# Parameter Estimation and Lexicalisation for PCFGs

## Informatics 2A: Lecture 23

Shay Cohen

9 November 2018

# Last Class

- Probabilistic CFGs attach to each rule in the grammar a probability
- The CYK algorithm can be turned probabilistic: we have a chart with three indices ranging over nonterminals, beginning index and end index
- Each such element in the chart has the maximal probability of generating a tree spanning the corresponding phrase headed by that nonterminal

But where do the probabilities come from?

What forms of grammar can we have?

# Recursive description of probabilistic CYK

Call $\mathrm{Chart}[A, i, j]$ the probability of the highest-probability derivation of $w_{i+1}...w_j$ from $A$.

Definition of the CYK algorithm:

$$\mathrm{Chart}[A, i, i+i] = p(A \to w_{i+1})$$

$$\mathrm{Chart}[A, i, j] = \max_{\{k: i<k<j\}} \max_{\{B,C: A \to B\ C \in G\}} \mathrm{Chart}[B, i, k] \times \mathrm{Chart}[C, k, j] \times p(A \to B\ C)$$

## Standard PCFGs

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

## Lexicalized PCFGs

- Lexicalization
- Head Lexicalization

Reading:

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

# 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

# Where the Probabilities Come From?

The case of hidden Markov models:

I/PRP was/VBD walking/VBG down/IN the/DT high/JJ street/NN yesterday/NN when/CC I/PRP noticed/VBD an/DT old/JJ man/NN acting/VBG suspiciously/RB . He/PRP was/VBD peering/VBG into/IN various/JJ shop/NN windows/NNS and/CC writing/VBG things/NNS in/IN a/DT notebook/NN . When/WRB he/PRP spotted/VBD me/PRP, he/PRP stuffed/VBD the/DT notebook/NN into/IN his/PRP$ pocket/NN and/CC wandered/VBD off/RP ./.

- Count the number of times word $w$ occurs with tag $t$.

$$p(w \mid t) = \mathrm{count}(w, t) / \sum_{w'} \mathrm{count}(w', t)$$

- Count the number of times tag $t$ appears after tag $t'$.

$$p(t \mid t') = \mathrm{count}(t', t) / \sum_{t''} \mathrm{count}(t', t'')$$

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

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

# 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.
- 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

Compute PCFG probabilities:

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]]'

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

# Parameter Estimation

Compute PCFG probabilities:

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]]'

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

# Parameter Estimation

Compute PCFG probabilities:

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]]'

| $r$ | 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

Compute PCFG probabilities:

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]]'

| $r$ | 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 | $1/2$ |

# Parameter Estimation

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

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

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

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

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

# 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

# Parameter Estimation: Intuition

Suppose that we have a bag containing two types of marbles: red and black. How would you estimate the ratio of red to black marbles in the bag?

More precisely, what is $p(red)$? (Note: $p(black) = 1 - p(red)$).

# Parameter Estimation: Intuition

Suppose that we have a bag containing two types of marbles: red and black. How would you estimate the ratio of red to black marbles in the bag?

More precisely, what is $p(red)$? (Note: $p(black) = 1 - p(red)$).

**Experiment.** Draw ten marbles from the bag (replacing them each time). Suppose you draw 7 red and 3 black marbles. What is $p(red)$?

# Parameter Estimation: Intuition

Suppose that we have a bag containing two types of marbles: red and black. How would you estimate the ratio of red to black marbles in the bag?

More precisely, what is $p(red)$? (Note: $p(black) = 1 - p(red)$).

**Experiment.** Draw ten marbles from the bag (replacing them each time). Suppose you draw 7 red and 3 black marbles. What is $p(red)$?

1. .3
2. .5
3. .7
4. 1

# Parameter Estimation: Intuition

Suppose that we have a bag containing two types of marbles: red and black. How would you estimate the ratio of red to black marbles in the bag?

More precisely, what is $p(red)$? (Note: $p(black) = 1 - p(red)$).

**Experiment.** Draw ten marbles from the bag (replacing them each time). Suppose you draw 7 red and 3 black marbles. What is $p(red)$?

1. .3
2. .5
3. .7
4. 1

**Why?**

# Parameter Estimation: Maximum Likelihood

Since we saw 7 red and 3 black marbles, we can write the likelihood of the observed data in terms of the unknown parameter $p(red)$:

$$p(data) = p(red)^7 \times (1 - p(red))^3 \tag{1}$$

$p(red)$ is unknown. What's a reasonable way to set it?

# Parameter Estimation: Maximum Likelihood

Since we saw 7 red and 3 black marbles, we can write the likelihood of the observed data in terms of the unknown parameter $p(red)$:

$$p(data) = p(red)^7 \times (1 - p(red))^3 \tag{1}$$

$p(red)$ is unknown. What's a reasonable way to set it?

How about this?

$$\arg \max_{p(red) \in [0,1]} p(data) = p(red)^7 \times (1 - p(red))^3 \tag{2}$$

# Parameter Estimation: Maximum Likelihood

Now we have a basic calculus problem. Solve:

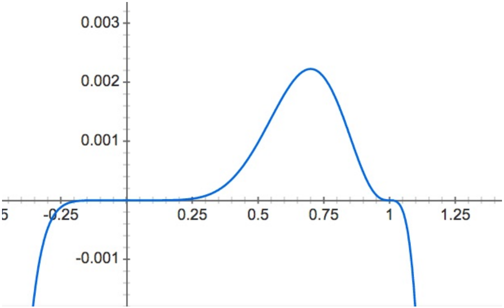
$$\arg\max_{p(red)\in[0,1]} p(data) = p(red)^7 \times (1 - p(red))^3 \tag{3}$$

# Parameter Estimation: Maximum Likelihood

Now we have a basic calculus problem. Solve:

$$\arg\max_{p(red)\in[0,1]} p(data) = p(red)^7 \times (1 - p(red))^3 \qquad (3)$$

What $p(data)$ looks like:



# Maximum Likelihood Estimation

MLE is one of the most basic parameter estimation methods.
When you have lots of data, it's a reasonable first choice.

What are some cases where it might not work?

# Maximum Likelihood Estimation

MLE is one of the most basic parameter estimation methods.
When you have lots of data, it's a reasonable first choice.

What are some cases where it might not work?

**Question.** What if you *don't* have lots of data (for the parameter you want to estimate)?

# Parameter Estimation

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.

# Problems with Standard PCFGs

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

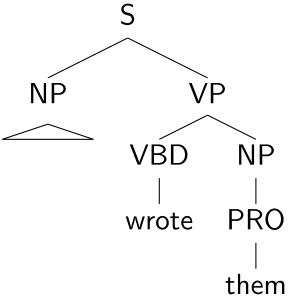
- where it is applied in the analysis;
- what else is (or isn't) in the analysis.

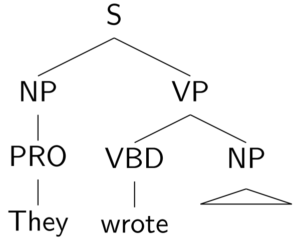
But this assumption is too simple!

# Problem 1: Assuming Independence

$$S \to NP\ VP \qquad NP \to PRO$$
$$VP \to VBD\ NP \qquad NP \to DT\ NOM$$

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

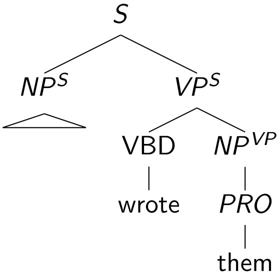
Question: which tree will get higher probability?

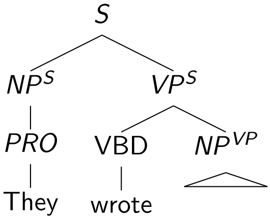
# Addressing the independence problem

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

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

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

$$\begin{array}{ll}
S \rightarrow NP\ VP & N \rightarrow sack \mid bin \mid\ \cdots \\
NP \rightarrow NNS \mid NN & NNS \rightarrow students \\
VP \rightarrow VBD\ NP \mid VBD\ NP\ PP & V \rightarrow dumped \mid spotted \\
PP \rightarrow P\ NP & DT \rightarrow a \mid the \\
NP \rightarrow DT\ NN & \mathrm{P} \rightarrow in
\end{array}$$

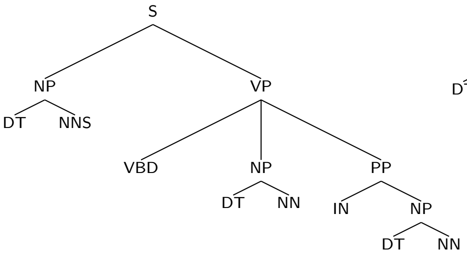
Consider the sentences:

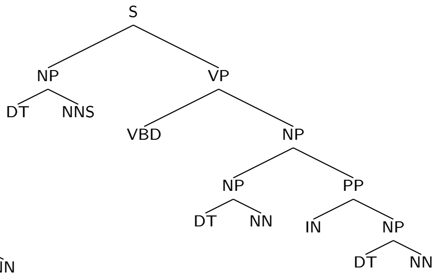
(3) a. The students dumped the sack in the bin.
b. The students spotted the flaw in the plan.

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*.

# Lexicalized PCFGs

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 | $\rightarrow$ | V NP PP | VP | $\rightarrow$ | V NP |
| NP | $\rightarrow$ | DT NN | NP | $\rightarrow$ | NP PP |
| NP | $\rightarrow$ | NNS | PP | $\rightarrow$ | P NP |

with rules such as

| | | |
|---|---|---|
| VP(dumped) | $\rightarrow$ | V(dumped) NP(sack) PP(in) |
| VP(spotted) | $\rightarrow$ | V(spotted) NP(flaw) PP(in) |
| VP(dumped) | $\rightarrow$ | V(dumped) NP(sack) |
| VP(spotted) | $\rightarrow$ | V(spotted) NP(flaw) |
| NP(flaw) | $\rightarrow$ | DT(the) NN(flaw) |
| PP(in) | $\rightarrow$ | P(in) NP(bin) |
| PP(in) | $\rightarrow$ | 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) | $\rightarrow$ | V(dumped) NP PP |
| VP(spotted) | $\rightarrow$ | V(spotted) NP PP |
| VP(dumped) | $\rightarrow$ | V(dumped) NP |
| VP(spotted) | $\rightarrow$ | V(spotted) NP |
| NP(flaw) | $\rightarrow$ | DT NN(flaw) |
| PP(in) | $\rightarrow$ | P(in) NP |

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

# Head Lexicalization

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:

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

# 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

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

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.

# 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.

**Demo: the Stanford parser:**
`http://nlp.stanford.edu:8080/parser/`