Statistical Parsing

Statistical Parse Disambiguation

Problem: ***How do we disambiguate among a set of parses of a given sentence?***

- We want to pick the parse tree that corresponds to the correct meaning.

Possible Solutions:

- Pass the problem onto Semantic Processing
- Use a probabilistic model to assign likelihoods to the alternative parse trees and select the best one.
  - **Associating probabilities with the grammar rules gives us such a model.**
  - **The most commonly used probabilistic grammar formalism is the probabilistic context-free grammar (PCFG), a probabilistic augmentation of context-free grammars in which each rule is associated with a probability.**

Probabilistic Context-Free Grammars (PCFGs)

- **The simplest augmentation of the context-free grammar is the Probabilistic Context-Free Grammar (PCFG), also known as the Stochastic Context-Free Grammar (SCFG).**

- A PCFG differs from a CFG by augmenting each rule with a conditional probability:

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

- Here p expresses the probability that non-terminal A will be expanded to sequence β.

- We can represent this probability as:

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

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

$$\sum_{\beta} P(A \rightarrow \beta) = 1$$

Probabilistic CFGs

- Associate a probability with each grammar rule.
- The probability reflects relative likelihood of using the rule in generating the LHS constituent.
- Assume for a constituent C we have k grammar rules of form $C \rightarrow \alpha_i$.
- We are interested in calculating $P(C \rightarrow \alpha_i | C)$: the probability of using rule i for deriving C.
- Such probabilities can be estimated from a corpus of parse trees:

$$P(C \rightarrow \alpha_i \mid C) = \frac{count(C \rightarrow \alpha_i)}{\sum_{j=1}^{k} count(C \rightarrow \alpha_j)} = \frac{count(C \rightarrow \alpha_i)}{count(C)}$$

Probabilistic CFGs

- Attach probabilities to grammar rules
- The expansions for a given non-terminal sum to 1

VP $\rightarrow$ Verb	[.55]
VP $\rightarrow$ Verb NP	[.40]
VP $\rightarrow$ Verb NP NP	[.05]

Assigning Probabilities to Parse Trees

- Assume that probability of a constituent is independent of context in which it appears in the parse tree.
- Probability of a constituent C' that was constructed from $A_1', \ldots, A_n'$ using the rule $C \rightarrow A_1, \ldots, A_n$ is:

$$P(C') = P(C \rightarrow A_1, \ldots, A_n | C)\ P(A_1') \ldots P(A_n')$$

- At the leafs of the tree, we use the POS probabilities $P(w_i|C)$.
- A derivation (tree) consists of the set of grammar rules that are in the tree
- The probability of a derivation (tree) is just the product of the probabilities of the rules in the derivation.

Assigning Probabilities to Parse Trees (Ex. Grammar)

S $\rightarrow$ NP VP	[0.6]
S $\rightarrow$ VP	[0.4]
NP $\rightarrow$ Noun	[1.0]
VP $\rightarrow$ Verb	[0.3]
VP $\rightarrow$ Verb NP	[0.7]
Noun $\rightarrow$ book	[0.2]
.	
.	
Verb $\rightarrow$ book	[0.1]

Parse Trees for : book book

- [S [NP [Noun book]] [VP [Verb book]]]

  P([Noun book]) = P(Noun→book) = 0.1

  P([Verb book]) = P(Verb→book) = 0.2

  P([NP [Noun book]]) = P(NP→Noun) P([Noun book]) = 1.0*0.1 = 0.1

  P([VP [Verb book]]) = P(VP→Verb) P([Verb book]) = 0.3*0.2 = 0.06

  P [S [NP [Noun book]] [VP [Verb book]]])

  = P(S→NP VP)*0.1*0.06 = 0.6*0.1*0.06 = 0.0036

- [S [VP [Verb book] [NP [Noun book]]]]

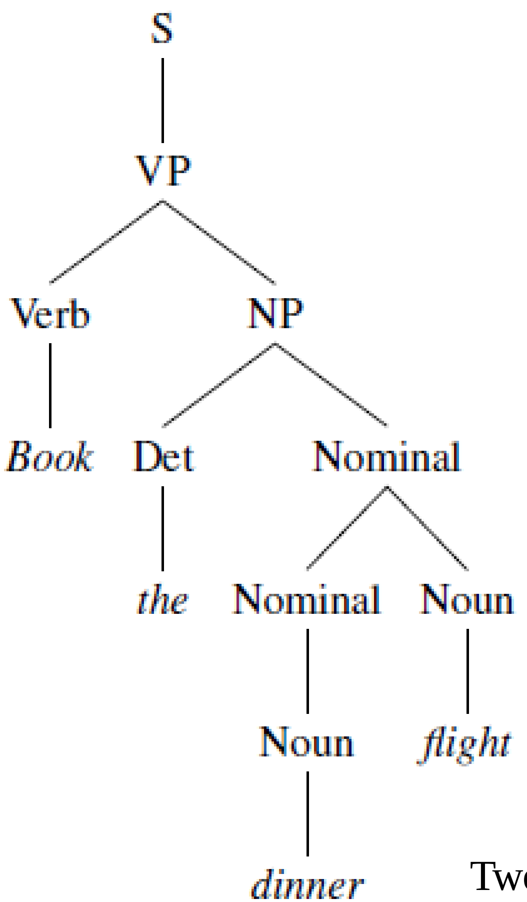
  P([VP [Verb book] [NP [Noun book]]]) = P(VP→Verb NP)*0.2*0.1 = 0.7*0.2*0.1 = 0.014

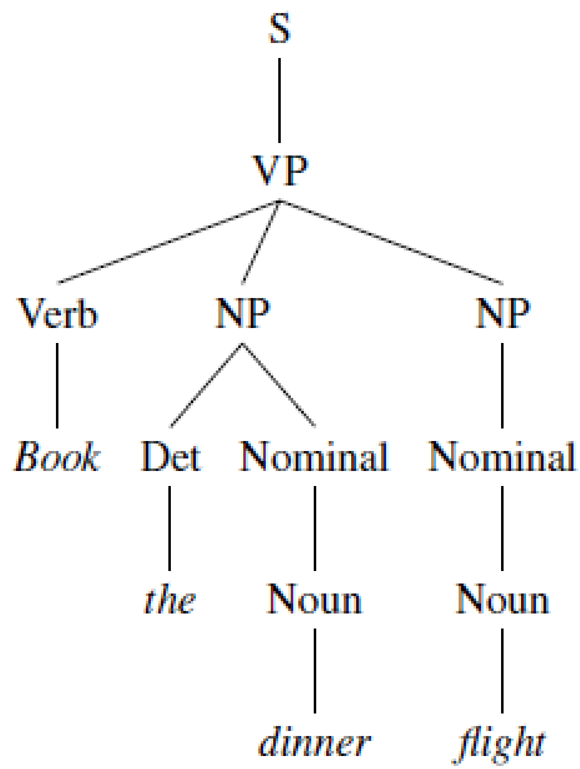
  P([S [VP [Verb book] [NP [Noun book]]]]) = P(S→VP)*0.014 = 0.4*.014 = 0.0056

Example: A PCFG

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]	| $meal$ [.05] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flight$ [.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]	

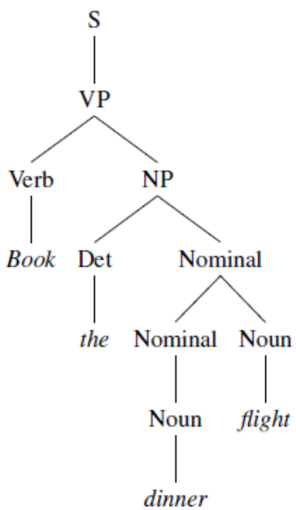
Assigning Probabilities to Parse Trees



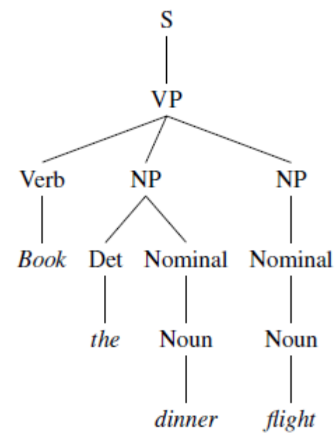


Two parse trees for an ambiguous sentence. The parse on the left corresponds to the sensible meaning "Book a flight that serves dinner", while the parse on the right corresponds to the nonsensical meaning "Book a flight on behalf of 'the dinner' "

Assigning Probabilities to Parse Trees



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	→	flight	.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	→	flight	.40

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

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

Problems with PCFGs

Poor independence assumptions: Main problem with Probabilistic CFG Model is that it does not take contextual effects into account.

- For example, pronouns are much more likely to appear in the subject position of a sentence than an object position.
  - In *Switchboard* corpus:

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

- Unfortunately, there is no way to represent this contextual difference in probabilities in a PCFG because rule **NP→Pronoun** has only one probability in a PCFG.
  - For *Switchboard* corpus:
  - Rule NP→Pronoun should have .91 probability value in subject positions and .34 probability value in object positions.

Problems with PCFGs

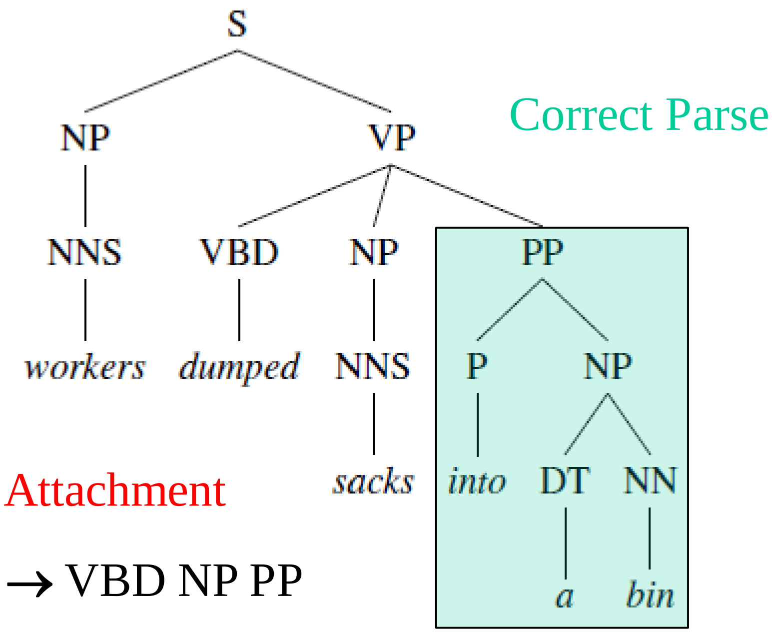
Lack of Sensitivity to Lexical Dependencies: PCFG rules don’t model syntactic facts about specific words, leading to problems with subcategorization ambiguities, preposition attachment, and coordinate structure ambiguities.

- Although words play a role in PCFGs since the parse probability includes the probability of a word given a part-of-speech, words (lexical information) is NOT used to resolve structure ambiguities such as prepositional phrase (PP) attachment ambiguities.
  - Prepositional phrases can attach to NP or VP nodes.

Problems with PCFGs

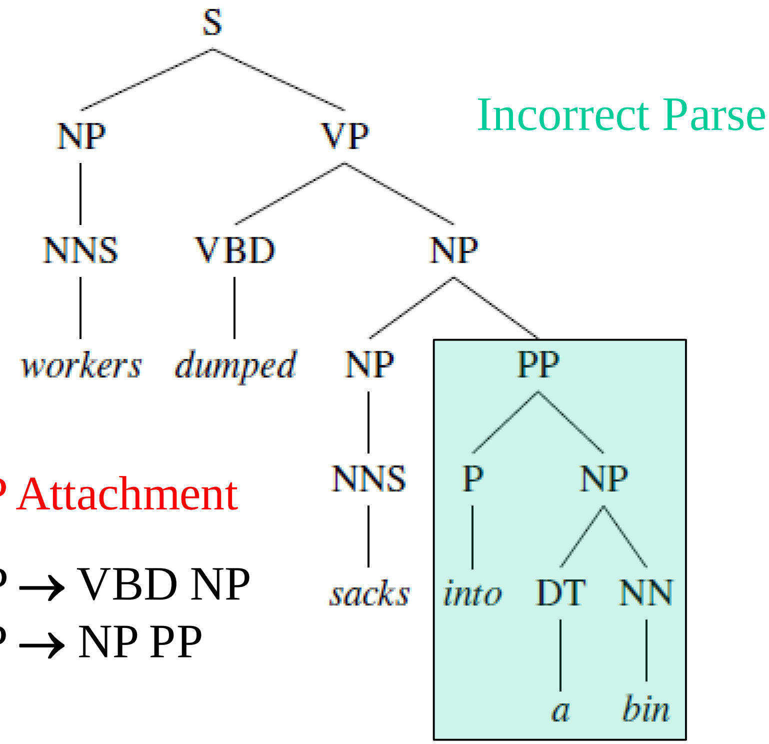
PP attachment ambiguities

Example: Workers dumped sacks into a bin



VP Attachment

$VP \rightarrow VBD\ NP\ PP$



NP Attachment

$VP \rightarrow VBD\ NP$

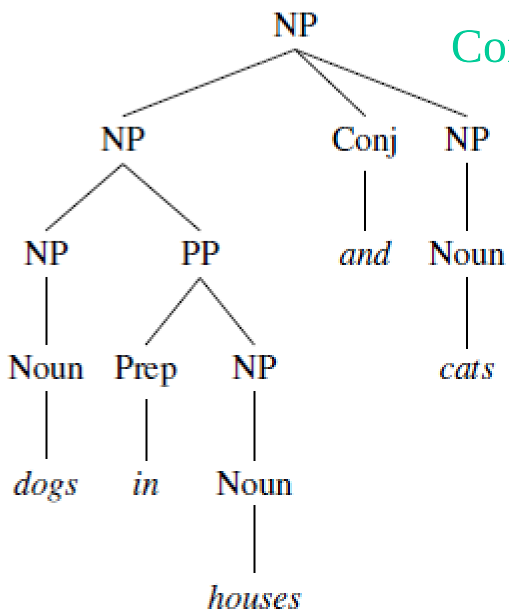
$NP \rightarrow NP\ PP$

- Depending on how these probabilities are set, a PCFG will always prefer NP attachment or VP attachment.
- NP attachment is slightly more common in English, we might always prefer NP attachment, causing us to misparse this sentence.

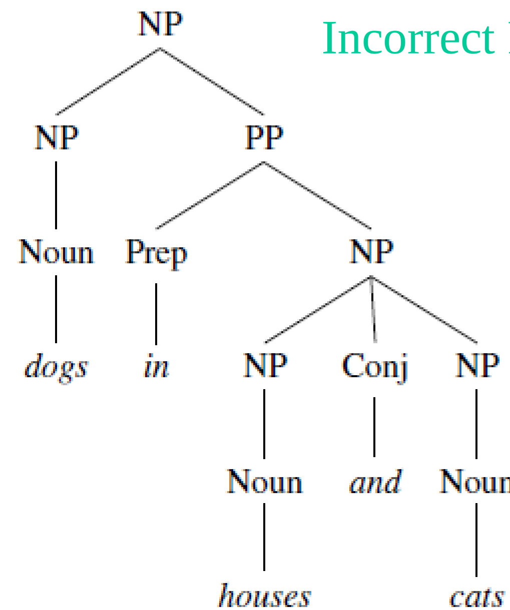
Problems with PCFGs

Coordination ambiguities

Example: dogs in houses and cats



Correct Parse



Incorrect Parse

- Because *dogs* is semantically a better conjunct for *cats* than *houses* (and because most *dogs* can't fit inside *cats*), the second parse is intuitively unnatural and should be dis-preferred.
- However these two parses have exactly same PCFG rules, and a PCFG will assign them same probability.

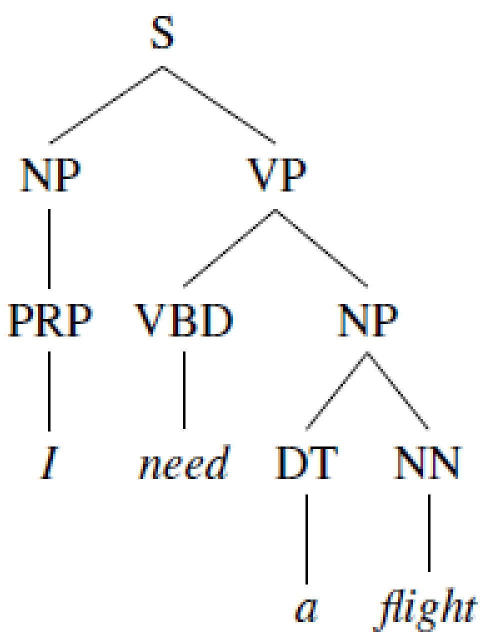
Improving PCFGs by Splitting Non-Terminals

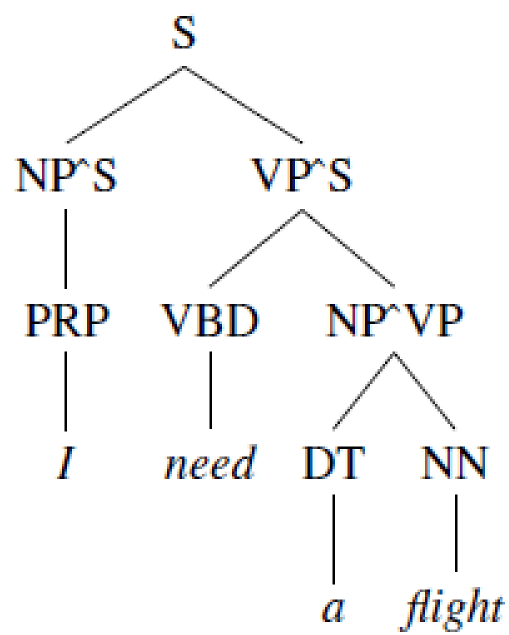
- PCFGs are not able to model structural dependencies such as:
  - NPs in subject position tend to be pronouns, whereas NPs in object position tend to have full lexical form.

- How could we augment a PCFG to correctly model this fact?
  - One idea to **split** NP non-terminal into two versions: one for subjects, one for objects.
  - Having two nodes (e.g., $\mathbf{NP_{subject}}$ and $\mathbf{NP_{object}}$) would allow us to correctly model their different distributional properties, since we would have different probabilities for the rule $\mathbf{NP_{subject} \rightarrow Pronoun}$ and the rule $\mathbf{NP_{object} \rightarrow Pronoun}$.

- One way to implement this intuition of splits is to do **parent annotation** in which we annotate each node with its parent in the parse tree.
  - Thus, an NP node that is the subject of the sentence and hence has parent S would be annotated **NPˆS**, while a direct object NP whose parent is VP would be annotated **NPˆVP**.

Improving PCFGs by Splitting Non-Terminals



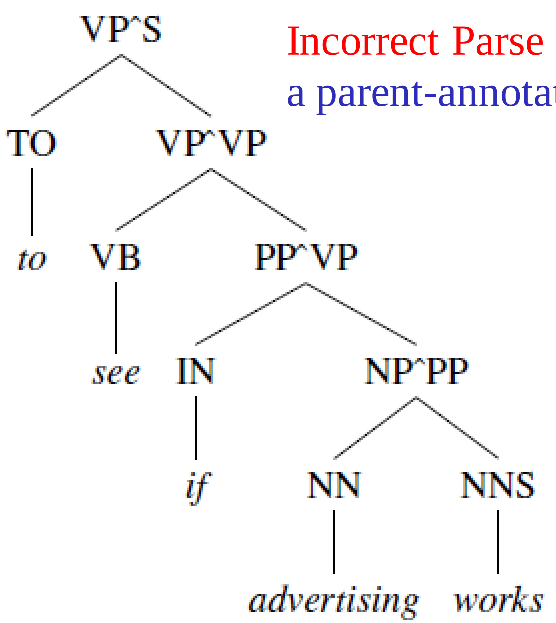


- A standard PCFG parse tree

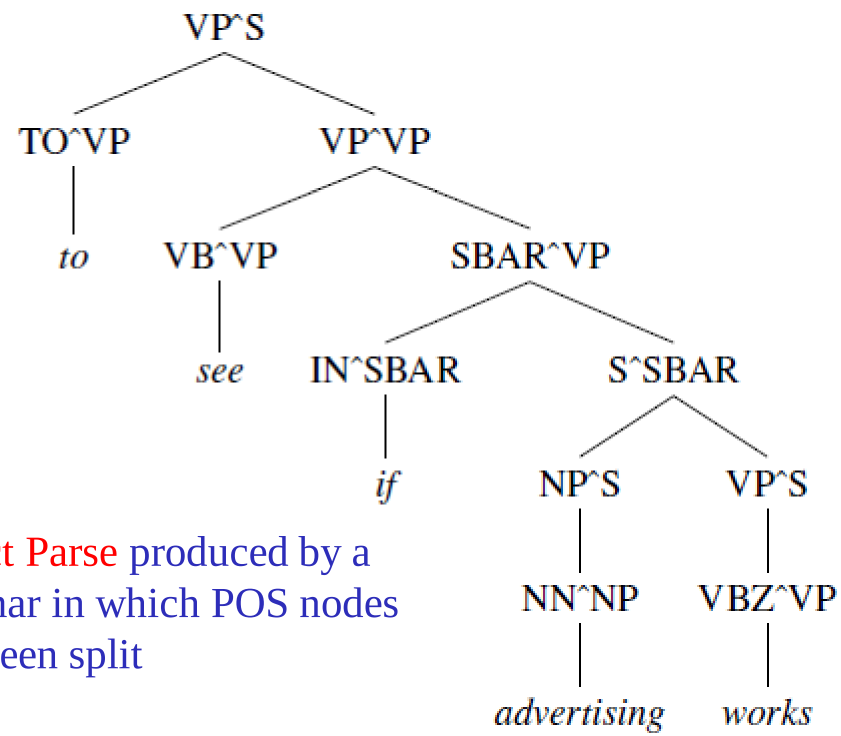
- A parse tree which has **parent annotation** on the nodes which aren't pre-terminal.
- All the non-terminal nodes (except the pre-terminal part-of-speech nodes) in parse have been annotated with the identity of their parent.

Improving PCFGs by Splitting Non-Terminals

- We can also improve a PCFG by splitting the pre-terminal part-of-speech nodes.
  - Different kinds of adverbs (RB) tend to occur in different syntactic positions:
    - most common adverbs with ADVP parents are *also* and *now*, most common adverbs with VP parents are *not*, and most common adverbs with NP parents are *only* and *just*.
    - Thus, add tags like RB^ADVP, RB^VP, and RB^NP to improve PCFG modeling.



Incorrect Parse even with a parent-annotated parse



Correct Parse produced by a grammar in which POS nodes have been split

Probabilistic Lexicalized CFGs

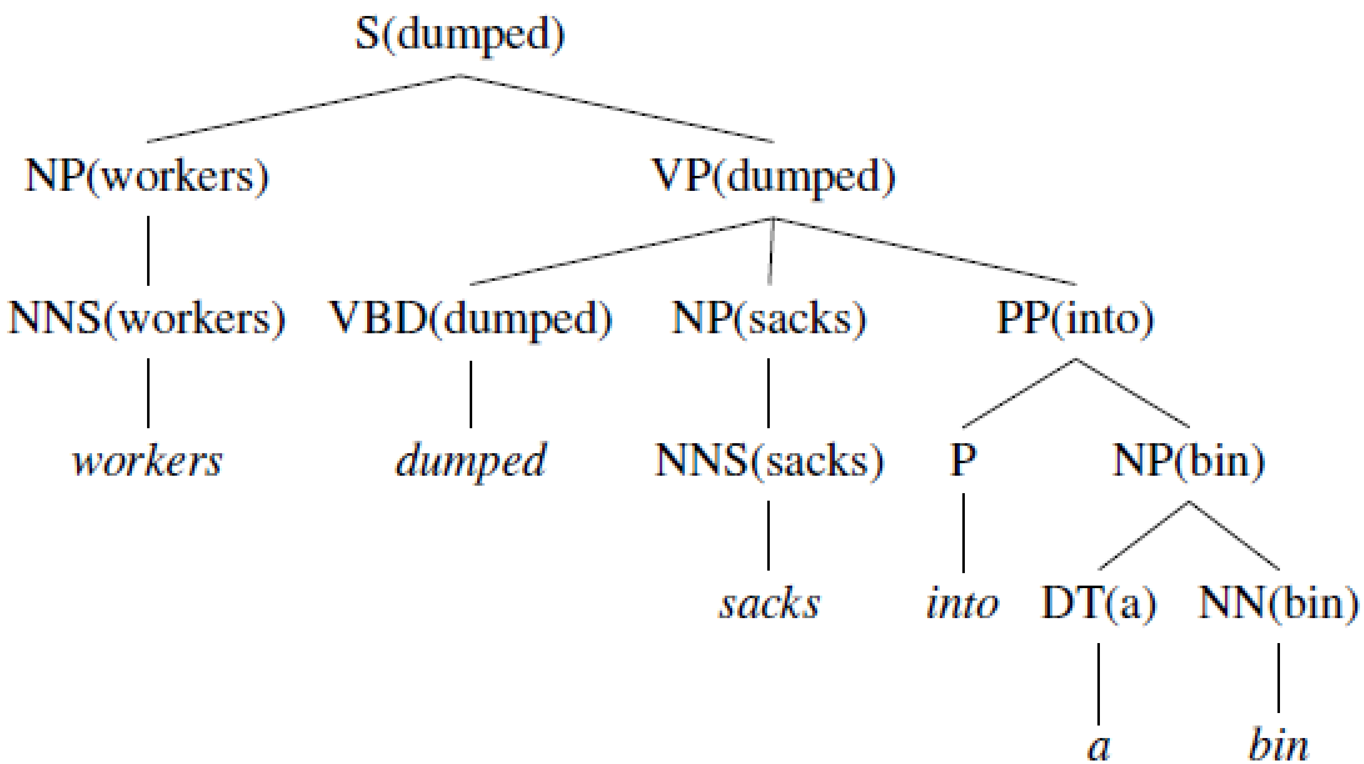
- Syntactic constituents can be associated with a **lexical head**.
- We can define a **lexicalized grammar** in which each non-terminal grammar is annotated with its **lexical head.**
- In a lexicalized grammar, the rule VP → VBD NP PP would be extended as

  VP(dumped) → VBD(dumped) NP(sacks) PP(into)

- In each lexicalized grammar rule, the lexical head of a non-terminal on the left is the lexical head of one of the constituents on the right.

Probabilistic Lexicalized CFGs

- A lexicalized tree



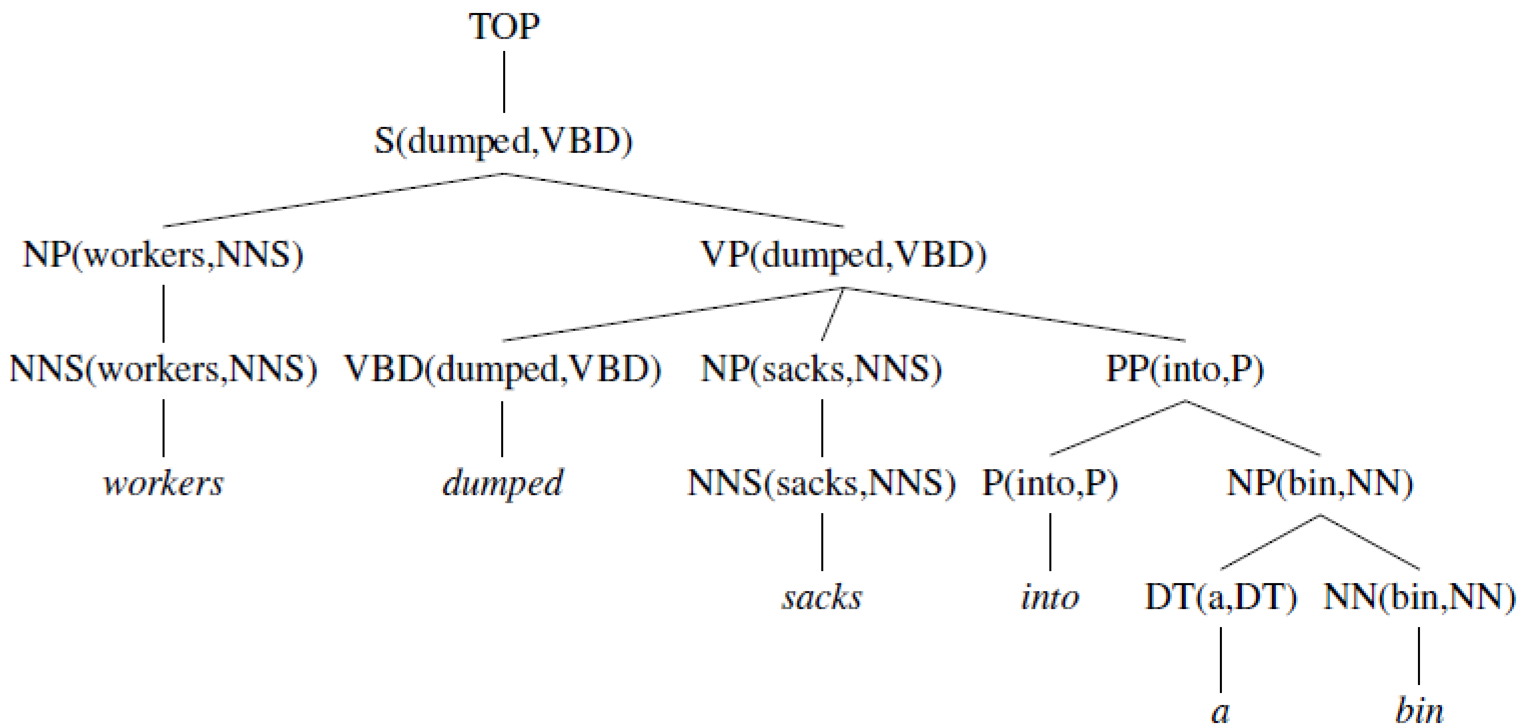
Probabilistic Lexicalized CFGs

- We can also associate non-terminals with **head tags** which are POS tags of their head words.
- Each rule is lexicalized by both the headword and the head tag of each constituent:

VP(dumped,VBD) → VBD(dumped,VBD) NP(sacks,NNS) PP(into,P)

Probabilistic Lexicalized CFGs

A lexicalized tree, including head tags



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

How to find the probabilities?

- In PCFGs, we compute probability of a rule as follows:
  - VP → VBD NP PP
  - P(VP → VBD NP PP | VP) = count(VP → VBD NP PP) / count(VP)
  - That's the count of this rule divided by the number of VPs in a treebank.
- In a lexicalized PCFG, we have to compute probability of a rule as follows:
  - rule: VP(dumped) → VBD(dumped) NP(sacks) PP(into)
  - P(rule | VP(dumped)) = count(rule) / count(VP(dumped))
  - Not likely to have significant counts in any treebank.
- In a lexicalized PCFG with head tags, we have to compute probability as follows:
  - rule: VP(dumped,VBD) → VBD(dumped,VBD) NP(sacks,NNS) PP(into,P)
  - P(rule | VP(dumped,VBD)) = count(rule) / count(VP(dumped,VBD))
  - Not likely to have significant counts in any treebank.

How to find the probabilities?

- When we stuck to compute probabilities directly, we exploit independence assumptions and collect the statistics using these independence assumptions.
  - We can use different independence assumptions. We look at a simple one, but there are more complicated ones (Collins Parser in the book).

Independence assumption: Rules only depend on their head non-terminals.

- In a lexicalized PCFG:
  - rule: VP(dumped) → VBD(dumped) NP(sacks) PP(into)
  - P(rule | VP(dumped)) = count(rule(dumped)) / count(VP(dumped))
  - i.e. How many times this ruled used with dumped, divided by the number of VPs that dumped appears in total.

- In a lexicalized PCFG with head tags:
  - rule: VP(dumped,VBD) → VBD(dumped,VBD) NP(sacks,NNS) PP(into,P)
  - P(rule | VP(dumped,VBD)) = count(rule(dumped,VBD)) / count(VP(dumped,VBD))
  - i.e. How many times this ruled used with dumped,VBD, divided by the number of VPs that dumped,VBD appears in total.

Probabilistic Parsing: Summary

- A **probabilistic context-free grammar (PCFG)** is a context-free grammar in which every rule is annotated with the probability of that rule being chosen.
  - Each PCFG rule is treated as if it were **conditionally independent**; thus, the probability of a sentence is computed by multiplying probabilities of each rule in parse of sentence.

- Raw PCFGs suffer from **poor independence assumptions** among rules and **lack of sensitivity to lexical dependencies**.
  - One way to deal with this problem is to **split** non-terminals.

- **Probabilistic lexicalized CFGs** are another solution to this problem in which the basic PCFG model is augmented with a lexical head for each rule.
  - The probability of a rule can then be conditioned on the lexical head.