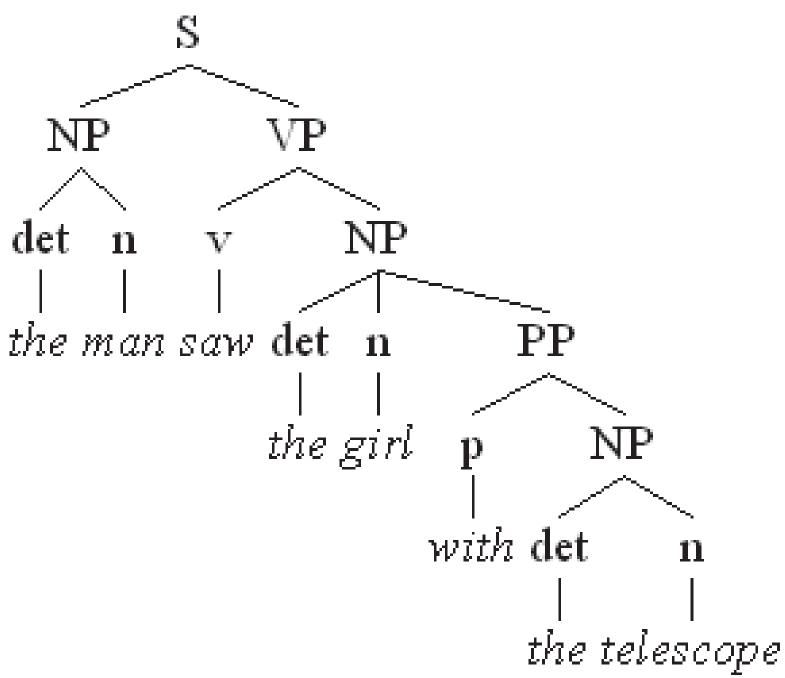
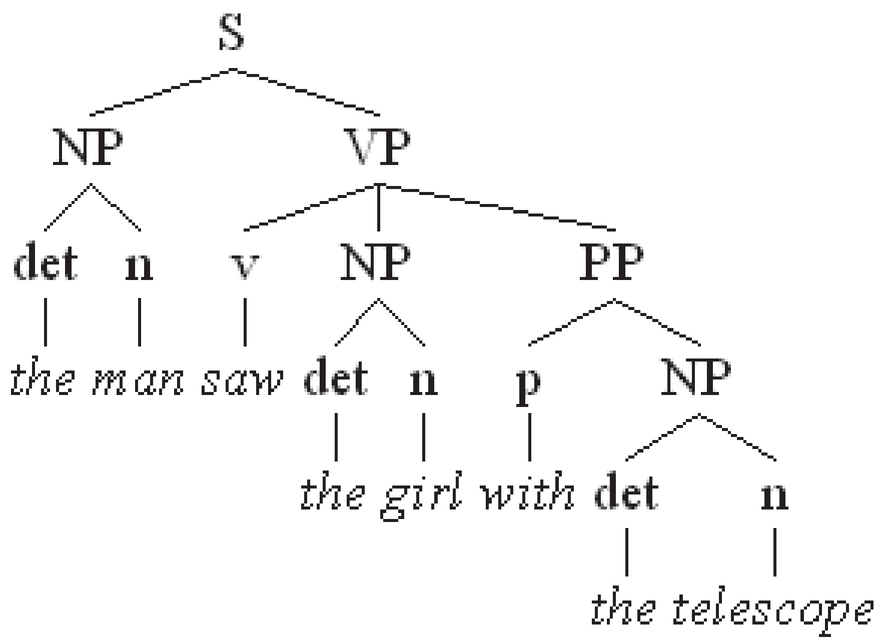
Ambiguity

the man saw the girl with the telescope



The girl has the telescope



The man has the telescope

Probabilistic Context-Free Grammar

- The simplest augmentation of the context-free grammar is the Probabilistic Context-Free Grammar (PCFG).
- Context-free grammar G is defined by four parameters:

N	a set of **non-terminal symbols** (or **variables**)
Σ	a set of **terminal symbols** (disjoint from N)
R	a set of **rules** or productions, each of the form $A \rightarrow \beta$,
	where A is a non-terminal,
	β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$
S	a designated **start symbol**

Probabilistic Context-Free Grammar

- Definition of a CFG:
  - Set of non-terminals (N)
  - Set of terminals (T)
  - Set of rules/productions (P), of the form A
  - Designated start symbol (S)
- Definition of a PCFG:
  - Same as a CFG, but with one more function, D
  - D assigns probabilities to each rule in P

Probabilistic Context-Free Grammar

- A PCFG augments each rule in P with conditional probability:

$$A \rightarrow \beta \; [p]$$

- A PCFG is a 5-tuple $G=(N, \Sigma, P, S, D)$ where D is a function.
- This function expresses the probability p that the given non-terminal A will be expanded to the sequence β as:

$$\Pr(A \rightarrow \beta) \quad \text{or} \quad \Pr(A \rightarrow \beta \mid A)$$

- If we consider all the possible expansions of a non-terminal, the sum of their probabilities must be 1.

Probabilistic Context-Free Grammar

- Attach probabilities to each grammar rule:
- VP → Verb .55

  VP → Verb NP .40

  VP → Verb NP NP .05

A PCFG

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

A probabilistic augmentation of the miniature English grammar and lexicon

Using Probabilities

- A PCFG assigns a probability to each parse-tree T of a sentence S.
- This attribute is useful in disambiguation.
- The probability of a parse T is the product of the probabilities of all rules r used to expand each node n in the parse tree:

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

- Probability P(T,S) is joint probability of the parse and the sentence

$$P(T,S) = P(T)\, P(S|T)$$

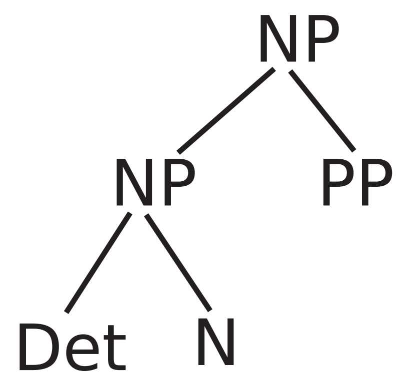
since a parse tree includes all the words of the sentence, $P(S|T) = 1$

Thus: $P(T,S) = P(T)$

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

Using Probabilities : A Sample

NP → Det N	:	0.4
NP → NPposs N	:	0.1
NP → Pronoun	:	0.2
NP → NP PP	:	0.1
NP → N	:	0.2



P(subtree above) = 0.1 x 0.4 = 0.04

English Practice

- What do you understand from the sentence:

  "Can you book TWA flights?"

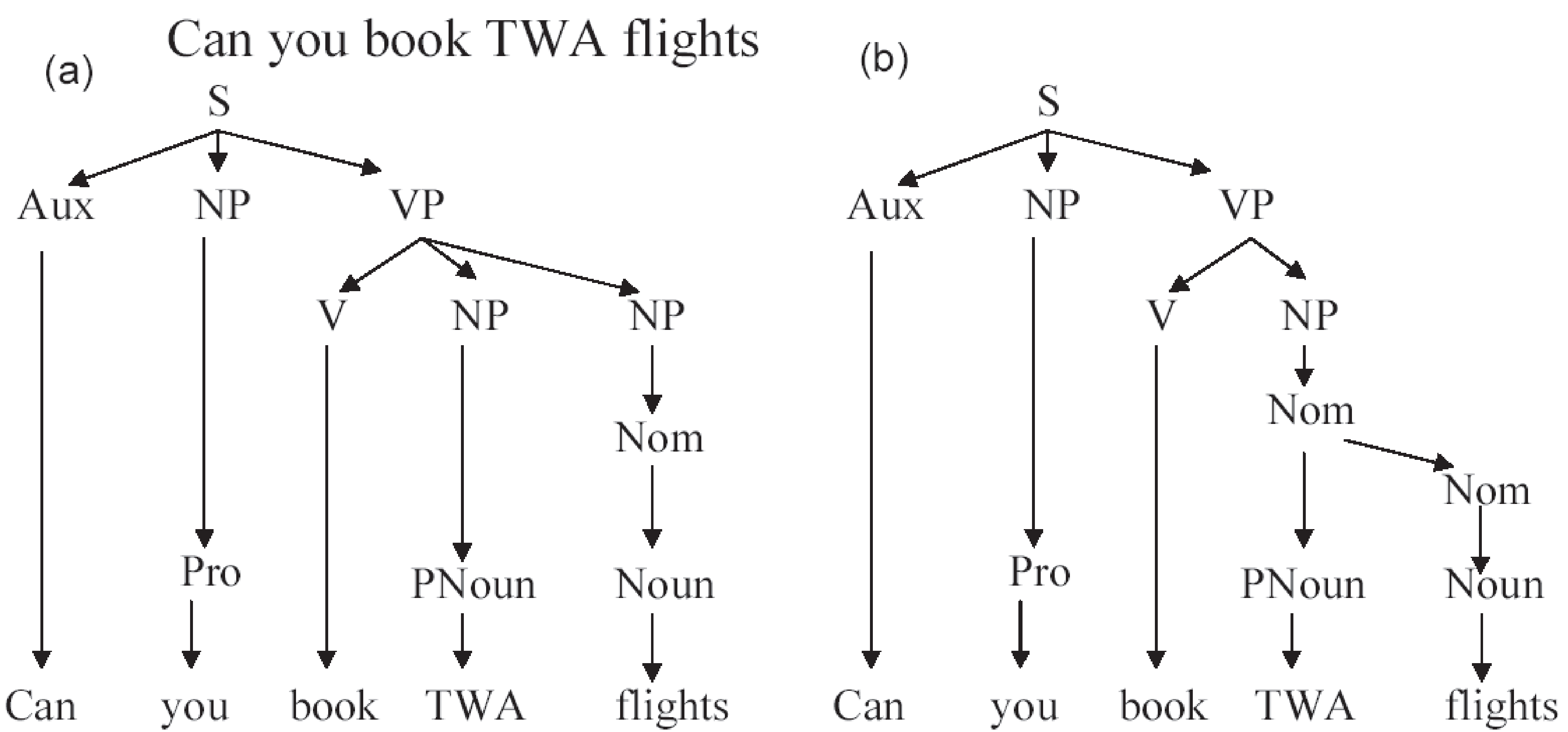
- Can you book flights on behalf of TWA?

  •[TWA] [flights]

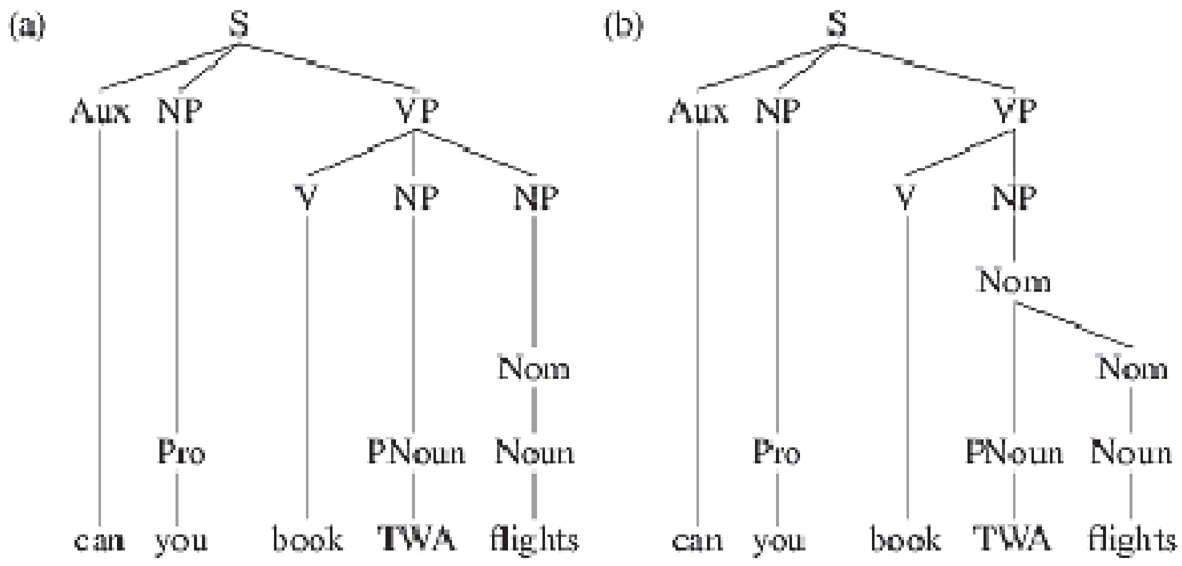
- Can you book flights run by TWA?

  •[TWA flights]

A Sample Parse



A Sample Parse with PCFG



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

A Sample Parse with PCFG

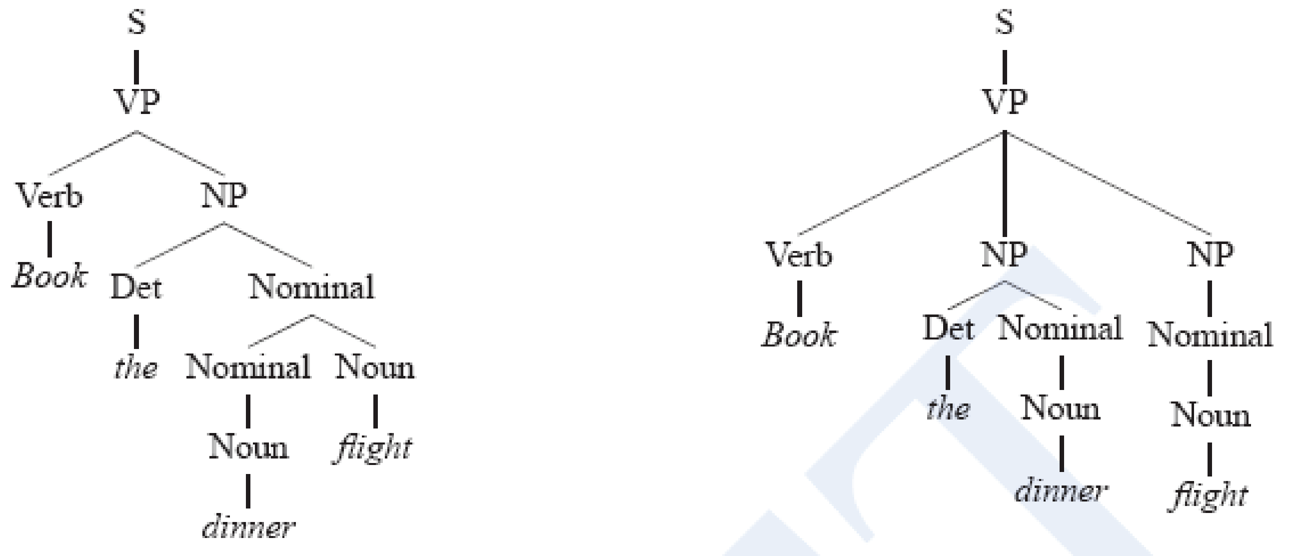
- $P(T_l)$ = .15 x .40 x .05 x .05 x .35 x .75 x .40 x .40 x .40 x .30 x .40 x .50

  = 1.5 x 10 e(-6)
- $P(T_r)$ = .15 x .40 x .40 x .05 x .05 x .75 x .40 x .40 x .40 x .30 x .40 x .50

  = 1.7 x 10 e(-6)
- The right tree T_r has a higher probability and would be chosen by a disambiguation algorithm.
- P(Sentence) = 1.5 x 10 e(-6) + 1.7 x 10 e(-6)

  = 3.2 x 10 e(-6)

A Sam



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

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

Figure 14.2 Two parse trees for an ambiguous sentence, The transitive parse (a) corresponds to the sensible meaning "Book flights that serve dinner", while the ditransitive parse (b) to the nonsensical meaning "Book flights on behalf of 'the dinner'".

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

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

Picking the best parse

- Picking the parse with the highest probability is the correct way to do disambiguation.
- Pick the best tree for a sentence S out of the set of parse trees for S.

$$\hat{T}(S) = \underset{T \in \tau(S)}{\operatorname{argmax}} P(T|S)$$

Parse tree *T* which is most likely given the sentence *S.*

$$= \operatorname{argmax} \frac{P(T,S)}{P(S)}$$

P(T|S) can be rewritten as P(T,S)/P(S).

$$= \operatorname{argmax} P(T,S)$$

Since we are maximizing over all parse for the same sentence P(S) will be a constant.

$$= \operatorname{argmax} P(T)$$

Since P(T,S)=P(T).

The most likely parse is choosing the parse with the highest probability

Getting Probabilities

- From an annotated database (a treebank)
- Learned from a corpus

Getting Probabilities – from treebank

- Get a large collection of parsed sentences
- Compute probability for each non-terminal rule expansion in the collection
- Normalize
- Done

Getting Probabilities – learn from corpus

- What if you don’t have a treebank (and can’t get one)
- Take a large collection of text and parse it.
- In the case of syntactically ambiguous sentences collect all the possible parses
- Prorate the rule statistics gathered for rules in the ambiguous case by their probability
- Proceed as you did with a treebank.
- **Inside-Outside** algorithm

Parsing of PCFGs

Probabilistic CYK

- Probabilistic CYK (Cocke-Younger-Kasami) algorithm for parsing Probabilistic CFG.
- Bottom-up dynamic parsing algorithm.
- Assume PCFG is in Chomsky Normal Form (CNF): production is either of the form $A \rightarrow B\ C$ or $A \rightarrow a$

$A \rightarrow BC$

Non-terminal Non-terminal

$A \rightarrow a$

Terminal

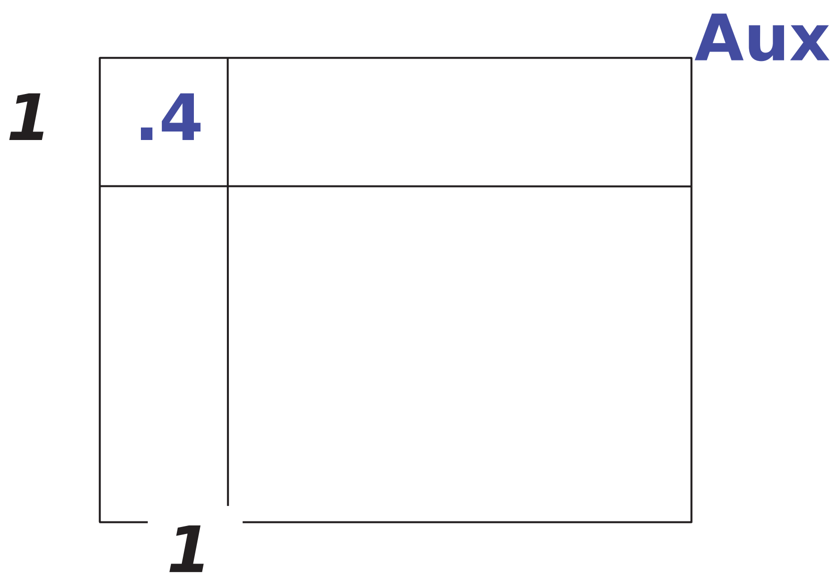
Probabilistic CYK

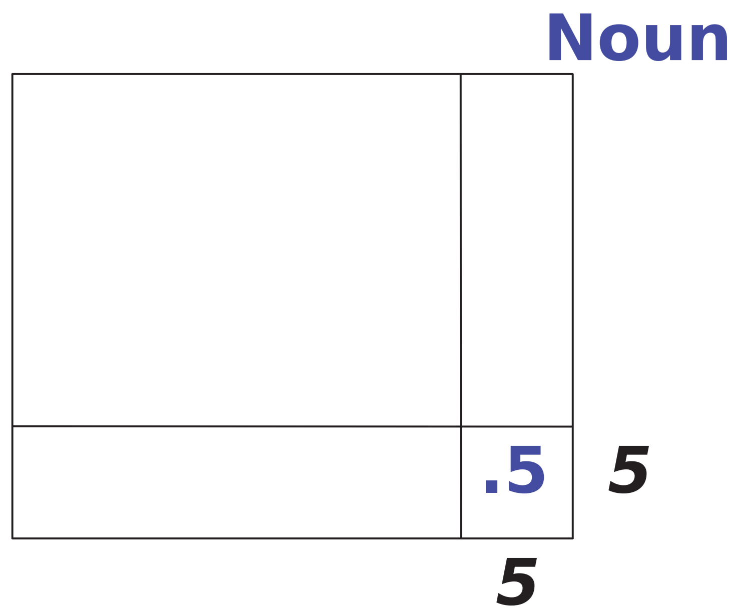
- CYK Algorithm: bottom-up parser
- Input:
  - A Chomsky normal form PCFG, G= (N, Σ, P, S, D). Assume that the $|N|$ non-terminals have indices 1, 2, …, $|N|$, and the start symbol S has index 1.
  - n words $w_1, \ldots, w_n$
- Data Structure:
  - A dynamic programming array $\Pi\pi[i,j,a]$ holds the maximum probability for a constituent with non-terminal index a spanning words $i..j$.
- Output:
  - The maximum probability parse $\Pi\pi[1,n,1]$: the parse tree whose root is S and which spans the entire string of words $w_1, \ldots, w_n$

CYK Algorithm : Base Case

- Consider the input *strings of length one* (individual words w_i)
- In CNF, the probability of a given non-terminal A expanding to a single word w_i must come only from the rule $A \rightarrow w_i$ *i*.e., $P(A \rightarrow w_i)$
- $\Pi\pi[i,j,a] = P(A \longrightarrow w_i)$

"Can$_1$ you$_2$ book$_3$ TWA$_4$ flights$_5$?"





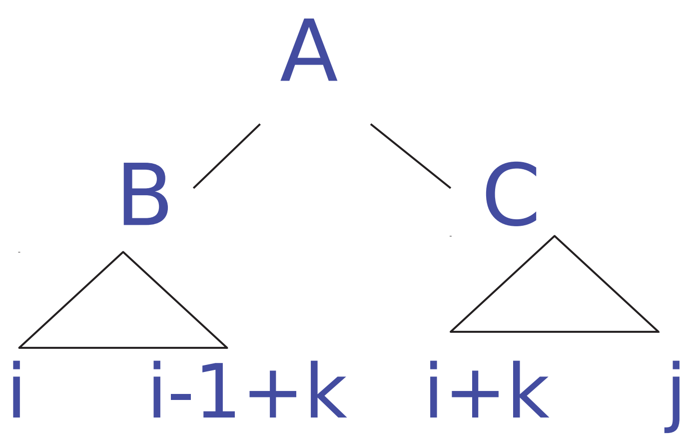
CYK Algorithm : Recursive Case

- For strings of words of length > 1,

  A $\Rightarrow$ w_{ij} *iff* there is at least one rule A $\rightarrow$BC

  where B derives the first *k* words (between *i* and *i*-1+*k*) and

  C derives the remaining ones (between *i*+*k* and *j*)
- (for each non-terminal) Choose the max among all possibilities
- $\pi\pi[i,j,A] = \pi\pi[i, i\text{-}1+k, B] \times \pi\pi[i+k, j, C] \times P(A \rightarrow BC)$



CYK Algorithm :

Function CYK(*words*, *grammar*)
    return the most probable parse and its probability

For i = ← 1 **to** *num_words*   #base case
    for *a* = ← 1 **to** *num_nonterminals*
        If ($A \rightarrow w_i$) is in grammar **then** $\pi\Pi[i, i, a] = \leftarrow P(A \rightarrow \rightarrow w_i)$

For ***j*** = ← 2 **to** *num_words*   #recursive
    For *i* = ← 1 **to** *num_words* – *j* + 1
        For *k* = ← 1 **to** *j* – 1
          for *A* = ← 1 **to** *num_nonterminals*
          for *B* = ← 1 **to** *num_nonterminals*
          for *C* = ← 1 **to** *num_nonterminals*
              $prob = \leftarrow \Pi\pi[i, k, B] \times \Pi\pi[i+k, j\text{-}k, C] \times P(A \rightarrow \rightarrow BC)$
              If ($prob > \pi\Pi[i, j, A]$) **then**
            $\pi\Pi[i, j, A] = prob$
            $back[i, j, A] = \{k, A, B\}$
Return *build_tree(back[1, num_words, 1]), π[1, num_words, 1]*

CYK Algorithm :

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]
The	flight	includes	a	meal

S	$\rightarrow$	$NP\ VP$	.80	Det	$\rightarrow$	the	.50
NP	$\rightarrow$	$Det\ N$	.30	Det	$\rightarrow$	a	.40
VP	$\rightarrow$	$V\ NP$	.20	N	$\rightarrow$	$meal$	.01
V	$\rightarrow$	$includes$	.05	N	$\rightarrow$	$flight$	.02

Problems with PCFG

- Poor independence assumptions
  - CFG impose an independence assumption on probabilities
  - Results in poor modeling of structural dependencies
- Lack of lexical conditioning
  - CFG rules don't model syntactic facts about words leading to problems with:

    subcategorization ambiguities

    preposition attachment

    coordinate structure ambiguities

Problems with PCFG – 1

- The expansion of any one non-terminal is independent of the expansion of any other non-terminal – assumption
- Statistics of English syntax shows that the choice of how a node expands is dependent on the location of the node in the parse tree.
- NP → Det NN .28

  NP → ProNoun .25
- PCFG don't allow a rule probability to be conditioned on surrounding context – hence equal probability

Problems with PCFG – 1

- The probability of expanding an *NP* as a pronoun versus a lexical *NP* were *conditioned* on whether the *NP* was a subject or an object.
- P[(NP→Pronoun) | NP=subj] >> P[(NP→Pronoun) | NP =obj]

	Pronoun	Non-Pronoun
Subject	91%	9%
Object	34%	66%

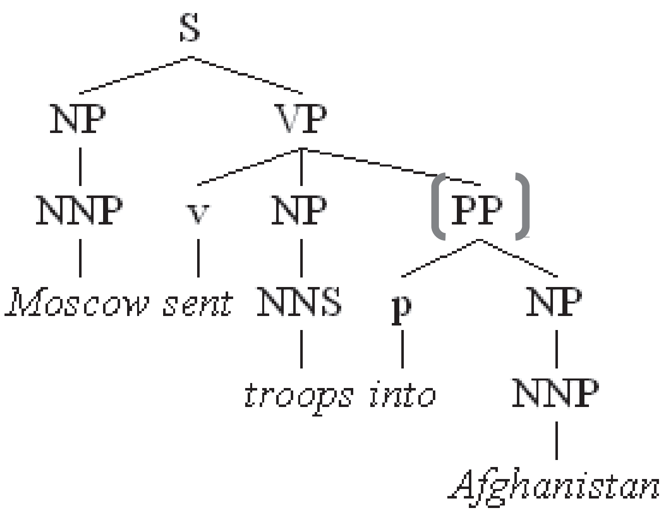
Problems with PCFG – 2

- Lexical information plays an important role in selecting the correct parsing of an ambiguous *PP* attachment
- Moscow *sent* more than 100,000 soldiers *into Afghanistan*.
- Here the PP [*into Afghanistan*] can be attached either to the *NP* [*more than 100,000 soldiers*] or to the VP headed by *sent*
- In PCFG, the choice between two rules:

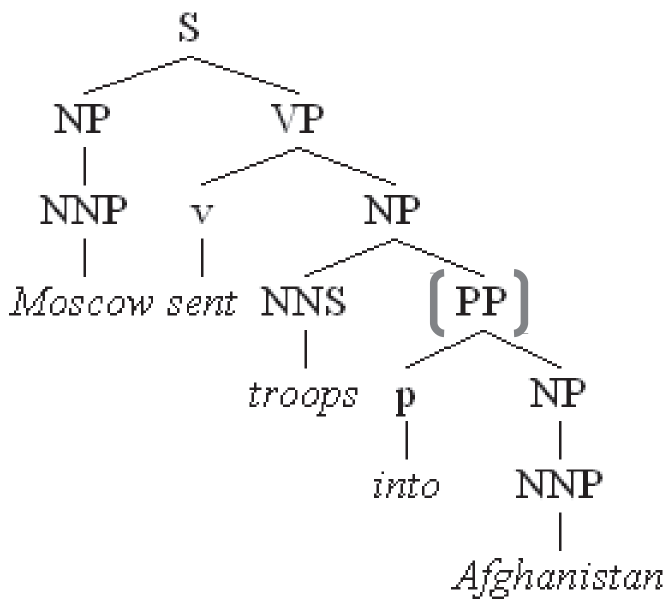
	$NP \rightarrow NP\ PP$	$VP \rightarrow NP\ PP$
Hindle and Rooth (1991)	*67%*	*33%*
Collins (1999)	*52%*	*48%*

- Verb *send* subcategorizes for destination, which can be expressed with *into*
- Keep separate lexical dependency statistics for different verbs

Problems with PCFG – 2



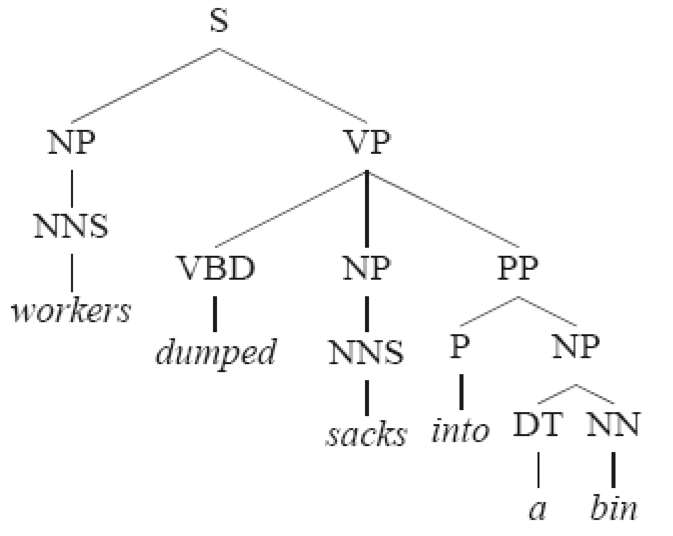
VP-attachment

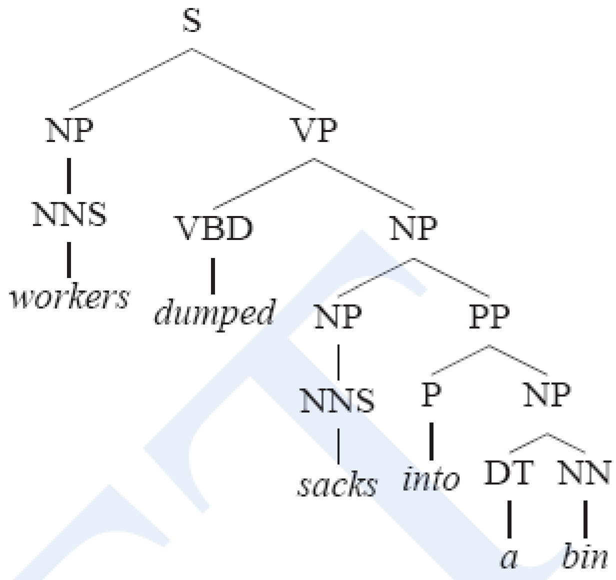


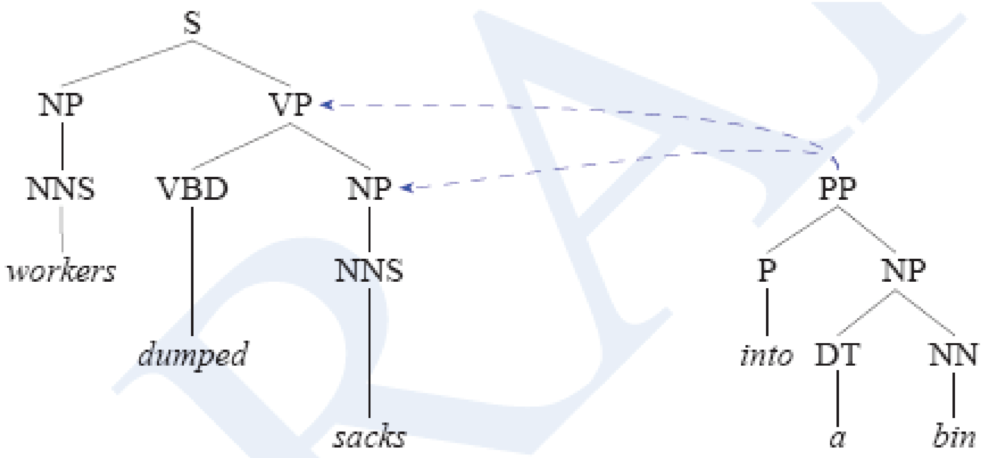
NP-attachment

Typically NP-attachment more frequent than VP-attachment

Problems with PCFG – 2





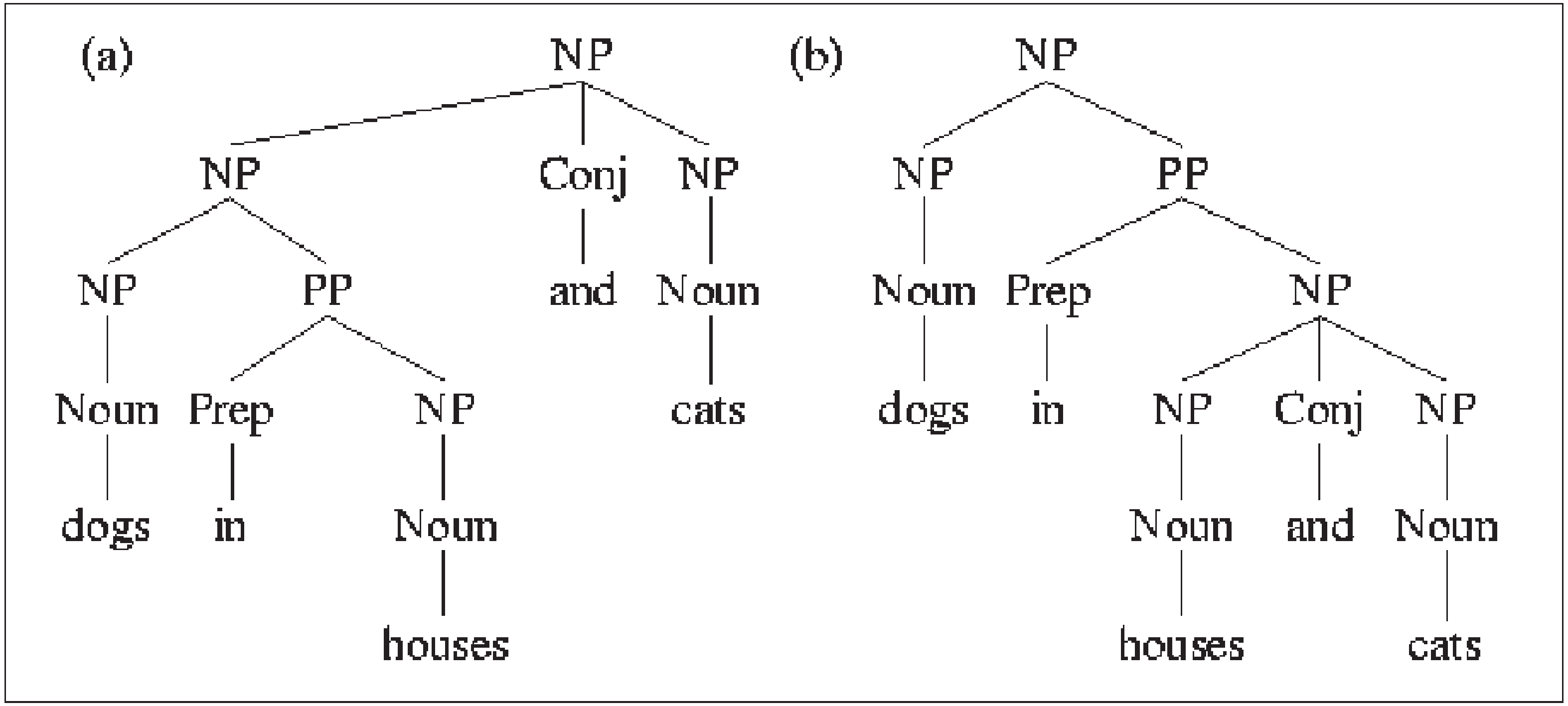


$VP \rightarrow VBD\ NP\ PP$

$NP \rightarrow NP\ PP$

Problems with PCFG – 3

- Coordination ambiguity:
- [[*dogs in houses*] *and* [*cats*]]
  dogs in [[*houses*] *and* [*cats*]]



- A PCFG assigns identical probabilities, since both structure use the exact same rules

Probabilistic Lexicalized CFG

Probabilistic Lexicalized CFG

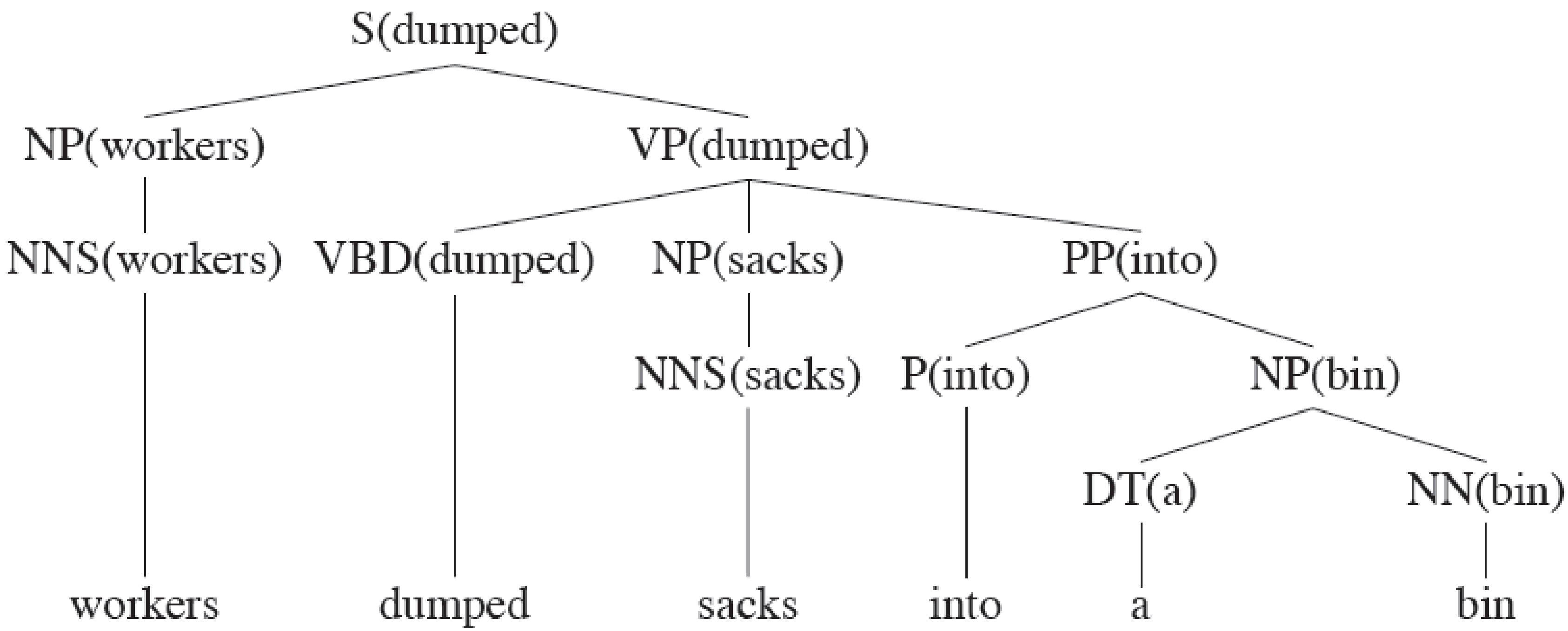
- Syntactic constituents could be associated with a lexical head.
- Each non-terminal in a parse tree is annotated with its lexical head.
- Each PCFG rule must be augmented to identify one RHS constituent to be the head daughter.
- The headword for a node is set to the headword of its head daughter.

  VP -> VBD NP PP
  VP(dumped) -> VBD(dumped) NP(sacks) PP(into)

- In a standard lexicalized grammar, associate the head tag with the POS tags of the headwords:

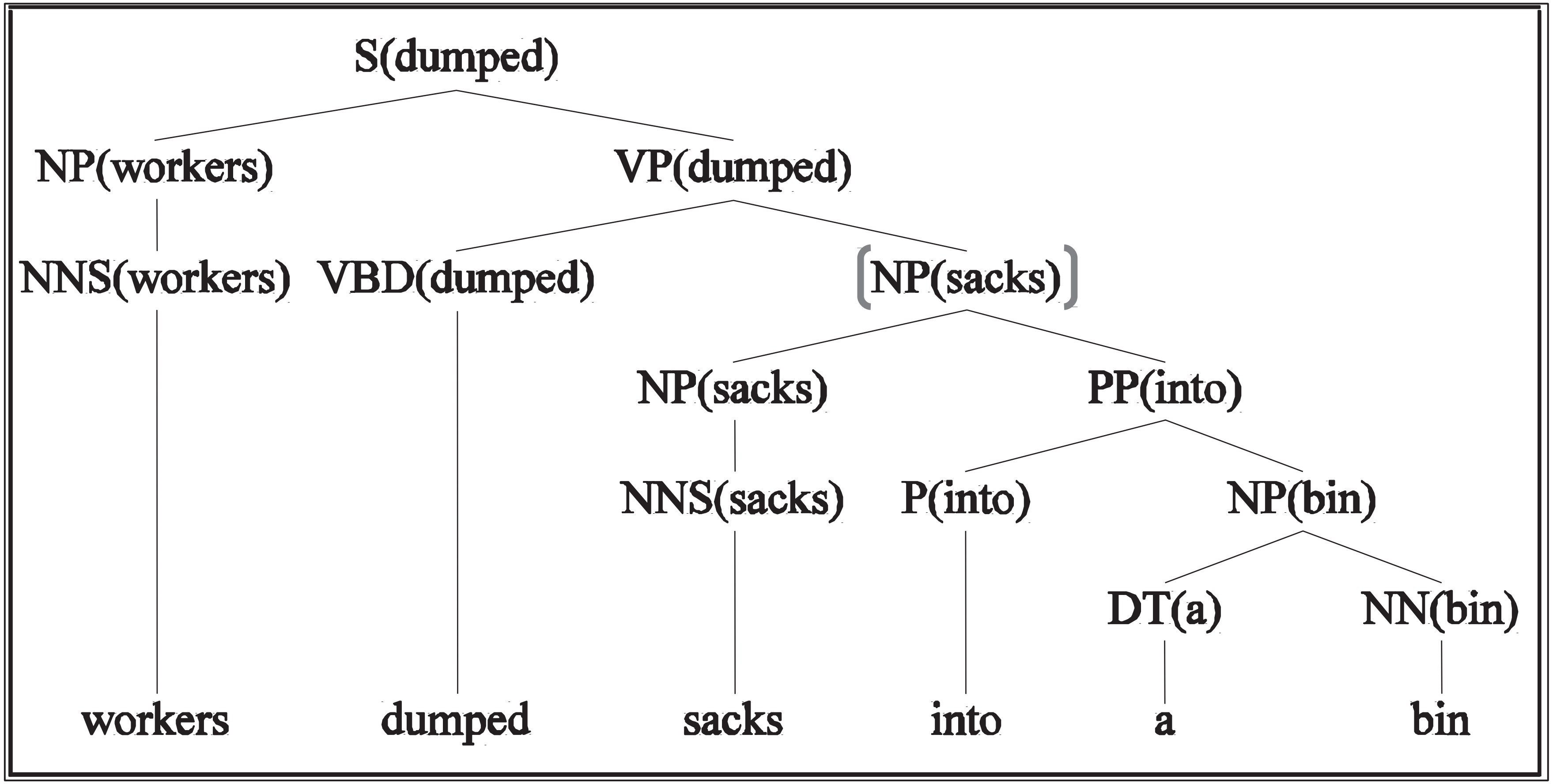
  VP(dumped,VBD) -> VBD(dumped,VBD) NP(sacks,NNS) PP(into,IN)

Probabilistic Lexicalized CFG



A lexicalized tree from Collins (1999)

Probabilistic Lexicalized CFG – Incorrect parse



An incorrect parse of the sentence from Collins (1999)

Probabilistic Lexicalized CFG

- Add the headword of the node $h(n)$ for computing the probability of a node being expanded via rule r.

  $p(r(n)|n, h(n))$ [$p(r(n))$ – syntactic category of n] – (1)

  $r = VP \rightarrow VBD\ NP\ PP$

  $p(r|VP, dumped)$

- What is the probability that a *VP* headed by *dumped* will be expanded as VBD NP PP?
- How to compute the probability of a head?
- The probability of a node n having a head h on two factors:
  - The syntactic category of the node n
  - The head of the node's mother $h(m(n))$

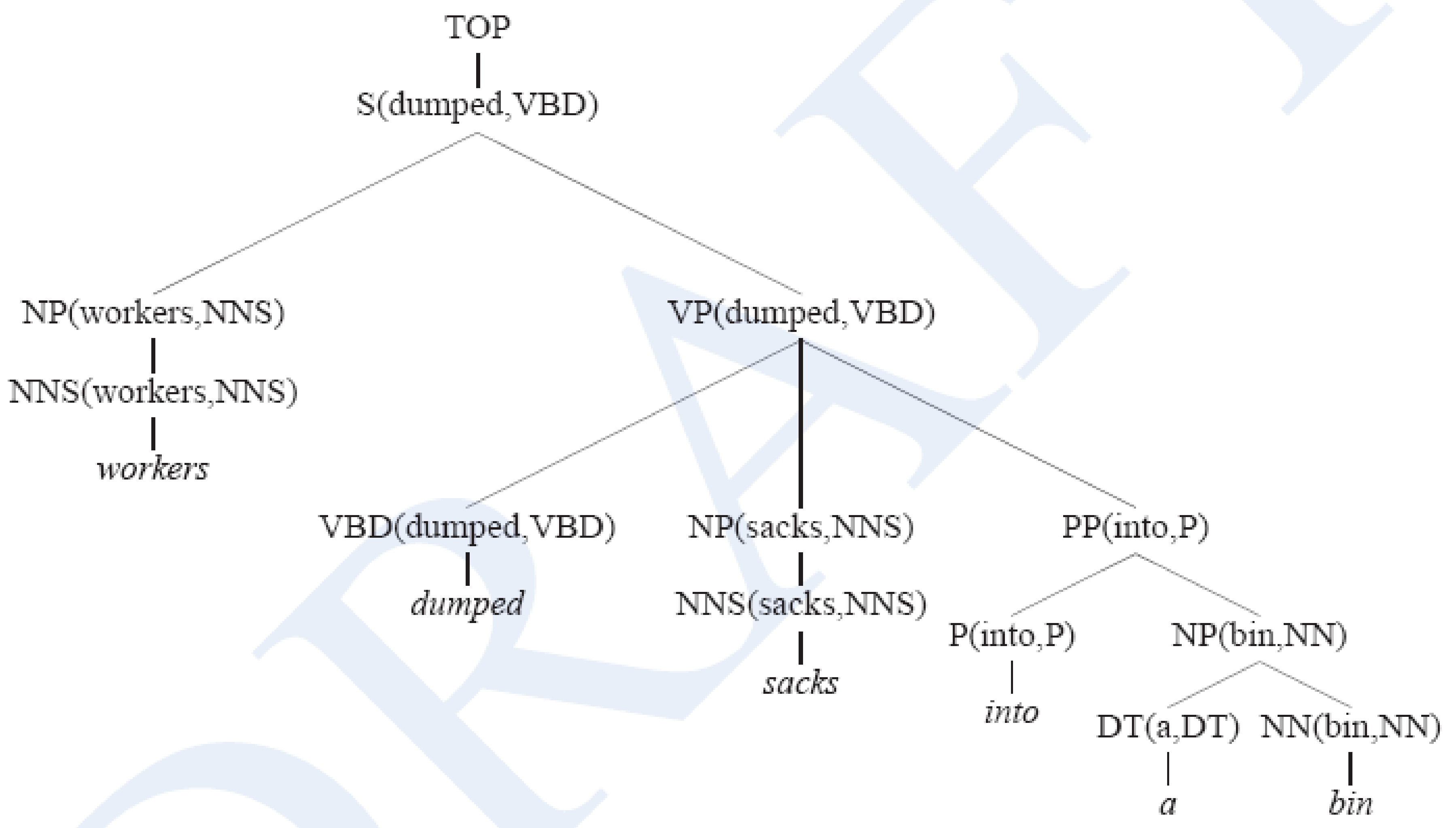
Probabilistic Lexicalized CFG

- $P\,(h(n) = word_i \mid n,\ h(m(n)))$ – (2)
- $p(head(n)=sacks \mid n=NP, h(m(n))=dumped)$
- What is the probability that an *NP* whose mother's head is *dumped* has the head *sacks?*

$$\begin{array}{c} X(dumped) \\ | \\ NP(?sacks?) \end{array}$$

- Head-probability is capturing *dependency information* between the words *dumped* and *sacks*

P



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

Lexical Rules		
NNS(workers,NNS)	→	workers
VBD(dumped,VBD)	→	dumped
NNS(sacks,NNS)	→	sacks
P(into,P)	→	into
DT(a,DT)	→	a
NN(bin,NN)	→	bin

Probabilistic Lexicalized CFG

- Probability of a parse is given by:

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

- The head-rule and head-head probabilities will correctly choose the *VP* attachment over the *NP* attachment.
- Head-rule probabilities calculated as:

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

how many times the rule occurs with h(n) as the headword versus how many times the mother/headword combination appear in total.

Probabilistic Lexicalized CFG

- From the Brown corpus: head-rule probabilities

  $p(VP \rightarrow VBD\ NP\ PP \mid VP, \mathit{dumped}) = 6 / 9 = .67$

  $p(VP \rightarrow VBD\ NP \mid VP, \mathit{dumped}) = 0 / 9 = 0$

- What about the head-head probabilities?

  $p(\mathit{into} \mid PP, \mathit{dumped}) = 2 / 9 = .22$

  $p(\mathit{into} \mid PP, \mathit{sacks}) = 0 / 0 = 0$

- So, the contribution of this part of the parse to the total scores for the two candidates is:

  [dumped into] $.67 \times .22 = .147$

  [sacks into] $0 \times 0 = 0$

References

- Slides were adapted from:
  Speech and Language Processing, *Jurfsky and Martin*