9 - Lexicalized and Probabilistic Parsing

CS 6320

Outline

- PP Attachment Problem
- Probabilistic CFG
- Problems with PCFG
- Probabilistic Lexicalized CFG
- The Collins Parser
- Evaluating parsers
- Example

PP-attachment Problem

I buy books for children

I buy (books for children)

Or

I buy (for children) (books)

Semantic selection

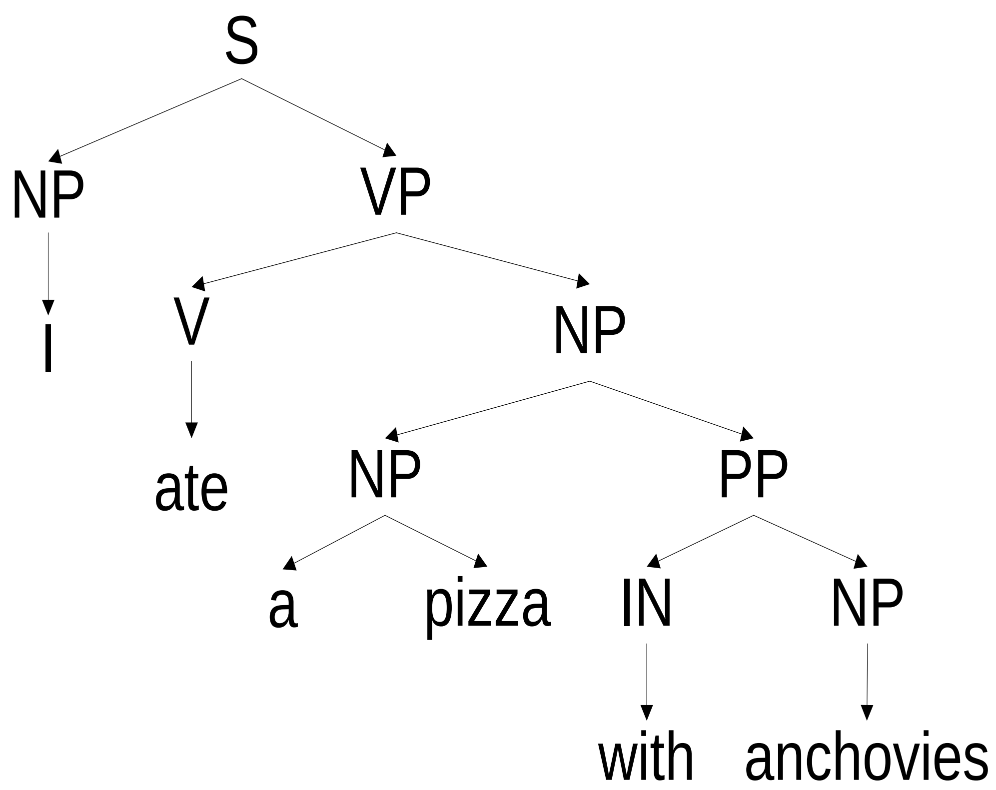
I ate a pizza with anchovies.

I ate a pizza with friends.

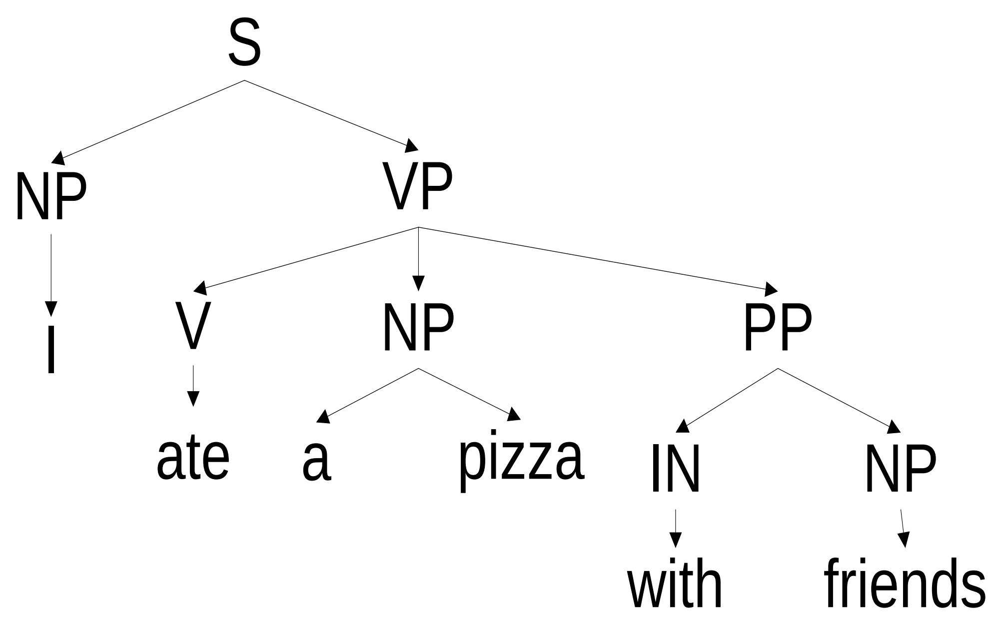
I ate (a pizza with anchovies).

I ate (a pizza) (with friends).

Parse tree for "pizza with anchovies"



Parse tree for "pizza with friends"



More than one PP

I saw the man in the house with the telescope.

I saw (the man (in the house (with the telescope)).

I saw (the man (in the house) (with the telescope))

I saw (the man) (in the house (with the telescope)).

I saw (the man) (in the house) (with the telescope).

Accuracy for "most probable attachment" for a preposition

Prep.	**%of total**	**% V**
of	13.47%	6.17%
to	13.27%	80.14%
in	12.42%	73.64%
for	6.87%	82.44%
on	6.21%	75.51%
with	6.17%	86.30%
from	5.37%	75.90%
at	4.09%	76.63%
as	3.95%	86.51%
by	3.53%	88.02%
into	3.34%	89.52%
that	1.74%	65.97%
about	1.45%	70.85%
over	1.30%	86.83%

Accuracy on UPenn-II Treebank: 81.73%

Semantic information

- Identify the correct meaning of the words
- Use this information to decide which is the most probable parse
- Ex.: *eat pizza with friends*

Pragmatic and discourse information

- To achieve 100% accuracy, we need this kind of information
- Examples
  - *Buy a car with a steering wheel* – you need knowledge about how the cars are made
  - *I saw that car in the picture* – you need also surrounding discourse

[McLauchlan 2001 – Maximum Entropy Models and Prepositional Phrase Ambiguity]

Probabilistic Context-Free Grammars

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 $A \rightarrow \beta\ [p]$,
where A is a non-terminal,
β is a string of symbols from the infinite set of strings $(\Sigma \cup N)*$,
and p is a number between 0 and 1 expressing $P(\beta|A)$

S a designated **start symbol**

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

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

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

$$P(RHS|LHS)$$

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

Probabilistic Context-Free Grammar

PCFG assigns a probability to each parse-tree T

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

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

$$\text{but} \quad P(S \mid T) = 1$$

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

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

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

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

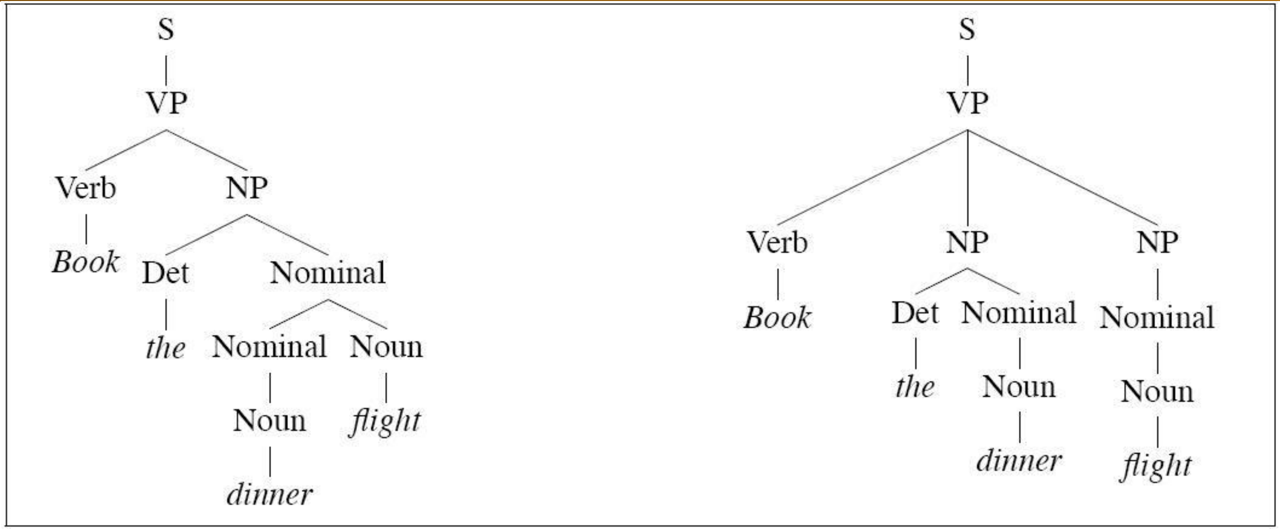
$$\hat{T}(S) = \underset{T s.t. S = \text{yield}(T)}{\arg\max} P(T)$$

Probabilistic Context-Free Grammars

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]	| $meal$ [.15] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flights$ [.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]	

Figure 14.1 A PCFG that is a probabilistic augmentation of the $\mathcal{L}_1$ miniature English CFG grammar and lexicon of Fig 13.1. These probabilities were made up for pedagogical purposes and are not based on a corpus (since any real corpus would have many more rules, so the true probabilities of each rule would be much smaller).

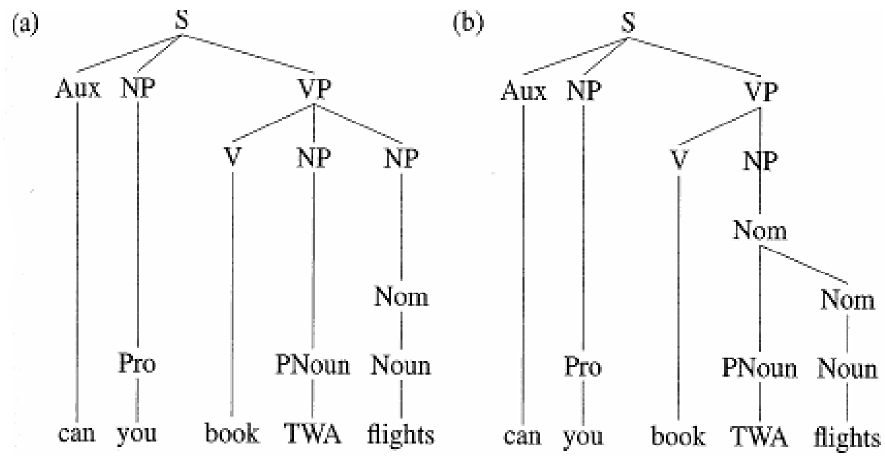
Probabilistic Context-Free Grammars



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

Figure 14.2 Two parse trees for an ambiguous sentence. The transitive parse on the left corresponds to the sensible meaning "Book a flight that serves dinner", while the ditransitive parse on the right corresponds to the nonsensical meaning "Book a flight on behalf of 'the dinner'".

Probabilistic Context Free Grammars



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

Two parse trees for an ambiguous sentence. Parse (a) corresponds to the meaning "Can you book flights on behalf of TWA", parse (b) to "Can you book flights which are run by TWA".

Probabilistic Context Free Grammars

$$
\begin{aligned}
P(T_l) &= .15 * .40 * .05 * .05 * .35 * .75 * \\
&\quad * .40 * .40 * .40 * .30 * \\
&\quad * .40 * .50 \\
&= 1.5 \times 10^{-6} \\
P(T_r) &= .15 * .40 * .40 * .05 * .05 * \\
&\quad .75 * .40 * .40 * .40 * .30 * \\
&\quad .40 * .50 \\
&= 1.7 \times 10^{-6}
\end{aligned}
$$

Note:

$$
\begin{aligned}
P(S) &= \sum_{T \in \tau(S)} P(T, S) \\
&= \sum_{T \in \tau(S)} P(T)
\end{aligned}
$$

- Probabilities of a sentence is the sum of probabilities of all parse trees.
- Useful for Language Modeling

Probabilistic CKY Parsing

```
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[1, LENGTH(words), S]), table[1, LENGTH(words), S]
```

Figure 14.3 The probabilistic CKY algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky normal form. *back* is an array of backpointers used to recover the best parse. The *build_tree* function is left as an exercise to the reader.

Probabilistic CKY Parsing

S	$\rightarrow$	$NP\ VP$	.80	Det	$\rightarrow$	the	.40
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

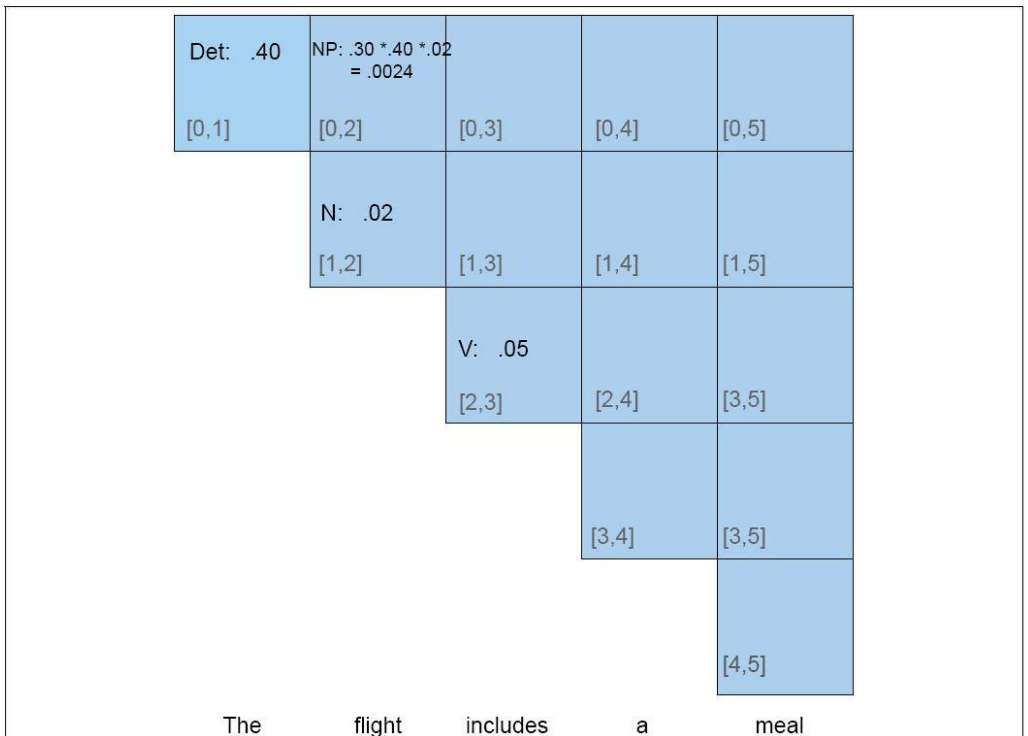


Figure 14.4 The beginning of the probabilistic CKY matrix. Filling out the rest of the chart is left as Exercise 14.4 for the reader.

Learning PCFG Probabilities

Treebank contains parse trees for a large corpus

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

Problems with PCFG and Enhancement

1. Assumption that production probabilities are independent does not hold.

   Often the choice of how a node expands depends on the location of that node in the parse tree.

   Ex: syntactic subjects are often realized with pronouns, whereas direct objects use more non-pronominal noun-phrases.

   NP → Pronoun

   NP → Det Noun

Problems with PCFG and Enhancement

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

$NP \rightarrow DT\ NN\ .28$

$NP \rightarrow PRP\ .25$

would be erroneous

Problems with PCFG and Enhancement

2.1 PCFG are insensitive to the words they expand;
In reality the lexical information about words plays an important role in selecting correct parse trees.

(a) I ate pizza with anchovies.

(b) I ate pizza with friends.

In (a) $NP \rightarrow NP\ PP$ (NP attachment)

(b) $VP \rightarrow NP\ PP$ (VP attachment)

PP attachment depends on the semantics of PP head noun.

Problems with PCFG and Enhancement

2.2 Lexical preference of verbs (Subcategorization)

Moscow sent more than 100,000 soldiers into Afghanistan.

NP *into Afghanistan* attaches to *sent* not to *soldiers*.

This is because the verb *send* subcategorizes for destination, expressed by preposition *into*.

Problems with PCFG and Enhancement

2.3 Coordination Ambiguities

(a) (*dogs in houses*) and (*cats*)

(b) *dogs in* (*houses and cats*)

(a) is preferred because dogs and cats are semantic siblings - i.e., animals.

Coordination Ambiguities

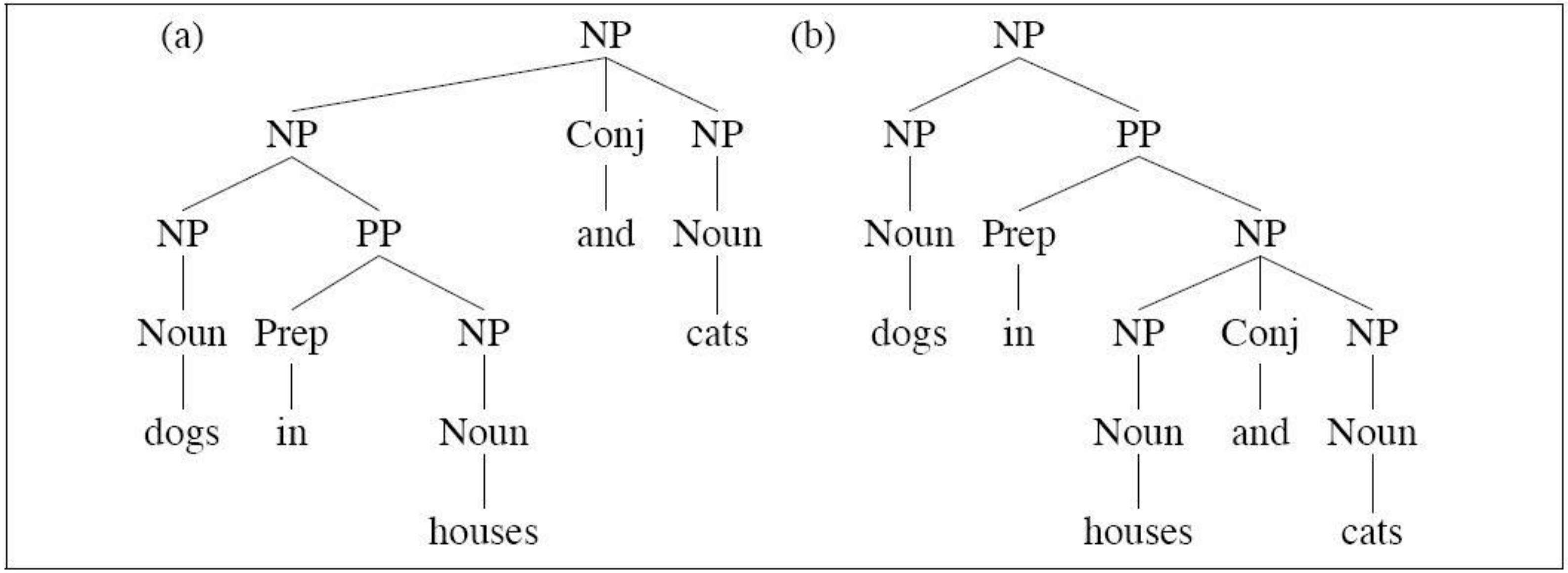


Figure 14.7 An instance of coordination ambiguity. Although the left structure is intuitively the correct one, a PCFG will assign them identical probabilities since both structures use exactly the same rules. After Collins (1999).

Probabilistic Lexicalized CFG

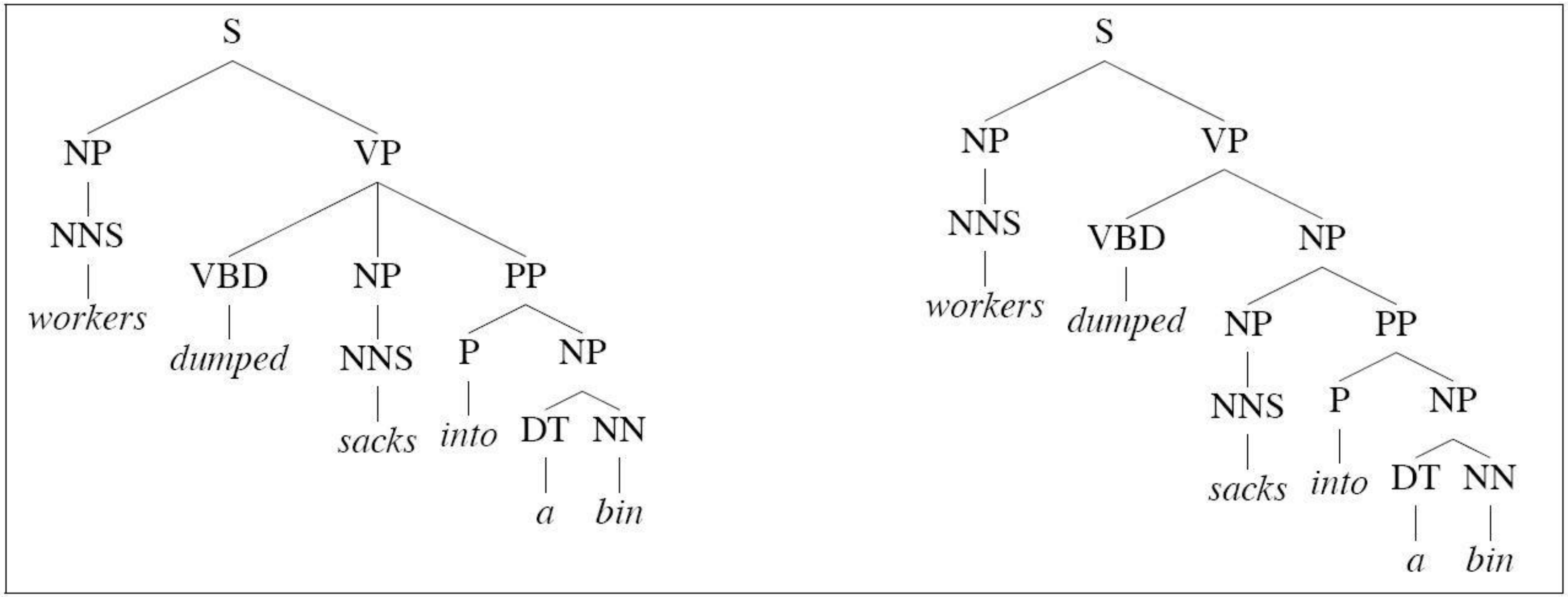


Figure 14.5 Two possible parse trees for a **prepositional phrase attachment ambiguity.** The left parse is the sensible one, in which "into a bin" describes the resulting location of the sacks. In the right incorrect parse, the sacks to be dumped are the ones which are already "into a bin", whatever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

Probabilistic Lexicalized CFG

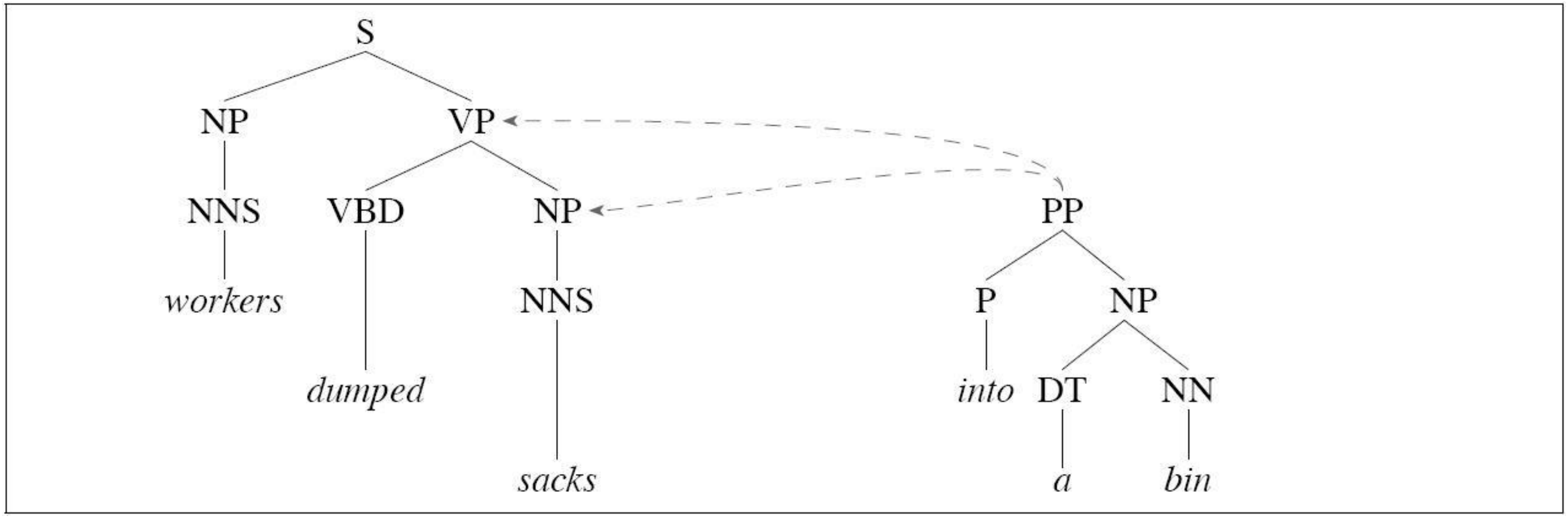
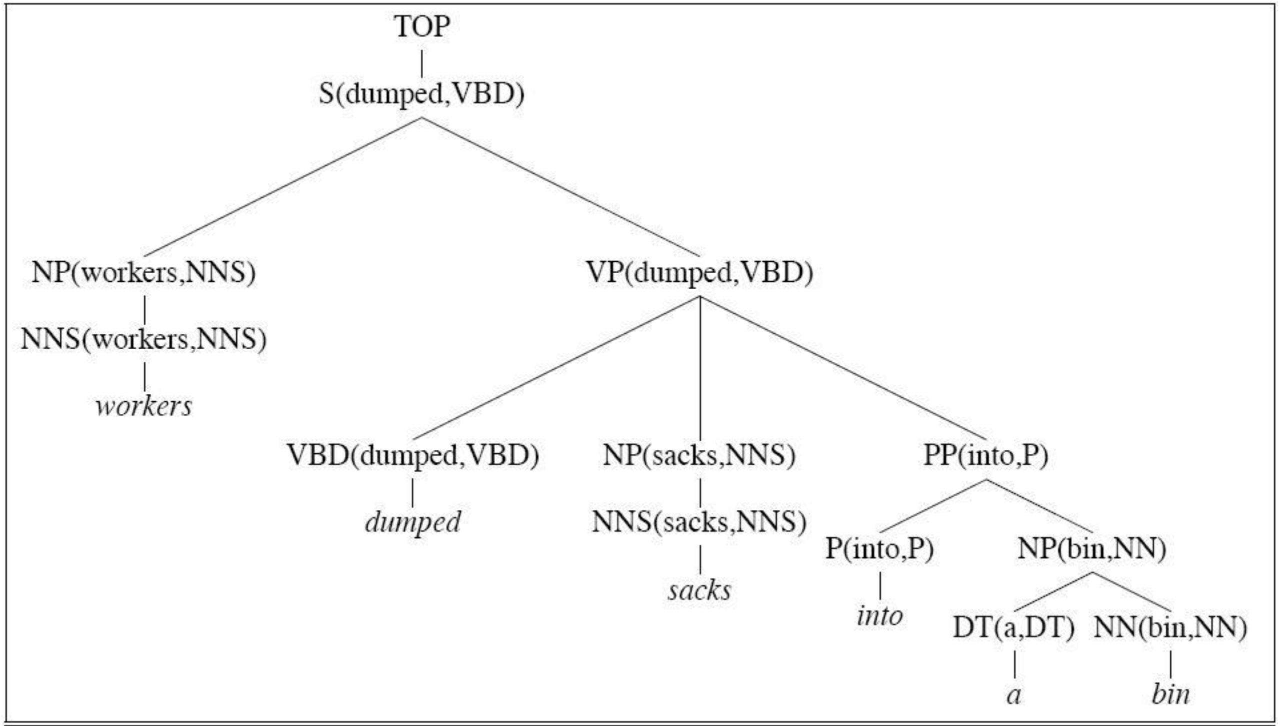


Figure 14.6 Another view of the preposition attachment problem. Should the *PP* on the right attach to the *VP* or *NP* nodes of the partial parse tree on the left?

Probabilistic Lexicalized CFG



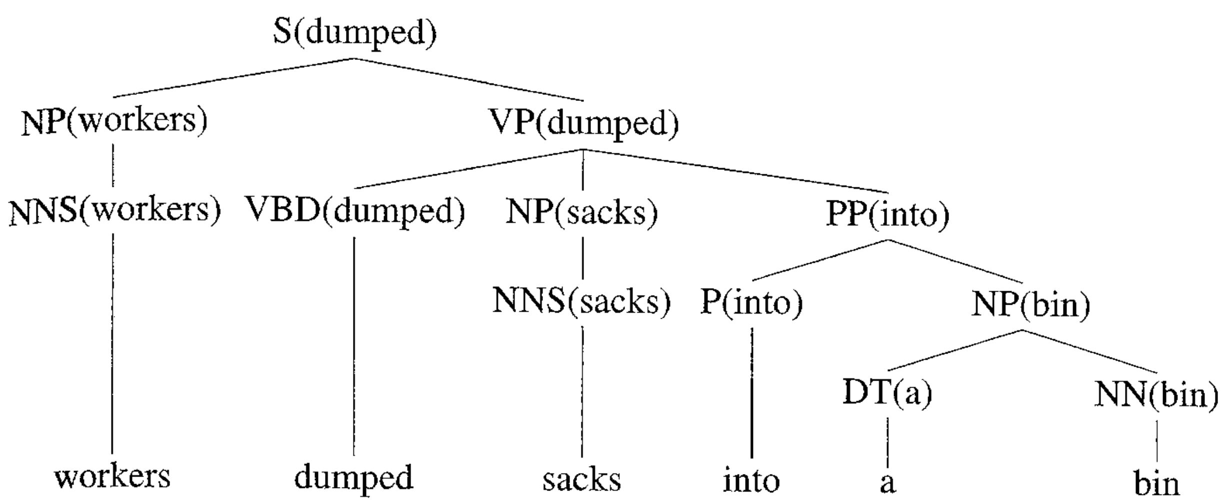
Internal Rules					**Lexical Rules**		
TOP	→	S(dumped,VBD)			NNS(workers,NNS)	→	workers
S(dumped,VBD)	→	NP(workers,NNS)	VP(dumped,VBD)		VBD(dumped,VBD)	→	dumped
NP(workers,NNS)	→	NNS(workers,NNS)			NNS(sacks,NNS)	→	sacks
VP(dumped,VBD)	→	VBD(dumped, VBD)	NP(sacks,NNS)	PP(into,P)	P(into,P)	→	into
PP(into,P)	→	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 that would be needed for this parse tree, internal rules on the left, and lexical rules on the right.

Probabilistic Lexicalized CFG

- Lexical heads play an important role since the semantics of the head dominates the semantics of that phrase.
- Annotate each non-terminal phrasal node in a parse tree with its lexical head.

"Workers dumped sacks into a bin."



Probabilistic Lexicalized CFG

- A lexicalized grammar shows lexical preferences between heads and their constituents. Probabilities are added to show the likelihood of each rule/head combination.

$$VP(dumped) \rightarrow VBD(dumped)NP(sacks)PP(into)[3\times 10^{-10}]$$

$$VP(dumped) \rightarrow VBD(dumped)NP(cats)PP(into)[8\times 10^{-11}]$$

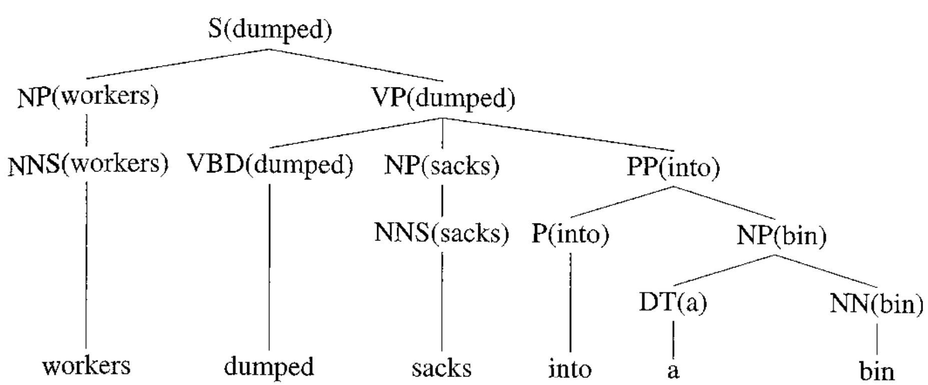
$$VP(dumped) \rightarrow VBD(dumped)NP(hats)PP(into)[4\times 10^{-10}]$$

$$VP(dumped) \rightarrow VBD(dumped)NP(sacks)PP(above)[1\times 10^{-12}]$$

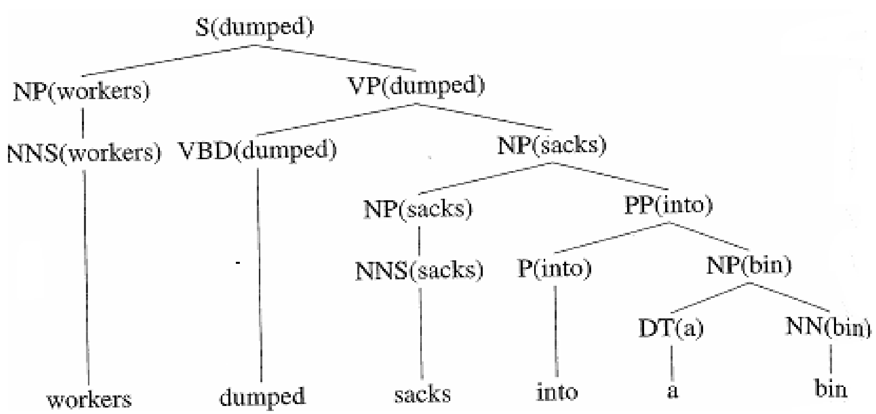
- Since it is not possible to store all possibilities, one solution is to cluster some of the cases based on their semantic category.
  E.g., hats and sacks are inanimated objects.
  E.g., *dumped* prefers preposition *into* over *above.*

Probabilistic Lexicalized CFG

A lexicalized tree from Collins (1999)



An incorrect parse of the sentence from Collins (1999)



Probabilistic Lexicalized CFG

$$p(VP \rightarrow VBD\ NP\ PP \mid VP, dumped)$$

$$= \frac{C(VP(dumped) \rightarrow VBD\ NP\ PP)}{\sum_{\beta} C(VP(dumped) \rightarrow \beta)}$$

$$= \frac{6}{9} = 0.67$$

$$p(VP \rightarrow VBD\ NP \mid VP, dumped)$$

$$= \frac{C(VP(dumped) \rightarrow VBD\ NP)}{\sum_{\beta} C(VP(dumped) \rightarrow \beta)}$$

$$= \frac{0}{9} = 0$$

Probabilistic Lexicalized CFG

- Head probabilities.

The mother head is *dumped* and the head of *PP* is *into*.

$$p(into \mid PP, dumped)$$

$$= \frac{C(X(dumped) \rightarrow \ldots PP(into) \ldots)}{\sum_{\beta} C(X(dumped) \rightarrow \ldots PP \ldots)}$$

$$= \frac{2}{9} = 0.22$$

Probabilistic Lexicalized CFG

The mother head is *sacks* and the head of *PP* is *into*.

$$
\begin{aligned}
& p(into \mid PP, sacks) \\
& = \frac{C(X(sacks) \rightarrow \ldots PP(into) \ldots)}{\sum_{\beta} C(X(sacks) \rightarrow \ldots PP \ldots)} \\
& = \frac{0}{0}
\end{aligned}
$$

Thus the head probabilities predict that *dumped* is more likely to be modified by *into* that is *sacks*.

Probabilistic Lexicalized CFG

- Modern parsers (Charniak, Collins, etc.) make simplifying assumptions about relating the heads of phrases to the heads of their constituents.
- In PCFG the probability of a node n being expanded by a rule r is conditioned only by the syntactic category of node n.
  Idea: add one more conditioning factor: the headword of the node $h(n)$.
  $p(r(n)|n, h(n))$
  is the conditional probability of expanding rule r given the syntactic category of n and lexical information $h(n)$.

$$p(r \mid VP, dumped)$$

r is VP $\rightarrow$ VBD NP PP

Probabilistic Lexicalized CFG

- How to compute the probability of a head?

  Two factors are important:
  - syntactic category of a node
  - neighboring heads

$$p(h(n) = word_i \mid n, h(m(n)))$$

where $h(m(n))$ is the head of the node's mother.

$$p(head(n) = sacks \mid n = NP, h(m(n)) = dumped)$$

is the probability that an *NP* whose mother node is dumped has the head sacks.

This probability captures the depending information between *dumped* and *sacks*.

Probabilistic Lexicalized CFG

- Update the formula for computing the probability of a parse.

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

An example:

Consider an incorrect parse tree for
"Workers dumped sacks into a bin,"
and compare it with the previous correct one.

How to Calculate Probabilities

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

can be estimated as

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

However, this is difficult due to small number of times such a specific rule applies.

Instead, make independence assumptions.

$$P(VP(dumped,VBD) \rightarrow VBD(dumped,VBD)\ \ NP(sacks,NNS)\ \ PP(into,P))$$

Note: Modern statistical parsers differ in which independent assumptions they make.

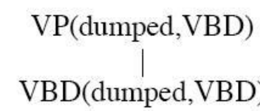
The Collins Parser

$$LHS \rightarrow L_n L_{n-1} \ldots L_1 H R_1 \ldots R_{n-1} R_n$$

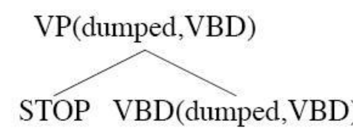
$$P(VP(dumped, VBD) \rightarrow \text{STOP}\ \ VBD(dumped, VBD)\ \ NP(sacks, NNS)\ \ PP(into, P)\text{STOP})$$

---

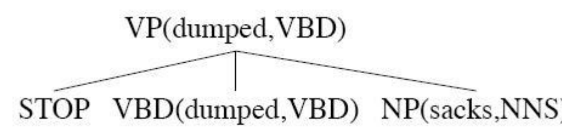
1) Generate the head VBD(dumped,VBD) with probability
P(H|LHS) = P(VBD(dumped,VBD) | VP(dumped,VBD))



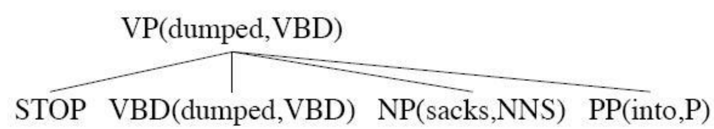
2) Generate the left dependent (which is STOP, since there isn't one) with probability
P(STOP| VP(dumped,VBD) VBD(dumped,VBD))



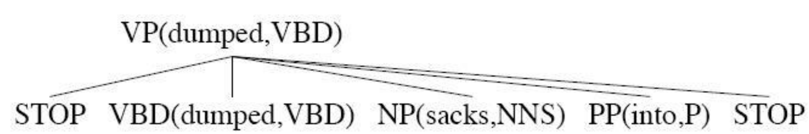
3) Generate right dependent NP(sacks,NNS) with probability
P_r(NP(sacks,NNS| VP(dumped,VBD), VBD(dumped,VBD))



4) Generate the right dependent PP(into,P) with probability
P_r(PP(into,P) | VP(dumped,VBD), VBD(dumped,VBD))



5) Generate the right dependent STOP with probability
P_r(STOP | VP(dumped,VBD), VBD(dumped,VBD))



The Collins Parser

$$P(VP(dumped, VBD) \rightarrow VBD(dumped, VBD)\ \ NP(sacks, NNS)\ \ PP(into, P))$$

$$\begin{aligned} P_H(VBD \mid VP, dumped) \times P_L(STOP \mid VP, VBD, dumped) \\ \times P_R(NP(sacks, NNS) \mid VP, VBD, dumped) \\ \times P_R(PP(into, P) \mid VP, VBD, dumped) \\ \times P_R(STOP \mid VP, VBD, dumped) \end{aligned}$$

$$\frac{\text{Count}(VP(dumped, VBD)\text{ with }NNS(sacks)\text{ as a daughter somewhere on the right})}{\text{Count}(VP(dumped, VBD))}$$

References

- Chart parsing
  - Caraballo, S. and Charniak, E. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics 24* (1998), 275-298
  - Charniak, E., Goldwater, S. and Johnson, M. Edge-based best-first chart parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*. 1998, 127-133
  - Charniak, E. A Maximum-Entropy-Inspired Parser *Proceedings of NAACL* -2000
- Maximum entropy
  - Berger, A.L., Pietra, S.A.D. and Pietra, V.J.D. A maximum entropy approach to natural language processing. *Computational Linguistics 22* 1 (1996), 39-71.
  - Ratnaparkhi, A. Learning to parse natural language with maximum entropy models. *Machine Learning 34* 1/2/3 (1999), 151-176.
- Charniak's parser on web:
  - ftp://ftp.cs.brown.edu/pub/nlparser/