

# Parameter Estimation and Lexicalization for PCFGs

Informatics 2A: Lecture 21

Shay Cohen

10 November 2015

## 1 Standard PCFGs

- Parameter Estimation
- Problem 1: Assuming Independence
- Problem 2: Ignoring Lexical Information

## 2 Lexicalized PCFGs

- Lexicalization
- Head Lexicalization

### Reading:

*J&M 2<sup>nd</sup> edition, ch. 14.2–14.6,  
NLTK Book, Chapter 8, final section on Weighted  
Grammar.*

$S \rightarrow NP VP$	(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

# 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) ))))
  (. .) ))
```

$S \rightarrow NP-SBJ VP$   
 $VP \rightarrow VBD ADJP-PRD$   
 $PP \rightarrow IN NP$   
 $NP \rightarrow NN CC NN$   
etc.

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.
- Calculate number of times LHS  $\rightarrow$  RHS occurs 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	$\alpha$	$P(r \alpha)$
$r1$	$S \rightarrow NP VP$	S	2/4
$r2$	$S \rightarrow NP VP AP$	S	2/4

# 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	$\alpha$	$P(r \alpha)$
$r1$	$S \rightarrow NP VP$	S	2/4
$r2$	$S \rightarrow NP VP AP$	S	2/4

# 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	$\alpha$	$P(r \alpha)$
$r1$	$S \rightarrow NP VP$	S	2/4
$r2$	$S \rightarrow NP VP AP$	S	2/4



# 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	$\alpha$	$P(r \alpha)$
$r1$	$S \rightarrow NP VP$	S	2/4
$r2$	$S \rightarrow NP VP AP$	S	2/4
$r3$	$NP \rightarrow grass$	NP	3/4

# 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	$\alpha$	$P(r \alpha)$
$r1$	$S \rightarrow NP VP$	S	2/4
$r2$	$S \rightarrow NP VP AP$	S	2/4
$r3$	$NP \rightarrow grass$	NP	3/4
$r4$	$NP \rightarrow bananas$	NP	1/4

# 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	$\alpha$	$P(r \alpha)$
$r1$	$S \rightarrow NP VP$	S	2/4
$r2$	$S \rightarrow NP VP AP$	S	2/4
$r3$	$NP \rightarrow grass$	NP	3/4
$r4$	$NP \rightarrow bananas$	NP	1/4
$r5$	$VP \rightarrow grows$	VP	3/4

# 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	$\alpha$	$P(r \alpha)$
$r1$	$S \rightarrow NP VP$	S	2/4
$r2$	$S \rightarrow NP VP AP$	S	2/4
$r3$	$NP \rightarrow grass$	NP	3/4
$r4$	$NP \rightarrow bananas$	NP	1/4
$r5$	$VP \rightarrow grows$	VP	3/4
$r6$	$VP \rightarrow grow$	VP	1/4

# 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	$\alpha$	$P(r \alpha)$
$r1$	$S \rightarrow NP VP$	S	2/4
$r2$	$S \rightarrow NP VP AP$	S	2/4
$r3$	$NP \rightarrow grass$	NP	3/4
$r4$	$NP \rightarrow bananas$	NP	1/4
$r5$	$VP \rightarrow grows$	VP	3/4
$r6$	$VP \rightarrow grow$	VP	1/4
$r7$	$AP \rightarrow fast$	AP	1/2

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

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

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

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

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

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

# Motivation behind such estimation

One criterion for finding rule weights of a PCFG (or parameters in general) is the *maximum likelihood* criterion.

It means we want to find rule weights which make the treebank we observe most likely if we multiply in all probabilities together (we assume the trees are independent)

Counting and normalising satisfies this criterion



What if we don't have a treebank, but we do have an unparsed corpus and (non-probabilistic) parser?

- 1 Take a CFG and set all rules to have equal probability.
- 2 Parse the (flat) 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.

While standard PCFGs are already useful for some purposes, they can produce poor result when used for disambiguation.

Why is that?

- 1 They **assume the rule choices are independent of one another.**
- 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 bad choices 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

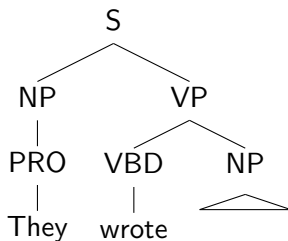
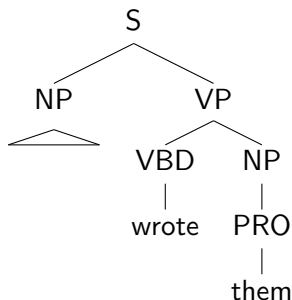
- where it is applied in the analysis;
- what else is (or isn't) in the analysis.

**But this assumption is too simple!**

# Problem 1: Assuming Independence

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

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 corpora, 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.

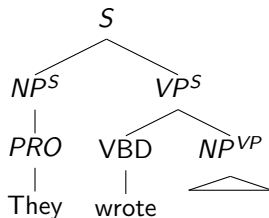
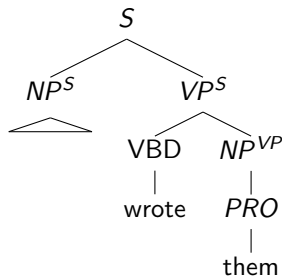
So the probability of NP  $\rightarrow$  PRO should depend on **where** in the analysis it applies (e.g., subject or object position).

## Another example of independence

Question: which tree will get higher probability?

# Addressing the independence problem

One way of introducing greater sensitivity into PCFGs is via **parent annotation**: subdivide (all or some) non-terminal categories according to the non-terminal that appears as the node's immediate parent. E.g.  $NP$  subdivides into  $NP^S$ ,  $NP^{VP}$ , ...

$$S \rightarrow NP^S VP^S$$
$$VP^S \rightarrow VBD^{VP} NP^{VP}$$
$$NP^S \rightarrow PRO$$
$$NP^{VP} \rightarrow PRO, \text{ etc.}$$


# Addressing the independence problem

Node-splitting via **parent annotation** allows different probabilities to be assigned e.g. to the rules

$$NP^S \rightarrow PRO, \quad NP^{VP} \rightarrow PRO$$

However, too much node-splitting can mean not enough data to obtain realistic rule probabilities, unless we have an enormous training corpus.

There are even algorithms that try to identify the optimal amount of node-splitting for a given training set!



## Problem 2: Ignoring Lexical Information

$S \rightarrow NP VP$

$NP \rightarrow NNS \mid NN$

$VP \rightarrow VBD NP \mid VBD NP PP$

$PP \rightarrow P NP$

$NP \rightarrow DT NN$

$N \rightarrow sack \mid bin \mid \dots$

$NNS \rightarrow students$

$V \rightarrow dumped \mid spotted$

$DT \rightarrow a \mid the$

$P \rightarrow in$

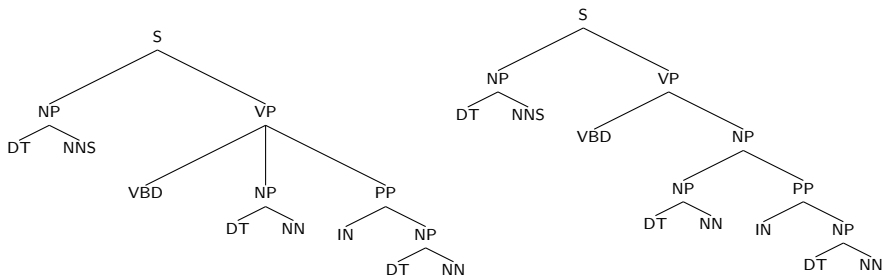
Consider the sentences:

- (3)
- a. The students dumped the sack in the bin.
  - b. The students spotted the flaw in the plan.

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

- (4)     **DT NNS VBD DT NN IN DT NN**

## Problem 2: Ignoring Lexical Information



Which do we want for *The students dumped the sack in the bin*?

Which for *The students spotted the flaw in the plan*?

The most appropriate analysis depends in part on the **actual words** occurring. The word *dumped*, implying motion, is more likely to have an associated prepositional phrase than *spotted*.

A PCFG can be lexicalised by associating a word 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	→	V NP PP	VP	→	V NP
NP	→	DT NN	NP	→	NP PP
NP	→	NNS	PP	→	P NP

with rules such as

VP(dumped)	→	V(dumped) NP(sack) PP(in)
VP(spotted)	→	V(spotted) NP(flav) PP(in)
VP(dumped)	→	V(dumped) NP(sack)
VP(spotted)	→	V(spotted) NP(flav)
NP(flav)	→	DT(the) NN(flav)
PP(in)	→	P(in) NP(bin)
PP(in)	→	P(in) NP(plan)

# Head Lexicalization

In principle, each of these rules can now have its own probability. But that would mean a ridiculous expansion in the set of grammar rules, with no parsed corpus large enough to estimate their probabilities accurately.

Instead we just lexicalize the **head** of phrase:

VP(dumped)	→	V(dumped) NP PP
VP(spotted)	→	V(spotted) NP PP
VP(dumped)	→	V(dumped) NP
VP(spotted)	→	V(spotted) NP
NP(flaw)	→	DT NN(flaw)
PP(in)	→	P(in) NP

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

For lexicalized PCFGs, rule probabilities can be reasonably estimated from a corpus.

In the simplest version, the lexicalized rules are supplemented by **head selection rules**, whose probabilities can also be estimated from a corpus:

VP	→	VP(dumped)
VP	→	VP(spotted)
NP	→	NP(sack)
NP	→	NP(flaw)
PP	→	PP(in)

# How many parameters in a lexicalized PCFG?

When all phrases are annotated with head words (say the grammar is in Chomsky normal form, and we have a vocabulary of size  $V$  and  $N$  nonterminals)?

When only the head phrase is annotated with a head word?

# Combining approaches

We can also combine the ideas of **head lexicalization** with **parent annotation**, leading to rules like

$$\begin{array}{ll} NP^{VP(dumped)} & \rightarrow NP(sack)^{VP(dumped)} \\ NP^{VP(spotted)} & \rightarrow NP(flaw)^{VP(spotted)} \\ PP^{VP(dumped)} & \rightarrow PP(in)^{VP(dumped)} \end{array}$$

The probabilities for such rules can be used to take account of commonly occurring **word combinations**, e.g. of verb-object or verb-preposition. These include **long-distance** correlations invisible to **N-gram** technology.

Grammars with these doubly-lexicalized rules are still feasible, given enough training data. This is roughly the idea behind the **Collins parser**.



# A highly simplified version of Collins model 1

Let a rule be  $\text{LHS} \rightarrow L_n L_{n-1} \dots L_1 H R_1 \dots R_{n-1} R_n$

Idea: break it into smaller probabilities, “generating” the head and then left dependents and right dependents:

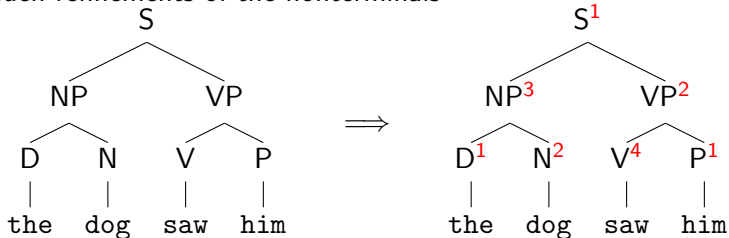
$$\begin{aligned} p(H|\text{LHS}) &\times p_L(L_1|H, \text{LHS}) \dots p_L(L_n|H, \text{LHS}) \times p_L(\text{STOP} \mid L_n, \text{LHS}) \\ &\times p_R(R_1|H, \text{LHS}) \dots p_R(R_n|H, \text{LHS}) \times p_R(\text{STOP} \mid R_n, \text{LHS}) \end{aligned}$$

Now can estimate each term separately, and the sparsity issue will be less severe

# Hidden State Grammars

Lexicalization and node-splitting require careful thought about the linguistic meaning of the splitting

We can try to automatically induce these splittings by introducing *hidden* refinements of the nonterminals



Now our PCFG looks like:

$$S[1] \rightarrow NP[2] VP[3]$$
$$VP[4] \rightarrow VBD[1] NP[3]$$
$$NP[1] \rightarrow PRO[2]$$
$$NP[2] \rightarrow PRO[1], \text{ etc.}$$

**Question:** What kind of normalization do we expect now (say we have  $m$  hidden states)?

# How to Estimate Hidden State Grammars?

Similar to the inside-outside algorithm:

- Start with equal probabilities for all rules with hidden states
- Parse the treebank such that for each sentence the correct tree has to be returned, while maximising the probability of the states

$$H = \arg \max p(H, T, W)$$

( $W$  is the sentence,  $T$  is the treebank sentence, and  $H$  is a set of hidden states associated with each node in  $T$ )

- Re-estimate the hidden state PCFG rules as if you have node splitting on the nonterminals dictated by  $H$

Experiments on the Penn treebank show that approximately 30-60 states are needed for getting good coverage

- 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.

**Demo: the Stanford parser:**

<http://nlp.stanford.edu:8080/parser/>