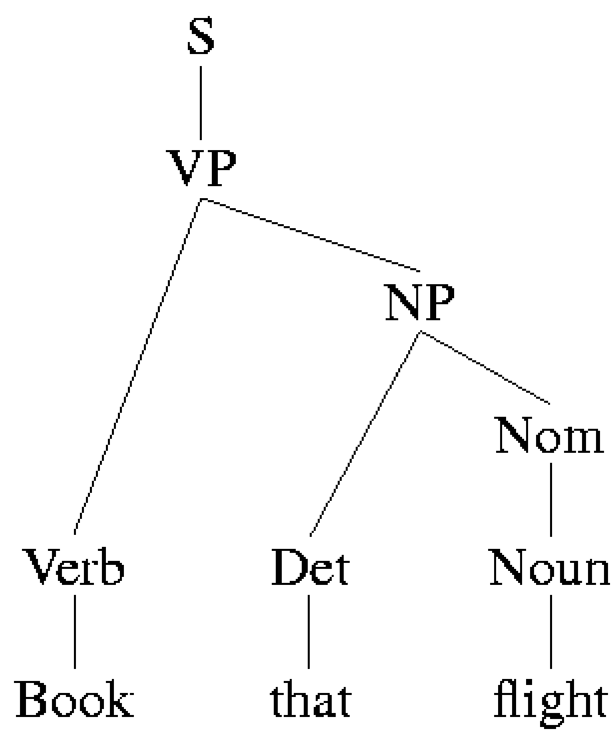
Proper-Noun $\rightarrow$ Houston | NWA

Aux $\rightarrow$ does

Prep $\rightarrow$ from | to | on | near | through

Parse Tree

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Prep NP

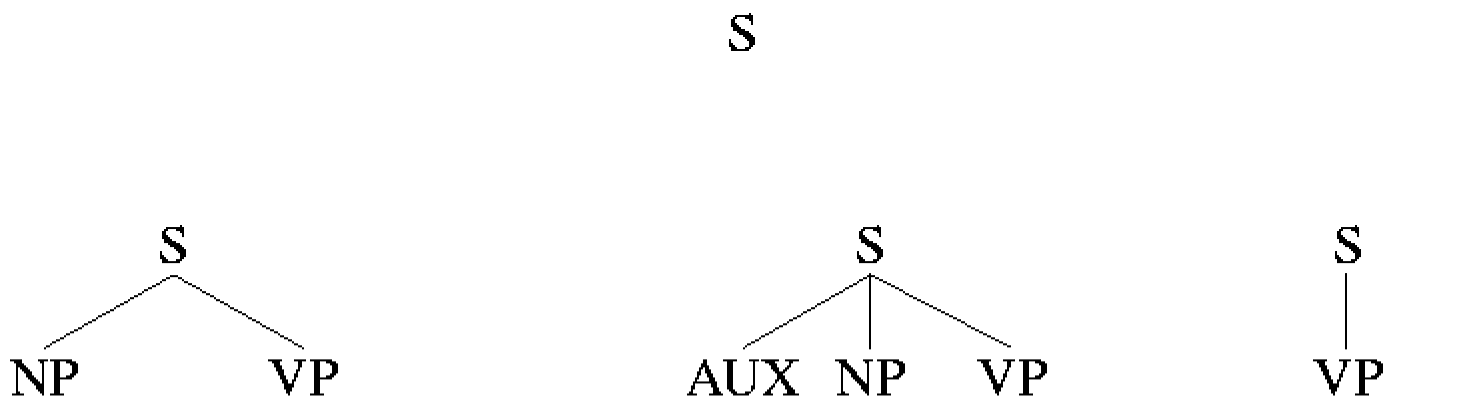


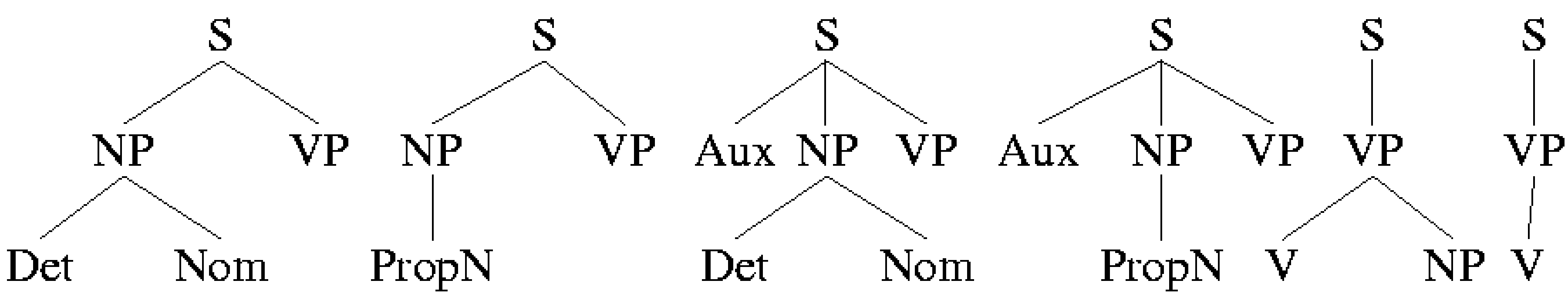
Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

Top-Down Parsing

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Prep NP

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

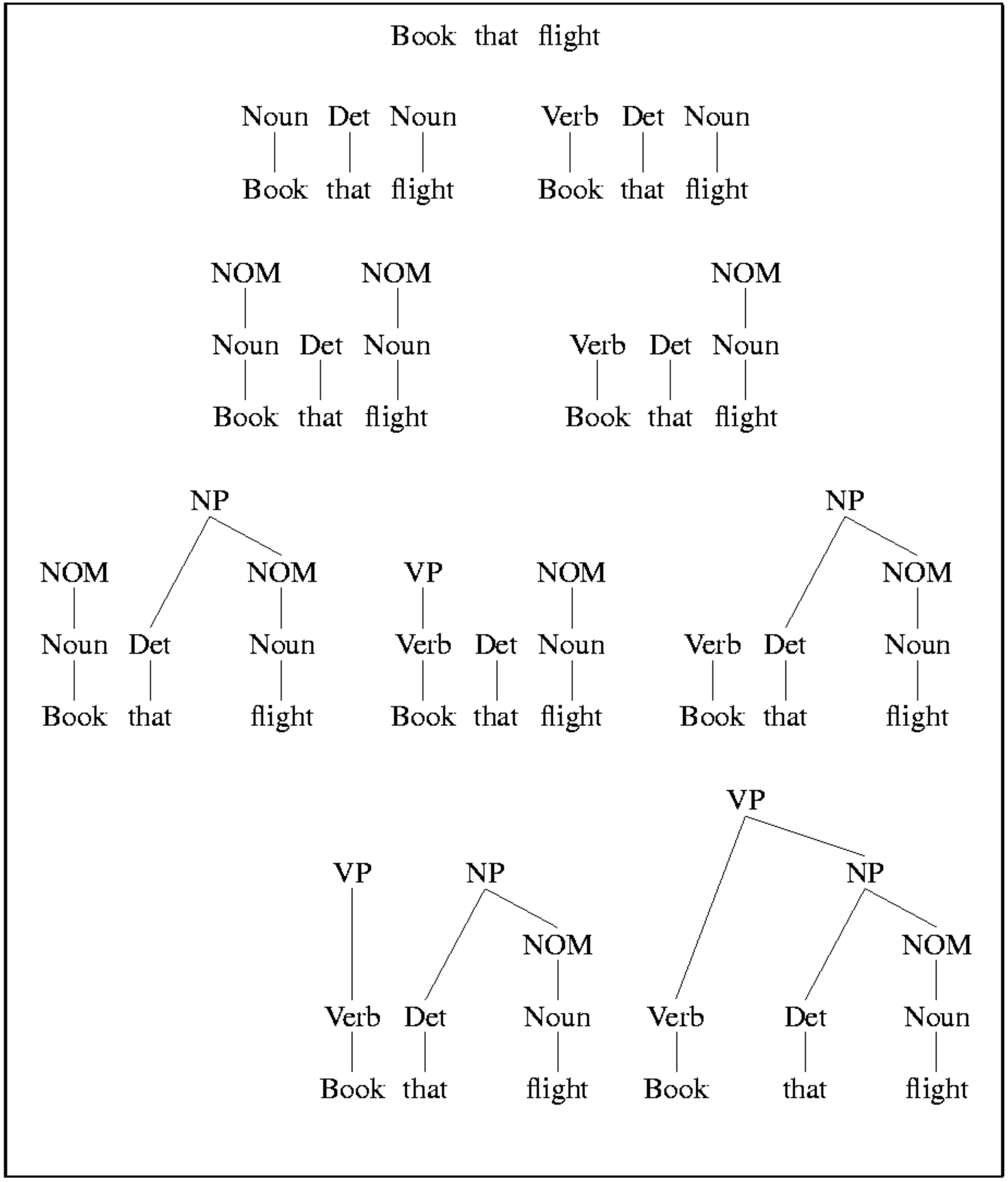




Bottom-Up Parsing

S → NP VP
S → Aux NP VP
S → VP
NP → Pronoun
NP → Proper-Noun
NP → Det Nominal
Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP
VP → Verb
VP → Verb NP
VP → Verb NP PP
VP → Verb PP
VP → VP PP
PP → Prep NP

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through



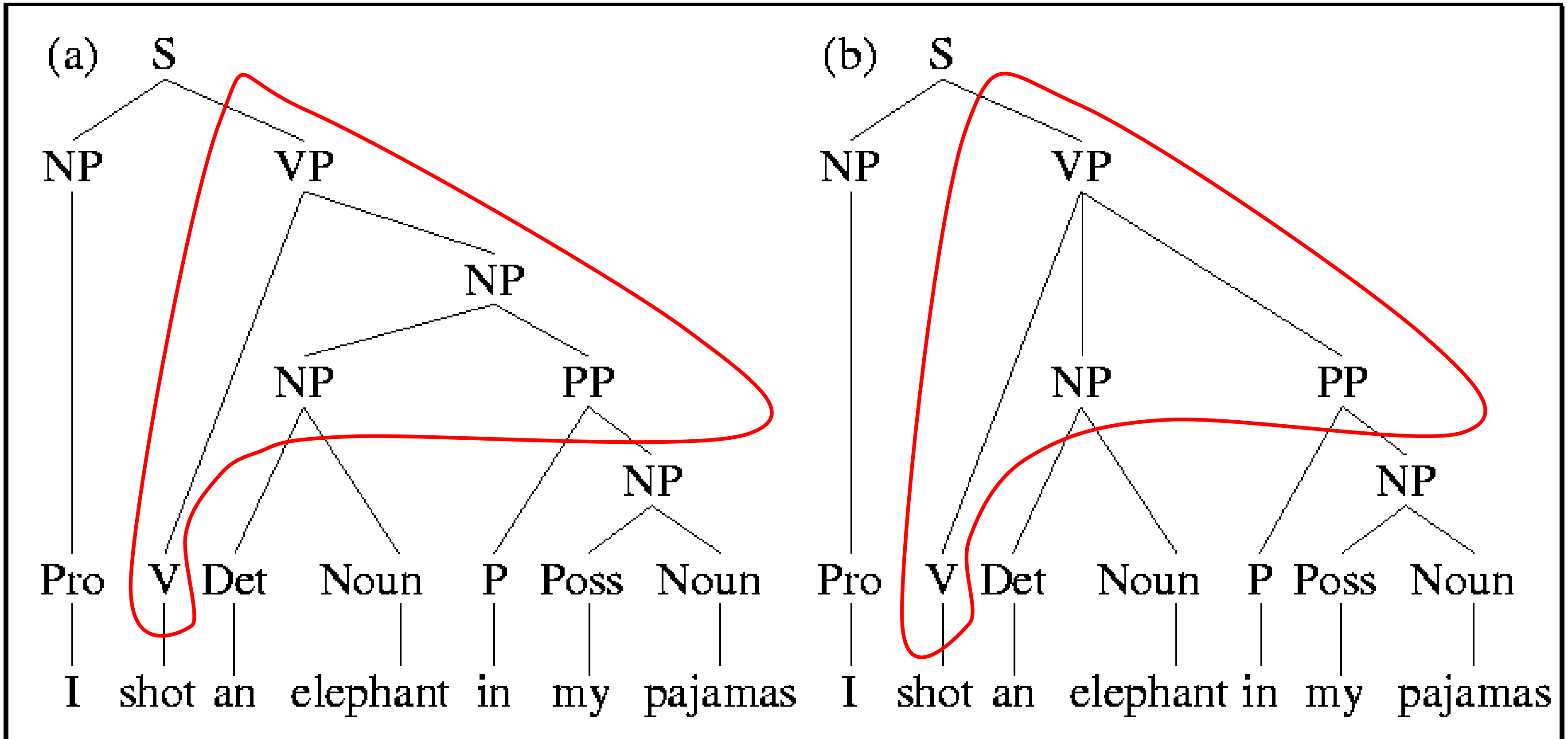
Top-Down vs. Bottom-Up Parsing

- Top-down parsing
  - goal-directed search
  - Never wastes time exploring trees that cannot result in an S
- Bottom-up parsing
  - data-directed search
  - Never suggests trees that are not grounded in the actual input sentence
- Need to incorporate features of both

Structural Ambiguity

- Attachment ambiguity
- Noun-phrase bracketing ambiguity
- Coordination ambiguity

Attachment Ambiguity

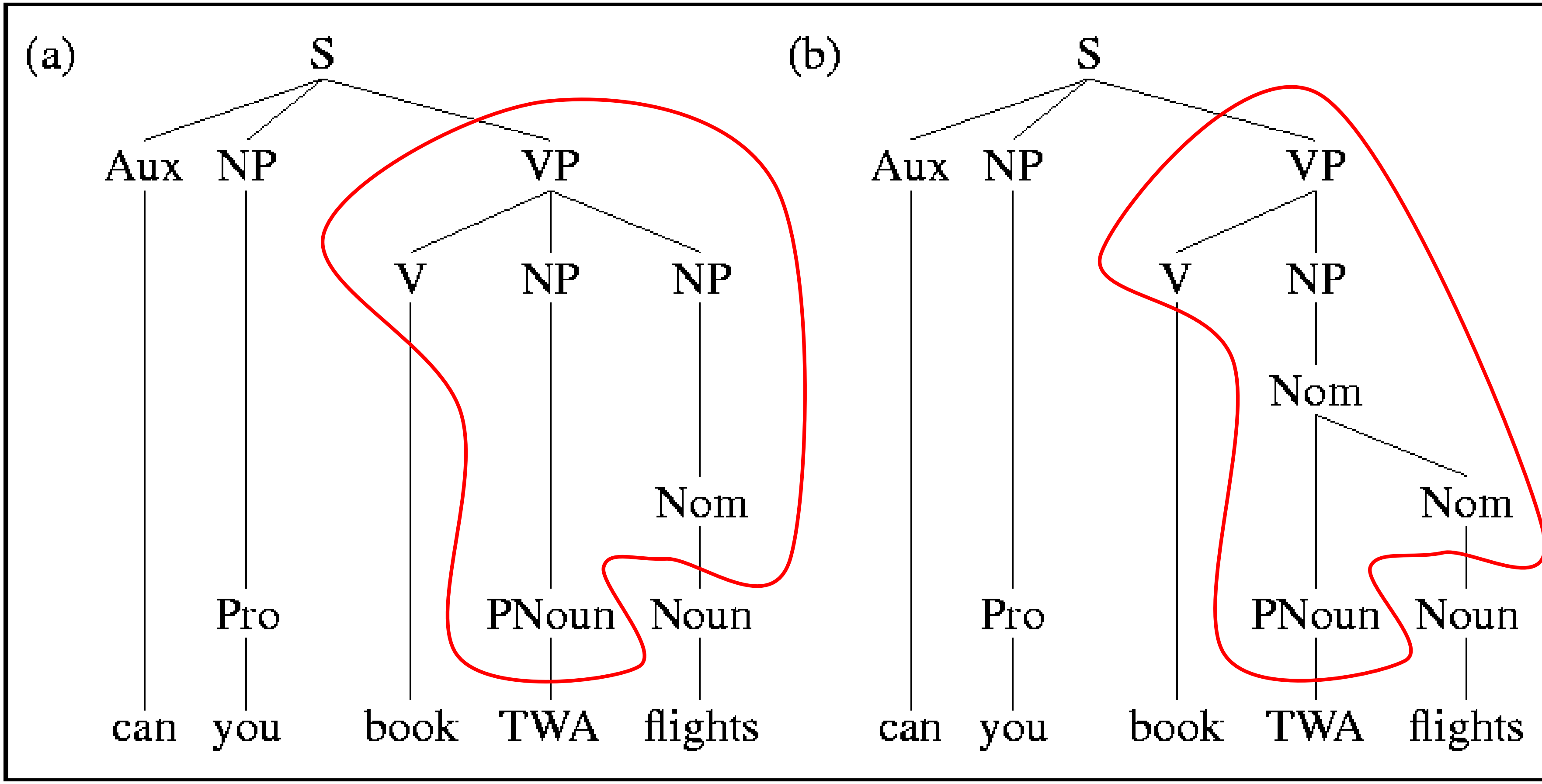


Attachment Ambiguity

We [saw [[the Eiffel Tower] [flying into Paris]]]

We [[saw] [the Eiffel Tower] [flying into Paris]]

Noun-Phrase Bracketing Ambiguity



Coordination Ambiguity

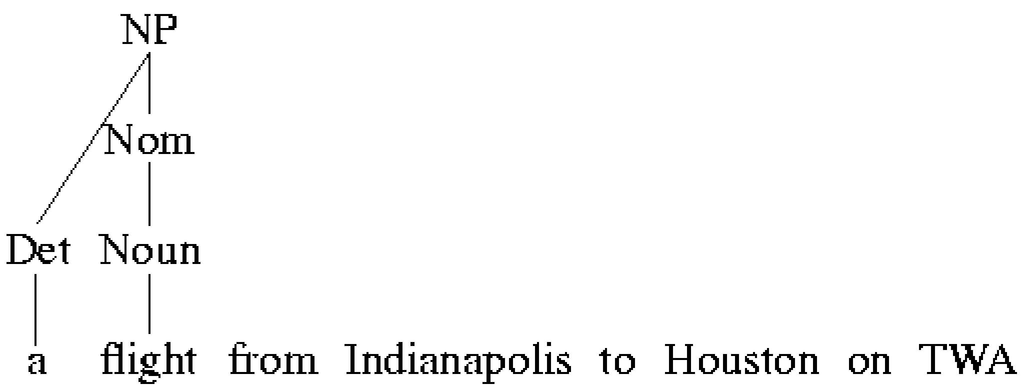
old [men and women]

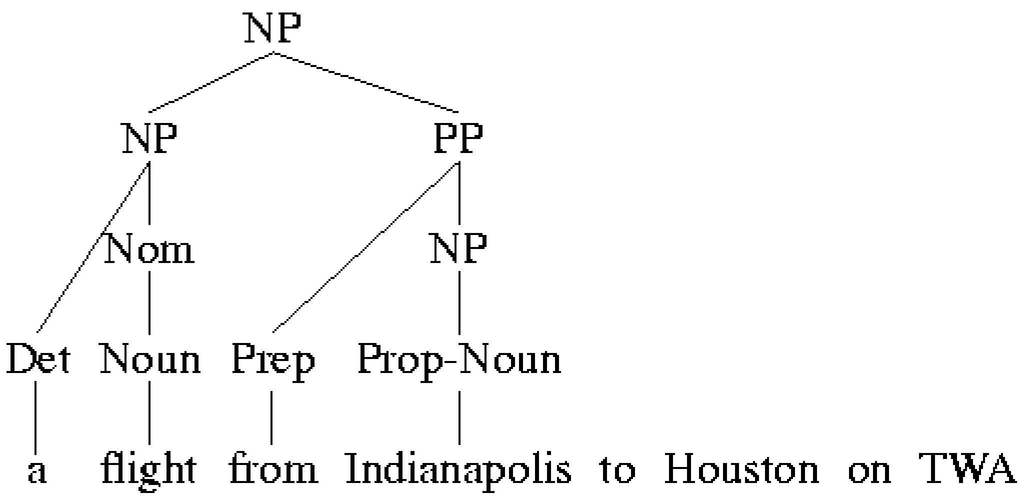
[old men] and [women]

Parsing as Search

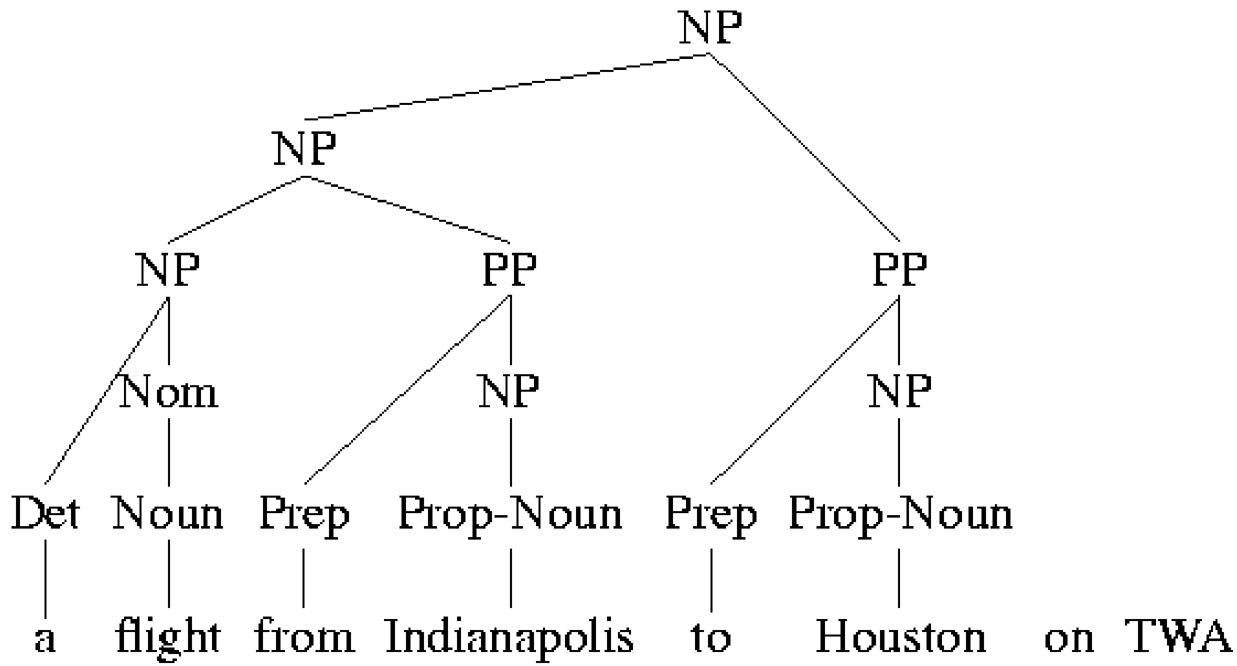
- Parsing involves searching the space of parse trees
- Agenda-based backtracking search leads to reduplication of work (repeated parsing of subtrees)

Repeated Parsing of Subtrees

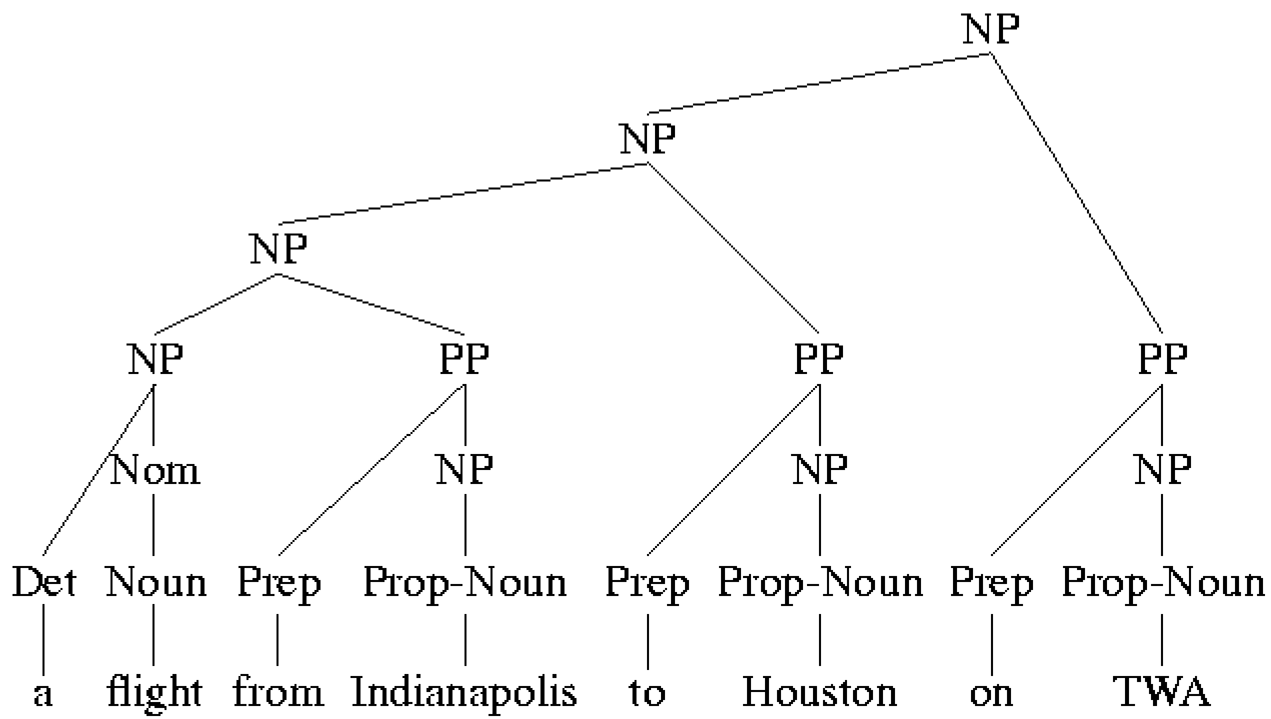




Repeated Parsing of Subtrees



Repeated Parsing of Subtrees



The CKY Algorithm

- CKY (Cocke-Kasami-Younger)
- Bottom-up dynamic programming algorithm
- Requires the CFG to be in Chomsky Normal Form (CNF)
  - The grammar is ε-free
  - Each production of the grammar is either of the form $A \rightarrow B\ C$ or $A \rightarrow a$ (i.e., either 2 non-terminal symbols or 1 terminal symbol on RHS)
- Any CFG can be converted into a weakly equivalent CFG in Chomsky Normal Form

A Sample CFG in CNF

$S \rightarrow$ NP VP
$S \rightarrow$ X1 VP
$X1 \rightarrow$ Aux NP
$S \rightarrow$ book | include | prefer
$S \rightarrow$ Verb NP
$S \rightarrow$ X2 PP
$S \rightarrow$ Verb PP
$S \rightarrow$ VP PP
$NP \rightarrow$ I | she | me
$NP \rightarrow$ Houston | NWA
$NP \rightarrow$ Det Nominal

Nominal $\rightarrow$ book | flight | meal | money
Nominal $\rightarrow$ Nominal Noun
Nominal $\rightarrow$ Nominal PP
$VP \rightarrow$ book | include | prefer
$VP \rightarrow$ Verb NP
$VP \rightarrow$ X2 PP
$X2 \rightarrow$ Verb NP
$VP \rightarrow$ Verb PP
$VP \rightarrow$ VP PP
$PP \rightarrow$ Prep NP

A Sample Lexicon

Det $\rightarrow$ that | this | a

Noun $\rightarrow$ book | flight | meal | money

Verb $\rightarrow$ book | include | prefer

Pronoun $\rightarrow$ I | she | me

Proper-Noun $\rightarrow$ Houston | NWA

Aux $\rightarrow$ does

Prep $\rightarrow$ from | to | on | near | through

The CKY Algorithm

0 book 1	a 2	flight 3	through 4	Houston 5
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	S, VP [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, PN [4,5]

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 PP
S → Verb PP
S → VP PP
NP → I | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → X2 PP
X2 → Verb NP
VP → Verb PP
VP → VP PP
PP → Prep NP

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

The CKY Algorithm: Filling the Cell [1,5]

0 book 1	a 2	flight 3	through 4	Houston 5
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	S, VP [0,5]
	Det L [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal R [2,5]
			Prep [3,4]	PP [3,5]
				NP, PN [4,5]

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 PP
S → Verb PP
S → VP PP
NP → I | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → X2 PP
X2 → Verb NP
VP → Verb PP
VP → VP PP
PP → Prep NP

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

The CKY Algorithm: Filling the Cell [1,5]

0 book 1 a 2 flight 3 through 4 Houston 5

S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	S, VP [0,5]
	Det [1,2]	NP L [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP R [3,5]
				NP, PN [4,5]

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 PP
S → Verb PP
S → VP PP
NP → I | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → X2 PP
X2 → Verb NP
VP → Verb PP
VP → VP PP
PP → Prep NP

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

The CKY Algorithm: Filling the Cell [1,5]

0 book 1	a 2	flight 3	through 4	Houston 5
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	S, VP [0,5]
	Det [1,2]	NP [1,3]	L [1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, PN R [4,5]

S → NP VP
S → X1 VP
X1 → Aux NP
S → book | include | prefer
S → Verb NP
S → X2 PP
S → Verb PP
S → VP PP
NP → I | she | me
NP → Houston | NWA
NP → Det Nominal
Nominal → book | flight | meal | money
Nominal → Nominal Noun
Nominal → Nominal PP
VP → book | include | prefer
VP → Verb NP
VP → X2 PP
X2 → Verb NP
VP → Verb PP
VP → VP PP
PP → Prep NP

Det → that | this | a
Noun → book | flight | meal | money
Verb → book | include | prefer
Pronoun → I | she | me
Proper-Noun → Houston | NWA
Aux → does
Prep → from | to | on | near | through

The CKY Algorithm

```
function CKY-Parse(words, grammar) returns table

for j ← from 1 to Length(words) do
    table[j – 1, j] ← { A | A → words[j]  ∈ grammar }
    for i ← from j – 2 downto 0 do
        for k ← i + 1 to j – 1 do
            table[i, j] ← table[i, j] ∪
                        { A | A → BC ∈ grammar, B ∈ table[i, k], C ∈ table[k, j] }
```

CKY Parsing

- To return all possible parses:
  - Augment each entry such that each non-terminal is paired with pointers to the entries from which it was derived
  - Permit multiple versions of the same non-terminal to be entered in an entry

Statistical Parsing

- Resolving structural ambiguity: choose the most probable parse

Probabilistic Context-Free Grammars

- PCFG
- $G = (N, \Sigma, P, S, D)$
  - A set of non-terminal symbols (or variables) N
  - A set of terminal symbols Σ ($N \cap \Sigma = \emptyset$)
  - A set of productions P, each of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (\Sigma \cup N)^*$
  - A designated start symbol $S \in N$
  - A function D that assigns a probability to each rule in P
- $P(A \rightarrow \alpha)$ or $P(A \rightarrow \alpha \mid A)$

Probabilistic Context-Free Grammars

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.05] | the [.80] | a [.15]	
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.35]	$Verb \rightarrow book$	[.30]
$NP \rightarrow Nom$	[.05]	$Verb \rightarrow include$	[.30]
$NP \rightarrow Pronoun$	[.40]	$Verb \rightarrow want$	[.40]
$Nom \rightarrow Noun$	[.75]	$Aux \rightarrow can$	[.40]
$Nom \rightarrow Noun\ Nom$	[.20]	$Aux \rightarrow does$	[.30]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[~~.40~~] **.60**
$VP \rightarrow Verb\ NP\ NP$	[.05]	$Pronoun \rightarrow you$ [.40] | I [.60]	

PCFG

$$P(T,S) = \prod_{n \in T} p(r(n))$$

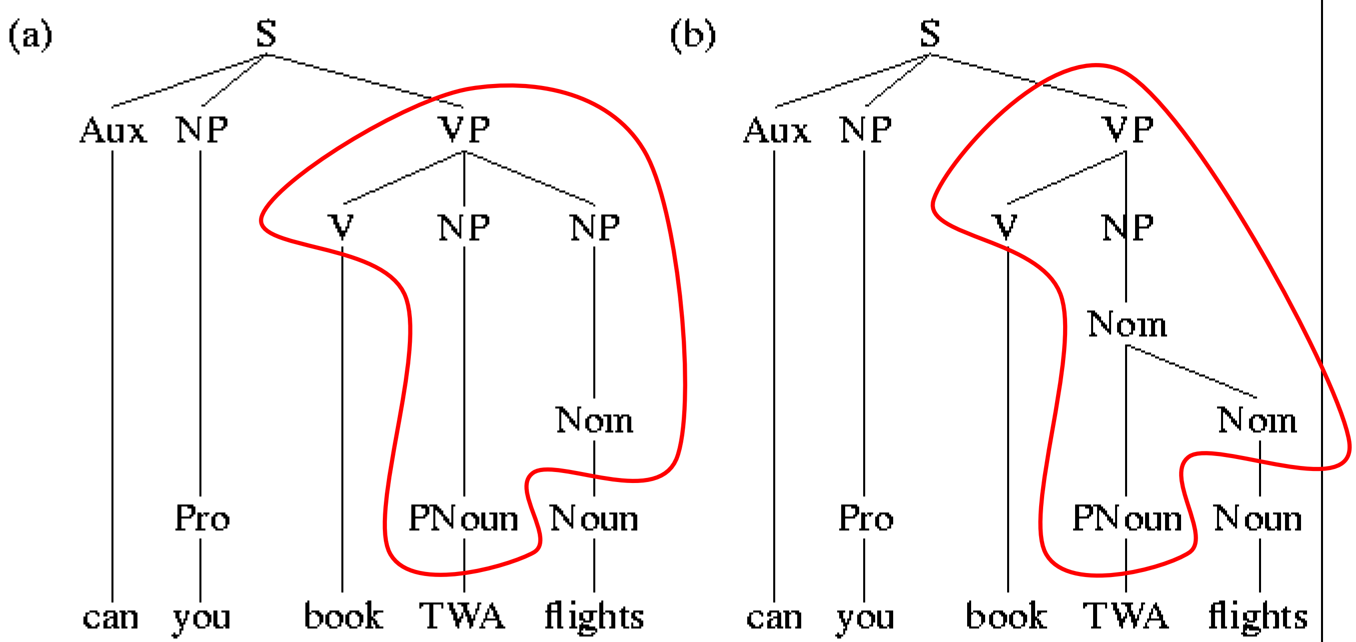
$$P(T_l) = .15 \times .40 \times .05 \times .05 \times .35 \times .75 \times .40 \times .40 \times .30 \times .40 \times .50 = 3.78 \times 10^{-7}$$

$$P(T_r) = .15 \times .40 \times .40 \times .05 \times .05 \times .75 \times .40 \times .40 \times .30 \times .40 \times .50 = 4.32 \times 10^{-7}$$

$$\hat{T}(S) = \underset{T\ s.t.S=yield(T)}{\arg\max}\ P(T \mid S) = \underset{T\ s.t.S=yield(T)}{\arg\max}\ \frac{P(T,S)}{P(S)}$$

$$= \underset{T\ s.t.S=yield(T)}{\arg\max}\ P(T,S) = \underset{T\ s.t.S=yield(T)}{\arg\max}\ P(T)$$

PCFG



Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP NP	.05
NP	→	Nom	.05
NP	→	PNoun	.35
Nom	→	Noun	.75
Aux	→	Can	.40
~~NP~~	~~→~~	~~Pro~~	~~.40~~
Pro	→	you	.40
Verb	→	book	.30
PNoun	→	TWA	.40
Noun	→	flights	.50

Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP	.40
NP	→	Nom	.05
Nom	→	PNoun Nom	.05
Nom	→	Noun	.75
Aux	→	Can	.40
~~NP~~	~~→~~	~~Pro~~	~~.40~~
Pro	→	you	.40
Verb	→	book	.30
Pnoun	→	TWA	.40
Noun	→	flights	.50

Probabilistic CKY Parsing

- Probabilistic CKY (Cocke-Kasami-Younger) algorithm for parsing PCFG
- Bottom-up dynamic programming algorithm
- Assume PCFG is in Chomsky Normal Form (production is either $A \rightarrow B\ C$ or $A \rightarrow a$)
- table[i, j, A] stores the max probability for a constituent A spanning word positions i to j
- Probability of the most probably parse is table[0, n, S]

The Probabilistic CKY Algorithm

0 book 1	a 2	flight 3	through 4	Houston 5
S, VP, Verb, Nominal, Noun [0,1]	[0,2]	S, VP, X2 [0,3]	[0,4]	S, VP [0,5]
	Det [1,2]	NP [1,3]	[1,4]	NP [1,5]
		Nominal, Noun [2,3]	[2,4]	Nominal [2,5]
			Prep [3,4]	PP [3,5]
				NP, PN [4,5]

The Probabilistic CKY Algorithm

```
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[0, Length(words), S]),  table[0, Length(words), S]
```

Learning PCFG Rule Probabilities

- Estimate from parsed sentences (treebank)

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{Count}(\alpha \to \beta)}{\sum_{\gamma} \text{Count}(\alpha \to \gamma)} = \frac{\text{Count}(\alpha \to \beta)}{\text{Count}(\alpha)}$$

Problems with PCFGs

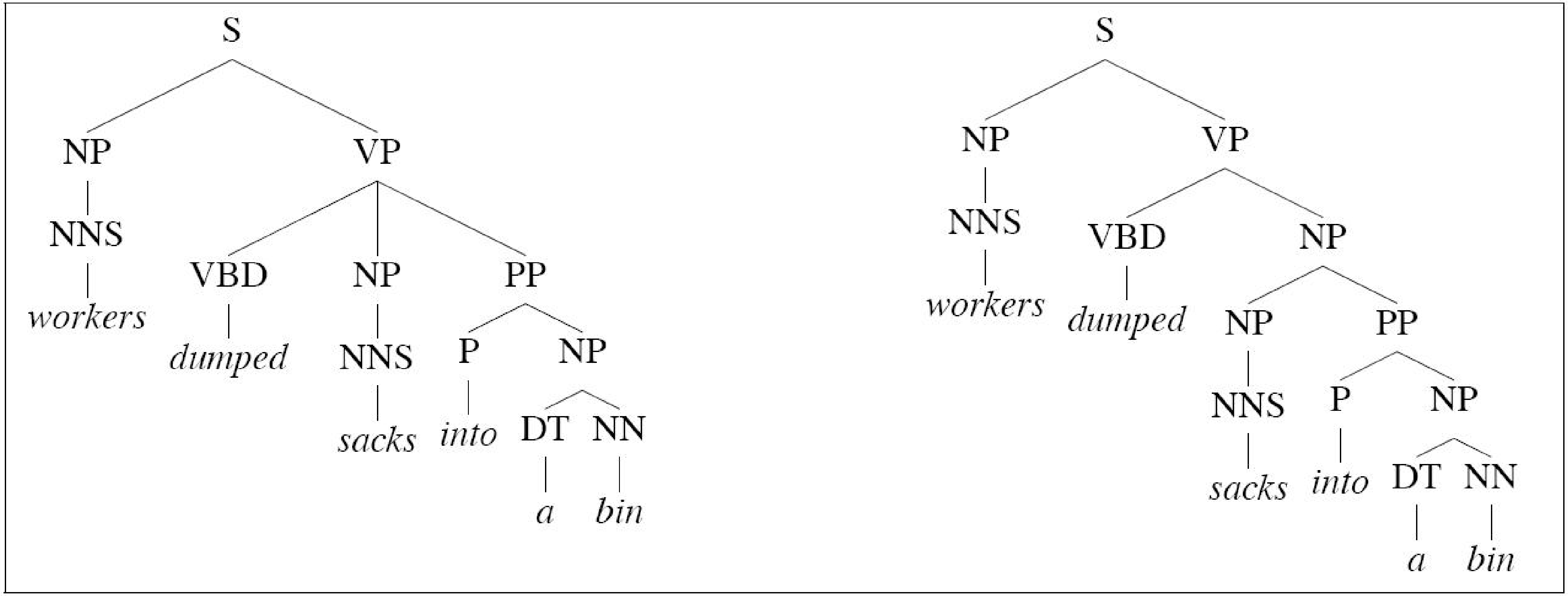
- Poor independence assumptions
- Lack of sensitivity to words

Poor Independence Assumptions

- The expansion of a non-terminal actually depends on its context
- Example:
  - An NP can be a pronoun or non-pronoun (e.g., proper noun or determiner noun sequence)
    - NP $\rightarrow$ PRP
    - NP $\rightarrow$ DT NN
  - Subject NP: 91% pronoun, 9% non-pronoun
  - Object NP: 34% pronoun, 66% non-pronoun

Lack of Sensitivity to Words

- Attachment ambiguity

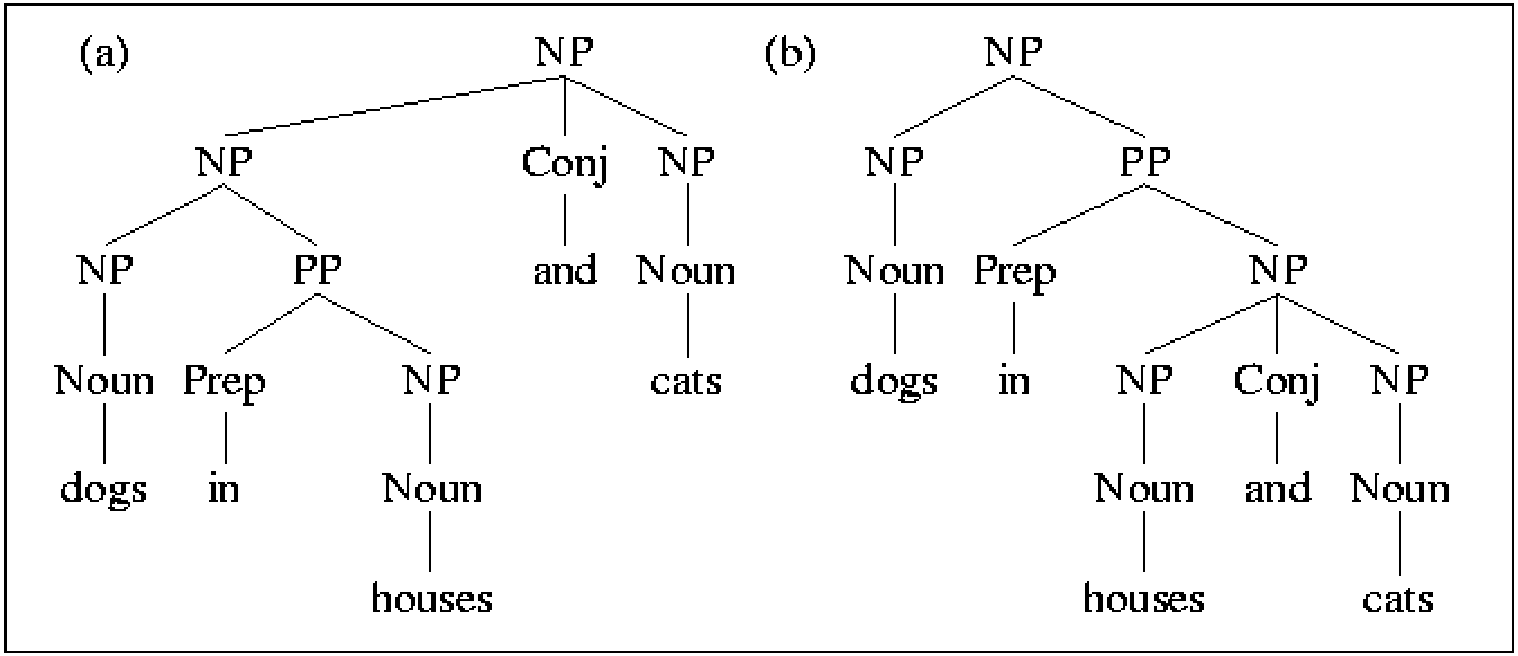


Lack of Sensitivity to Words

- Attachment ambiguity
  - VP attachment: Workers dumped sacks <u>into a bin</u>
  - NP attachment: Workers sold sacks <u>of a chemical</u>
  - Knowing the particular words (*dumped, sacks, into, sold, sacks, of*) helps in disambiguation

Lack of Sensitivity to Words

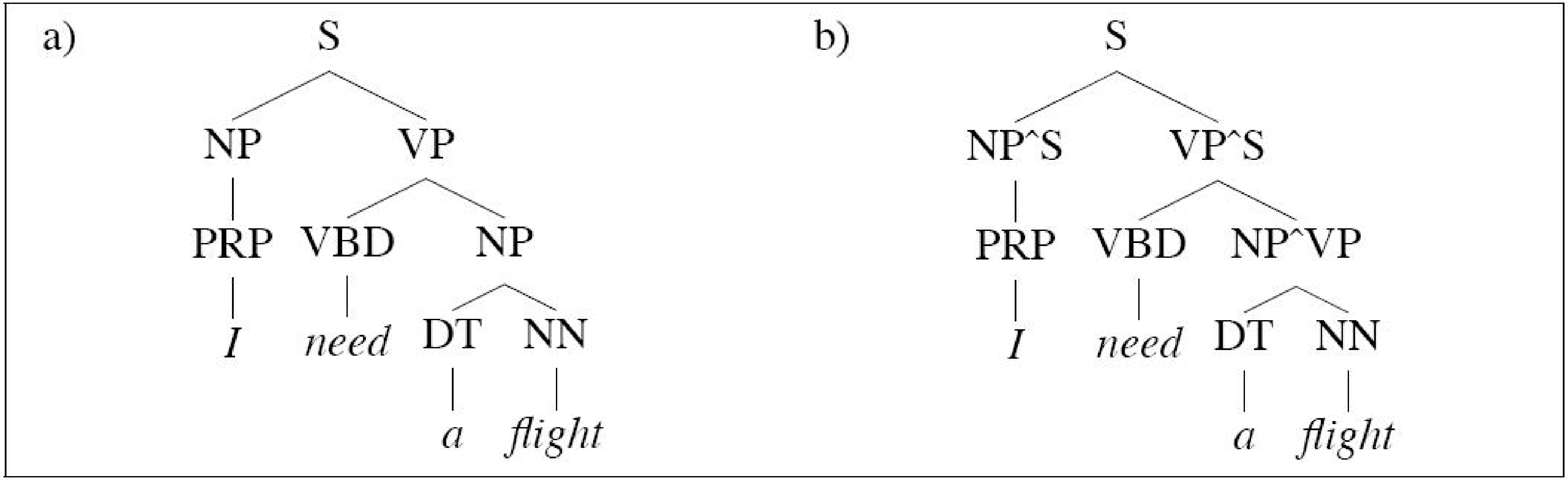
- Coordination ambiguity



Same set of rules used and hence the same probability

Splitting Non-terminals

- Parent annotation



Probabilistic Lexicalized CFGs

- Each non-terminal in a parse tree is annotated with a word (lexical head)
- Identify one RHS constituent of a PCFG rule as the head child of that rule
- Lexical head of the parent (LHS of a PCFG rule) is the lexical head of the head child

Probabilistic Lexicalized CFGs

Head child (underlined):

$S \rightarrow NP\ \underline{VP}$

$NP \rightarrow \underline{NNS}$

$VP \rightarrow \underline{VBD}\ NP\ PP$

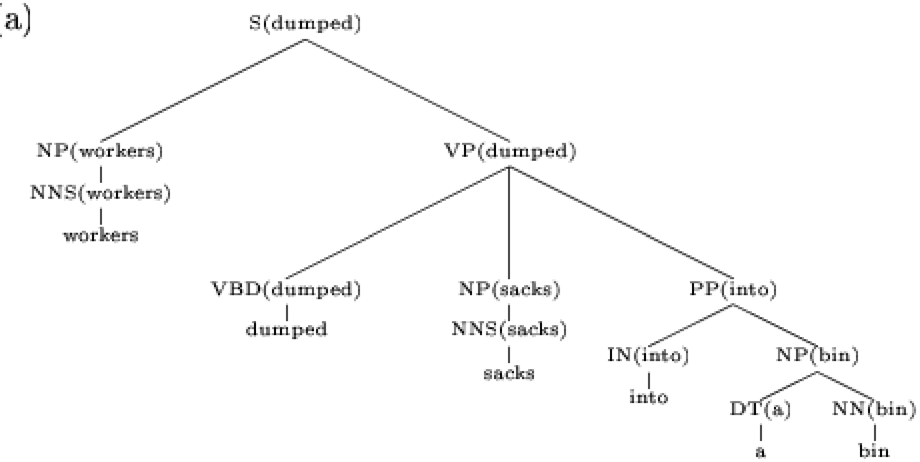
$PP \rightarrow \underline{IN}\ NP$

$NP \rightarrow DT\ \underline{NN}$

$NP \rightarrow \underline{NP}\ PP$

Probabilistic Lexicalized CFGs

(a)



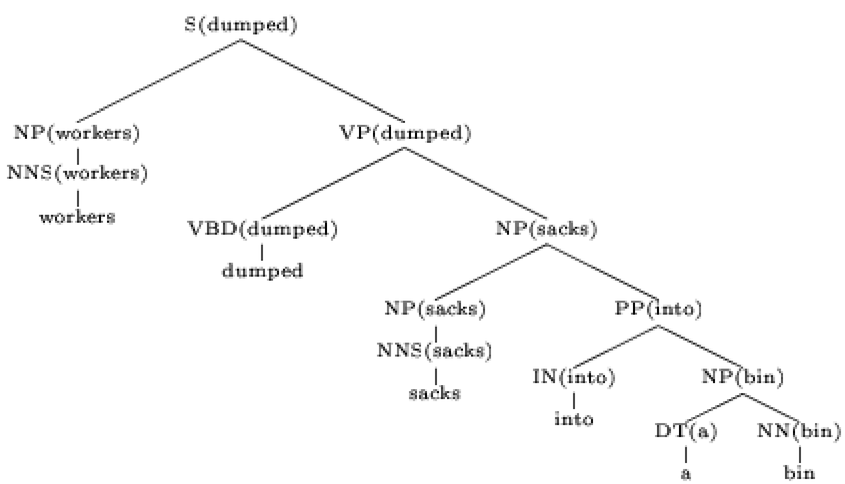
(b)

Dependencies
workers → dumped
dumped → START
sacks → dumped
into → dumped
a → bin
bin → into

modifier → head

Probabilistic Lexicalized CFGs

a')



b')

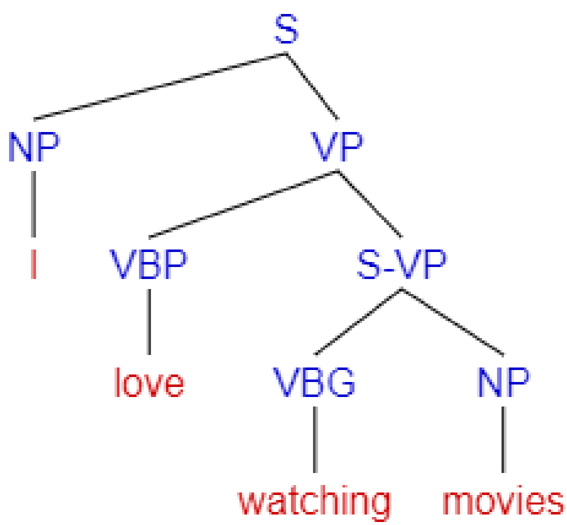
Dependencies
workers $\rightarrow$ dumped
dumped $\rightarrow$ START
sacks $\rightarrow$ dumped
into $\rightarrow$ sacks
a $\rightarrow$ bin
bin $\rightarrow$ into

modifier $\rightarrow$ head

Neural Constituency Parsing

- Materials from "A minimal span-based neural constituency parser", Mitchell Stern, Jacob Andreas, Dan Klein, ACL 2017

Parsing as Span Classification



0 1 2 3 4

I	love	watching	movies
NP [0,1]	[0,2]	[0,3]	S [0,4]
	VBP [1,2]	[1,3]	VP [1,4]
		VBG [2,3]	S-VP [2,4]
			NP [3,4]

Parse Tree Scoring Function

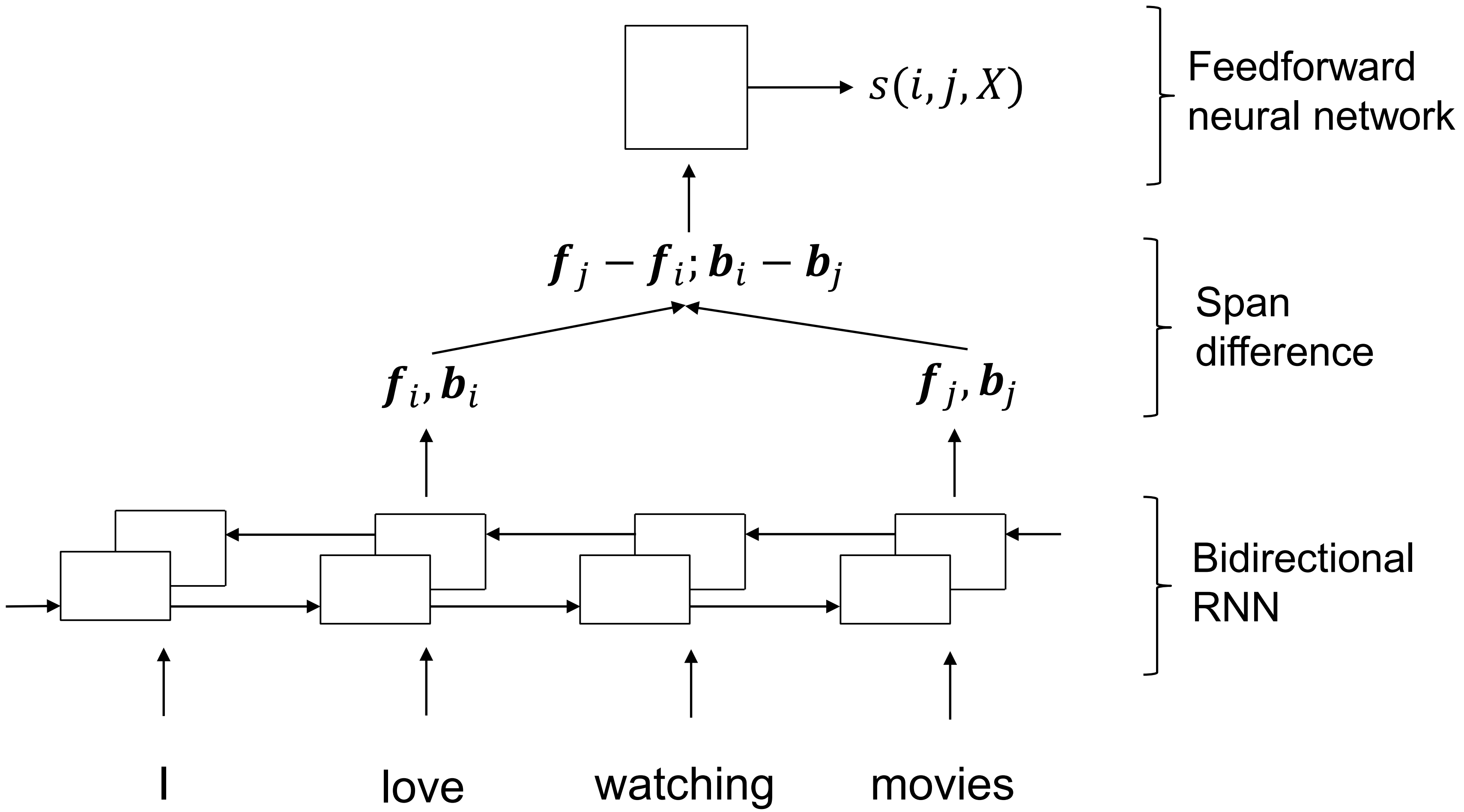
$$s_{\text{tree}}(T) = \sum_{(i,j,l)\epsilon T} s(i,j,l)$$

$$= s(0,1,NP)$$
$$+ s(1,2,VBP)$$
$$+ s(2,3,VBG)$$
$$+ s(3,4,NP)$$
$$+ s(2,4,S{-}VP)$$
$$+ s(1,4,VP)$$
$$+ s(0,4,S)$$

0 1 2 3 4

I	love	watching	movies
NP [0,1]	[0,2]	[0,3]	S [0,4]
	VBP [1,2]	[1,3]	VP [1,4]
		VBG [2,3]	S-VP [2,4]
			NP [3,4]

NN Implementation of Scoring Function



Dynamic Programming Algorithm

$$s_{\text{best}}(i,j) = \max_{l}[s(i,j,l)] \quad \text{if } j - i = 1$$

$$s_{\text{best}}(i,j) = \max_{l}[s(i,j,l)] + \max_{k}[s_{\text{best}}(i,k) + s_{\text{best}}(k,j)] \text{ if } j - i > 1$$

- Grammar rules are not used
- Label (non-terminal symbol) and break point selected independently

Training

- Margin-based training
- Let T^* be the gold parse tree
- Objective: $s_{\text{tree}}(T^*) > s_{\text{tree}}(T)$ for all $T \neq T^*$
- Compute the best predicted tree $\hat{T}$ under the current model using dynamic programming

$$\hat{T} = \underset{T}{\operatorname{argmax}}[s_{\text{tree}}(T)]$$

- Hinge loss:

$$\sum_{\substack{(T^*,\hat{T}) \in D_{\text{train}} \\ \hat{T} \neq T^*}} \max(0, 1 - (s_{\text{tree}}(T^*) - s_{\text{tree}}(\hat{T})))$$

Evaluating Parsers

- Benchmark corpus: Penn Treebank
- PARSEVAL measures:

$$\text{labeled recall } R = \frac{\#\text{correct constituents in parser's parse of } s}{\#\text{ constituents in treebank's parse of } s}$$

$$\text{labeled precision } P = \frac{\#\text{correct constituents in parser's parse of } s}{\#\text{ constituents in parser's parse of } s}$$

$$\text{labeled } F1 = \frac{2RP}{R+P}$$

- A constituent is correct if there is a constituent in the treebank's parse that spans the same words with the same non-terminal symbol

Parser Accuracy

Parser	F1 Score (%)
PCFG	73.0
Lexicalized PCFG	87.2
Span-based RNN	91.7
Span-based transformer + Pretrained LM	96.0