Parameter Estimation and Lexicalization for PCFGs

Informatics 2A: Lecture 20

Mirella Lapata

School of Informatics
University of Edinburgh

04 November 2011

Reading:

J&M 2nd edition, ch. 14.2–14.6.1, NLTK Book, Chapter 8, final section on Weighted Grammar

Parameter Estimation

In a PCFG every rule is associated with a probability.
But where do these rule probabilities come from?

Use a large parsed corpus such as the Penn Treebank.

```
( (S
    (NP-SBJ (DT That) (JJ cold)
      (, ,)
      (JJ empty) (NN sky) )
    (VP (VBD was)
      (ADJP-PRD (JJ full)
        (PP (IN of)
          (NP (NN fire)
            (CC and)
            (NN light) ))))
    (. .) ))
```

$S \rightarrow NP\text{-}SBJ\ VP$
$VP \rightarrow VBD\ ADJP\text{-}PRD$
$PP \rightarrow IN\ NP$
$NP \rightarrow NN\ CC\ NN$

Parameter Estimation

In a PCFG every rule is associated with a probability.
But where do these rule probabilities come from?

Use a large parsed corpus such as the Penn Treebank.

- obtain grammar rules by reading them off the trees;
- Number of times LHS $\rightarrow$ RHS occurs in corpus over number of times LHS occurs

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

Parameter Estimation

Corpus of parsed sentences:

'S1: [S [NP grass] [VP grows]]'
'S2: [S [NP grass] [VP grows] [AP slowly]]'
'S3: [S [NP grass] [VP grows] [AP fast]]'
'S4: [S [NP bananas] [VP grow]]'

Compute PCFG probabilities:

r	Rule	α	$P(r\|\alpha)$
$r1$	S → NP VP	S	2/4
$r2$	S → NP VP AP	S	2/4
$r3$	NP → grass	NP	3/4
$r4$	NP → bananas	NP	1/4
$r5$	VP → grows	VP	3/4
$r6$	VP → grow	VP	1/4
$r7$	AP → fast	AP	1/2
$r8$	AP → slowly	AP	1/2

Parameter Estimation

With these parameters (rule probabilities), we can now compute the probabilities of the four sentences S1–S4:

$$\begin{aligned} P(S1) &= P(r1|S)P(r3|NP)P(r5|VP) \\ &= 2/4 \cdot 3/4 \cdot 3/4 = 0.28125 \end{aligned}$$

$$\begin{aligned} P(S2) &= P(r2|S)P(r3|NP)P(r5|VP)P(r7|AP) \\ &= 2/4 \cdot 3/4 \cdot 3/4 \cdot 1/2 = 0.140625 \end{aligned}$$

$$\begin{aligned} P(S3) &= P(r2|S)P(r3|NP)P(r5|VP)P(r7|AP) \\ &= 2/4 \cdot 3/4 \cdot 3/4 \cdot 1/2 = 0.140625 \end{aligned}$$

$$\begin{aligned} P(S4) &= P(r1|S)P(r4|NP)P(r6|VP) \\ &= 2/4 \cdot 1/4 \cdot 1/4 = 0.03125 \end{aligned}$$

Parameter Estimation

What if we don't have a treebank but we do have a (non-probabilistic) parser?

1. Take a CFG and set all rules to have equal probability
2. Parse the corpus with the CFG
3. Adjust the probabilities
4. Repeat steps two and three until probabilities converge

This is the **Inside-Outside algorithm** (Baker, 1979), a type of Expectation Maximisation algorithm. It can also be used to induce a grammar, but only with limited success.

Problems with Standard PCFGs

While standard PCFGs are useful for a number of applications, they can produce a wrong result when used to choose the correct parse for an ambiguous sentence.

How can that be?

1. The independence of the rules in a PCFG.
2. They ignore lexical information until the very end of the analysis, when word classes are rewritten to word tokens.

How can this lead to the wrong choice among possible parses?

Problem 1: Assuming Independence

By definition, a CFG assumes that the expansion of non-terminals is completely independent: It doesn't matter:

- where a non-terminal is in the analysis;
- what else is (or isn't) in the analysis.

The same assumption holds for standard PCFGs: The probability of a rule is the same, no matter

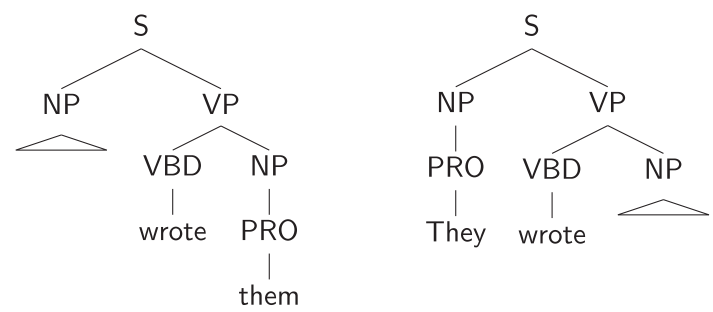
- where it is applied in the analysis;
- what else is (or isn't) in the analysis.

But this assumption is too simple!

Problem 1: Assuming Independence

$S \rightarrow NP\ VP$	$NP \rightarrow PRO$
$VP \rightarrow VBD\ NP$	$NP \rightarrow DT\ NOM$

The above rules assign the same probability to both these trees, because they use the same re-write rules, and probability calculations do not depend on where rules are used.



Problem 1: Assuming Independence

But in speech corpus, 91% of 31021 subject NPs are pronouns:

(1) a. She's able to take her baby to work with her.
b. My wife worked until we had a family.

while only 34% of 7489 object NPs are pronouns:

(2) a. Some laws absolutely prohibit it.
b. It wasn't clear how NL and Mr. Simmons would respond if Georgia Gulf spurns them again.

So the probability of NP → PRO should depend on where in the analysis it applies (e.g., subject or object position).

Problem 2: Ignoring Lexical Information

$S \rightarrow NP\ VP$	$N \rightarrow queen \mid bin$
$NP \rightarrow NNS \mid NN$	$NNS \rightarrow workers \mid sacks \mid cars$
$VP \rightarrow VBD\ NP \mid VBD\ NP\ PP$	$V \rightarrow dumped \mid repaired$
$PP \rightarrow P\ NP$	$DT \rightarrow a \mid the$
$NP \rightarrow DT\ NN$	$\mathrm{P} \rightarrow into \mid of$

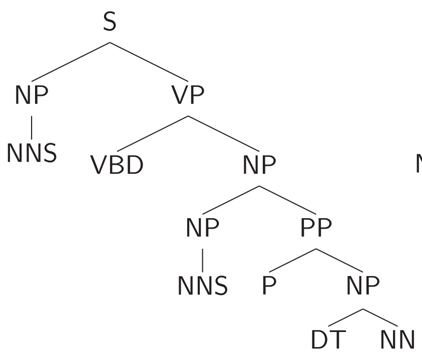
Consider the sentences:

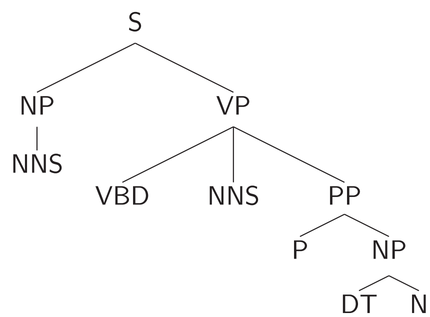
(3) a. Workers dumped sacks into a bin.
b. Workers repaired cars of the queen.

Because rules for rewriting non-terminals ignore word tokens until the very end, let's consider these simply as strings of POS tags:

(4) a. PRO V DT N PREP DT N
b. PRO V DT N PREP DT N

Problem 1: Ignoring Lexical Information





Which do we want for *"Workers dumped sacks into a bin"*? Which for *"Workers repaired cars of the queen"*?

Most appropriate analysis depends, in part, on the actual words.

Lexicalized PCFGs

A PCFG can be lexicalised by associating a word and part-of-speech tag with every non-terminal in the grammar.

It is head-lexicalised if the word is the head of the constituent described by the non-terminal.

Each non-terminal has a head that determines syntactic properties of phrase (e.g., which other phrases it can combine with).

Example

Noun Phrase (NP): Noun
Adjective Phrase (AP): Adjective
Verb Phrase (VP): Verb
Prepositional Phrase (PP): Preposition

Lexicalization

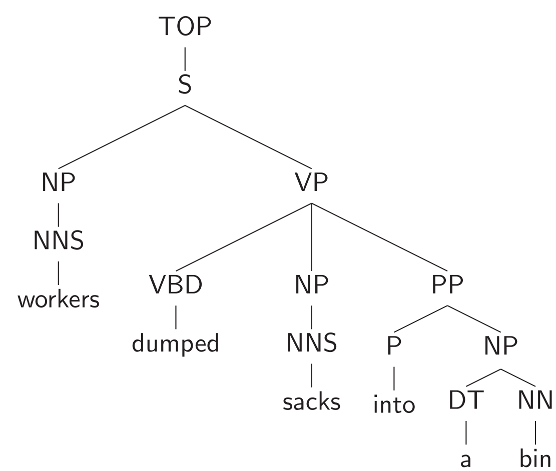
We can lexicalize a PCFG by annotating each non-terminal with its head word, starting with the terminals – replacing

VP	→	VBD NP PP
VP	→	VBD NP
NP	→	DT NN
NP	→	NNS
PP	→	P NP

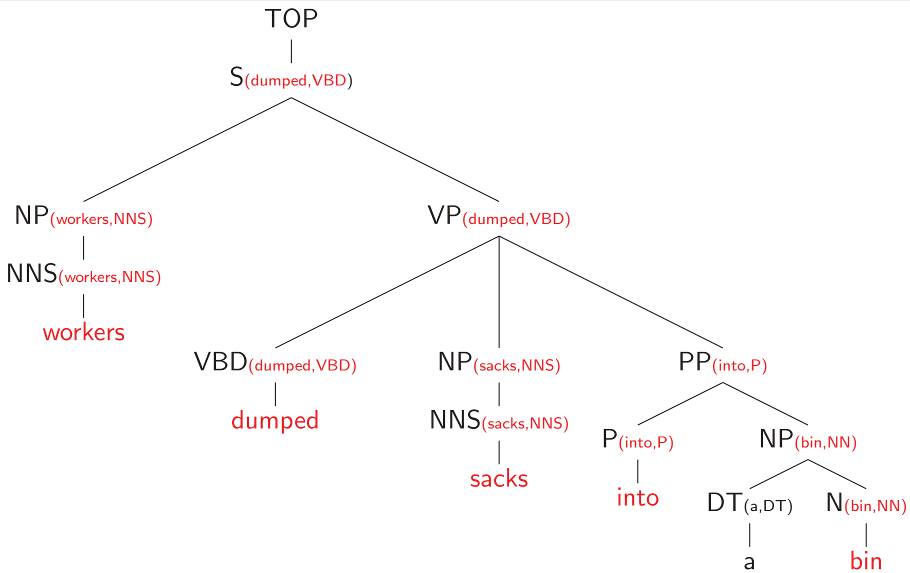
with rules of the form

VP(dumped)	→	V(dumped) NP(sacks) PP(into)
VP(repaired)	→	V(repaired) NP(cars) PP(of)
VP(dumped)	→	V(dumped) NP(sacks)
VP(repaired)	→	V(repaired) NP(cars)
NP(queen)	→	DT(the) NN(queen)
PP(into)	→	P(into) NP(bins)

Lexicalization Example



Lexicalization Example



Head Lexicalization

But this would mean an enormous expansion in grammar rules, with no parsed corpus big enough to estimate their probabilities accurately.

Instead we just lexicalize the head of phrase:

VP(dumped)	→	V(dumped) NP PP
VP(repaired)	→	V(repaired) NP PP
VP(dumped)	→	V(dumped) NP
VP(repaired)	→	V(repaired) NP
NP(queen)	→	DT NN(queen)
PP(of)	→	P(of) NP

Such grammars are called lexicalized PCFGs or, alternatively, probabilistic lexicalized CFGs.

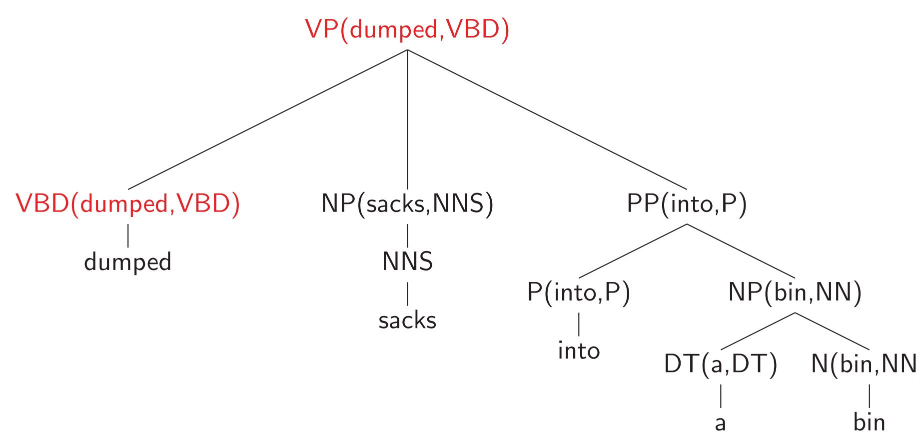
The Collins Parser

Intuition: $LHS \rightarrow L_n L_{n1} \dots L_1 H R_1 \dots R_{n1} R_n$

1. Generate head of the phrase $H(hw, ht)$ with probability $P_h(H(hw, ht)|LHS, hw, ht)$
2. Generate modifiers to the left of head with total probability:
$$\prod_{i=1}^{n+1} P_L(L_i(lw_i, lt_i)|LHS, H, hw, ht)$$
s.t. $L_{n+1}(lw_{n+1}, lt_{n+1}) = STOP$, and we stop generating once weve generated a $STOP$ token.
3. Generate modifiers to the right of head with total probability:
$$\prod_{i=1}^{n+1} P_R(R_i(rw_i, rt_i)|LHS, H, hw, ht)$$
s.t $R_{n+1}(rw_{n+1}, rt_{n+1}) = STOP$, and we stop generating once weve generated a STOP token.

The Collins Parser: Example

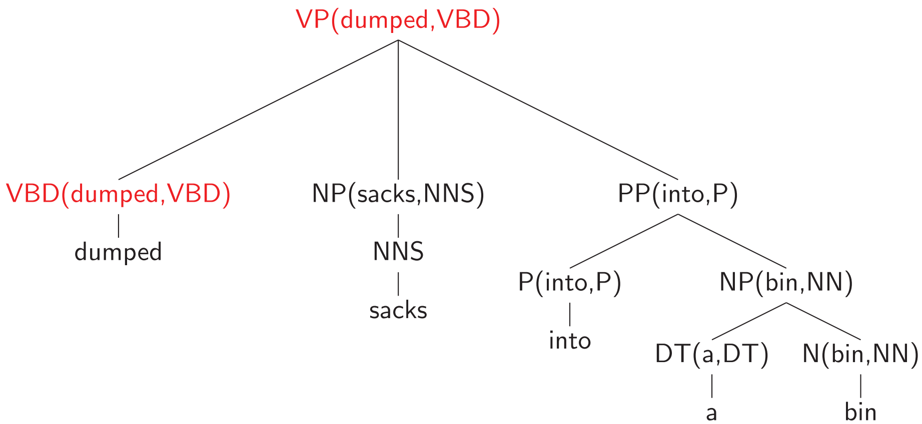
$P(VP(dumped, VBD) \rightarrow$
$STOP\ VBD(dumped, VBD) NP(sacks, NNS) PP(into, P)\ STOP)$



$P(VBD(dumped, VBD)|VP(dumped, VBD))$

The Collins Parser: Example

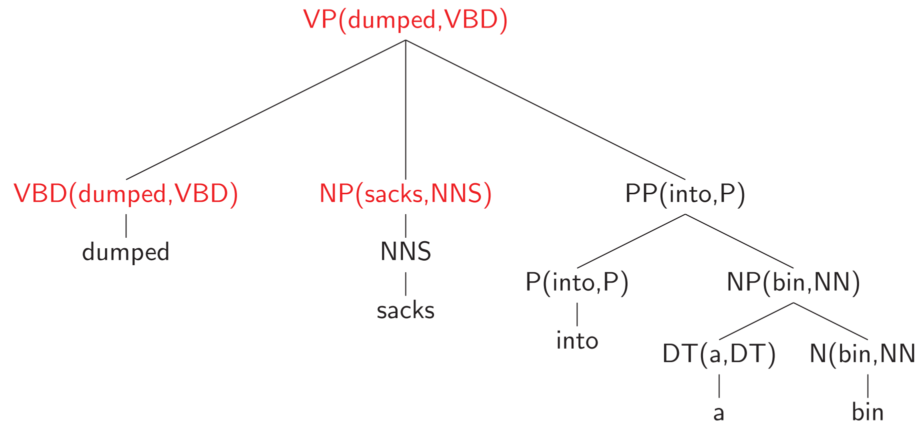
$$P(VP(dumped, VBD) \to STOP\ VBD(dumped, VBD) NP(sacks, NNS) PP(into, P)\ STOP)$$



$$P_L(STOP | VP(dumped, VBD) VBD(dumped, VBD))$$

The Collins Parser: Example

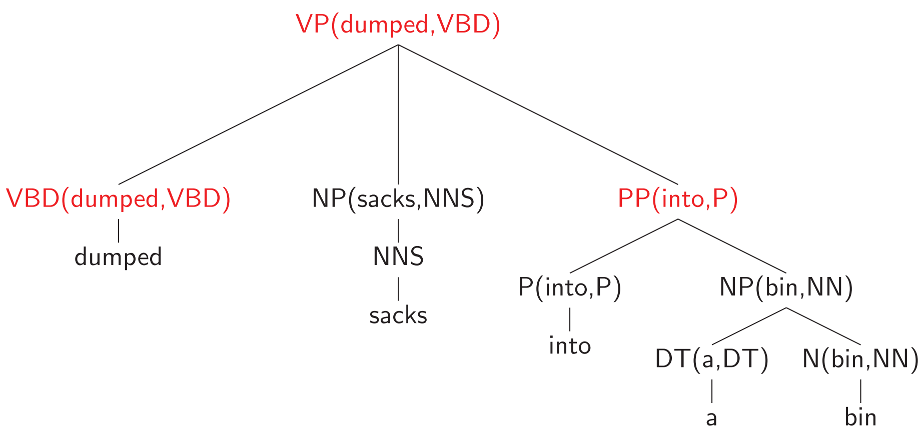
$$P(VP(dumped, VBD) \to STOP\ VBD(dumped, VBD) NP(sacks, NNS) PP(into, P)\ STOP)$$



$$P_R(NP(sacks, NNS | VP(dumped, VBD) VBD(dumped, VBD))$$

The Collins Parser: Example

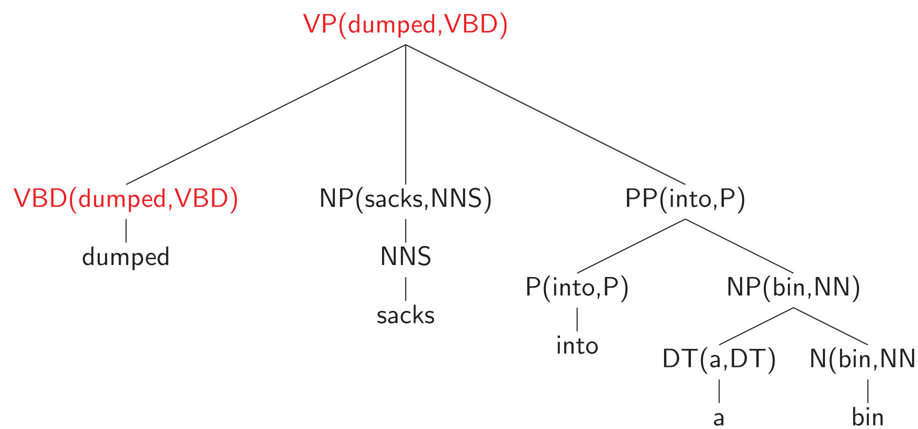
$$P(VP(dumped, VBD) \to STOP\ VBD(dumped, VBD) NP(sacks, NNS) PP(into, P)\ STOP)$$



$$P_R(PP(into, P) | VP(dumped, VBD) VBD(dumped, VBD))$$

The Collins Parser: Example

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



$$P_R(P(STOP) | VP(dumped, VBD) VBD(dumped, VBD))$$

The Collins Parser: Example

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

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

These probabilities can be estimated from smaller amounts of data!

The Collins Parser

- We have just described Model 1.
- A distance function is also included in the conditioning information for the left and right modifiers.
- This is used to measure the number of words between the current modifier and the head.
- It has the effect of preferring right branching structures and dispreferring dependencies which cross a verb.
- Model 2 incorporates verb subcategorisation information.
- Model 3 incorporates long distance dependency information.

Clicker Question

$S \rightarrow NP\ VP$	(1.0)	$NPR \rightarrow John$	(0.5)
$NP \rightarrow DET\ N$	(0.7)	$NPR \rightarrow Mary$	(0.5)
$NP \rightarrow NPR$	(0.3)	$V \rightarrow saw$	(0.4)
$VP \rightarrow V\ PP$	(0.7)	$V \rightarrow loves$	(0.6)
$VP \rightarrow V\ NP$	(0.3)	$DET \rightarrow a$	(1.0)
$PP \rightarrow Prep\ NP$	(1.0)	$N \rightarrow cat$	(0.6)
		$N \rightarrow saw$	(0.4)

What is the probability of the sentence *John saw a saw*?

1. 0.02
2. 0.00016
3. 0.00504
4. 0.0002

Summary

- The rule probabilities of a PCFG can be estimated by counting how often the rules occur in a corpus.
- The usefulness of PCFGs is limited by the lack of lexical information and by strong independence assumptions.
- These limitations can be overcome by lexicalizing the grammars, i.e., by conditioning the rule probabilities on the head word of the rule.
- The Collins parser (Model 1).

Next lecture: Complexity and Character of Human Languages.