Natural Language Processing

Lecture 15:
Treebanks and Probabilistic CFGs

TREEBANKS: A (RE)INTRODUCTION

Two Ways to Encode a Grammar

- Explicitly
  - As a collection of context-free rules
  - Written by hand or learned automatically
- Implicitly
  - As a collection of sentences parsed into trees
  - Probably generated automatically, then corrected by linguists
- Both ways involve a lot of work and impose a heavy cognitive load
- This lecture is about the second option: treebanks (plus the PCFGs you can learn from them)

The Penn Treebank (PTB)

- The first big treebank, still widely used
- Consists of the Brown Corpus, ATIS (Air Travel Information Service corpus), Switchboard Corpus, and a corpus drawn from the *Wall Street Journal*
- Produced at University of Pennsylvania (thus the name)
- About 1 million words
- About 17,500 distinct rule types
  - PTB rules tend to be "flat"—lots of symbols on the RHS
  - Many of the rules types only occur in one tree

Digression: Other Treebanks

- PTB is just one, very important, treebank
- There are many others, though…
  - They are often much smaller
  - They are often dependency treebanks
- However, there are plenty of constituency/phrase structure tree banks in addition to PTB

Digression: Other Treebanks

- Google universal dependencies
  - Internally consistent (if somewhat counter-intuitive) set of universal dependency relations
  - Used to construct a large body of treebanks in various languages
  - Useful for cross-lingual training (since the PoS and dependency labels are the same, cross-linguistically)
  - **Not** immediately applicable to what we are going to talk about next, since it's relatively hard to learn constituency information from dependency trees
  - **Very relevant** to training dependency parsers

Context-Free Grammars

- Vocabulary of terminal symbols, Σ
- Set of nonterminal symbols, N
- Special start symbol $S \in N$
- Production rules of the form $X \rightarrow \alpha$

  where

  $X \in N$

  $\alpha \in (N \cup \Sigma)^*$ (in CNF: $\alpha \in N^2 \cup \Sigma$)

Treebank Tree Example

```
( (S
    (NP-SBJ
      (NP (NNP Pierre) (NNP Vinken) )
      (, ,)
      (ADJP
        (NP (CD 61) (NNS years) )
        (JJ old) )
      (, ,) )
    (VP (MD will)
      (VP (VB join)
        (NP (DT the) (NN board) )
        (PP-CLR (IN as)
          (NP (DT a) (JJ nonexecutive) (NN director) ) )
        (NP-TMP (NNP Nov.) (CD 29) ) ) )
    (. .) ) )
```

PROPER AMBIVALENCE TOWARD TREEBANKS

Proper Ambivalence

- Why you should have great respect for treebanks.
- Why you should be cautious around treebanks.

The Making of a Treebank

- Develop initial coding manual (hundreds of pages long)
  - Linguists define categories and tests
  - Try to foresee as many complications as possible
- Develop annotation tools (annotation UI, pre-parser)
- Collect data (corpora)
  - Composition depends on the purpose of the corpus
  - Must also be pre-processed
- Automatically parse the corpus/corpora
- Train annotators ("coders")
- Manually correct the automatic annotations ("code")
  - Generally done by non-experts under the direction of linguists
  - When cases are encountered that are not in the coding manual…
    - Revise the coding manual to include them
    - Check that already-annotated sections of the corpus are consistent with the new standard

This is expensive and time-consuming!

Why You Should Respect Treebanks

- They require great skill
  - Expert linguists make thousands of decisions
  - Many annotators must all remember all of the decisions and use them consistently, including knowing which decision to use
  - The “coding manual” containing all of the decisions is hundreds of pages long
- They take many years to make
  - Writing the coding manual, training coders, building user-interface tools, ...
  - and the coding itself with quality management
- They are expensive
  - Somebody had to secure the funding for these projects

Why You Should be Cautious Around Treebanks

- They are too big to fail
  - Because they are so expensive, they cannot be replaced easily
  - They have a long life span, not because they are perfect, but because nobody can afford to replace them
- They are produced under pressure of time and funding
- Although most of the decisions are made by experts, most of the coding is done by non-experts

Why It Is Important for You to Invest Some Time to Understand Treebanks

- To create a good model you should understand what you are modeling
- In machine learning improvement in the state of the art comes from:
  - improvement in the training data
  - improvement in the models
- To be a good NLP scientist, you should know when the model is at fault and when the data is at fault
- I will go out on a limb and claim that 90% of NLP researchers do not know how to understand the data

WHERE DO PRODUCTION RULES COME FROM?

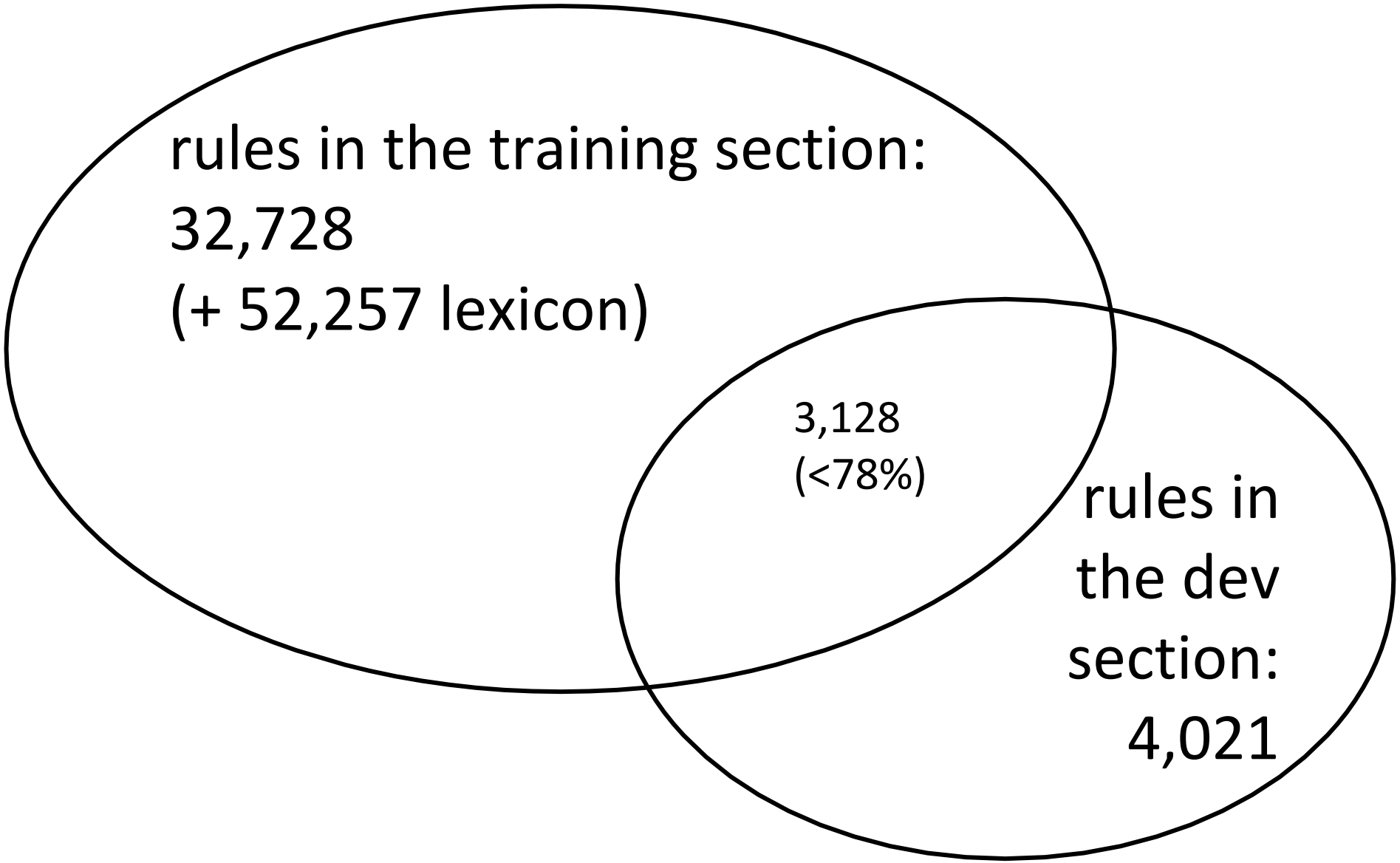
```
( (S
    (NP-SBJ-1
      (NP (NNP Rudolph) (NNP Agnew) )
      (, ,)
      (UCP
        (ADJP
          (NP (CD 55) (NNS years) )
          (JJ old) )
        (CC and)
        (NP
          (NP (JJ former) (NN chairman) )
          (PP (IN of)
            (NP (NNP Consolidated) (NNP Gold) (NNP Fields) (NNP PLC) ) ) ) ) )
      (, ,) )
    (VP (VBD was)
      (VP (VBN named)
        (S
          (NP-SBJ (-NONE- *-1) )
          (NP-PRD
            (NP (DT a) (JJ nonexecutive) (NN director) )
            (PP (IN of)
              (NP (DT this) (JJ British) (JJ industrial) (NN conglomerate) ) ) ) ) ) ) )
    (. .) ) )
```

Some Rules

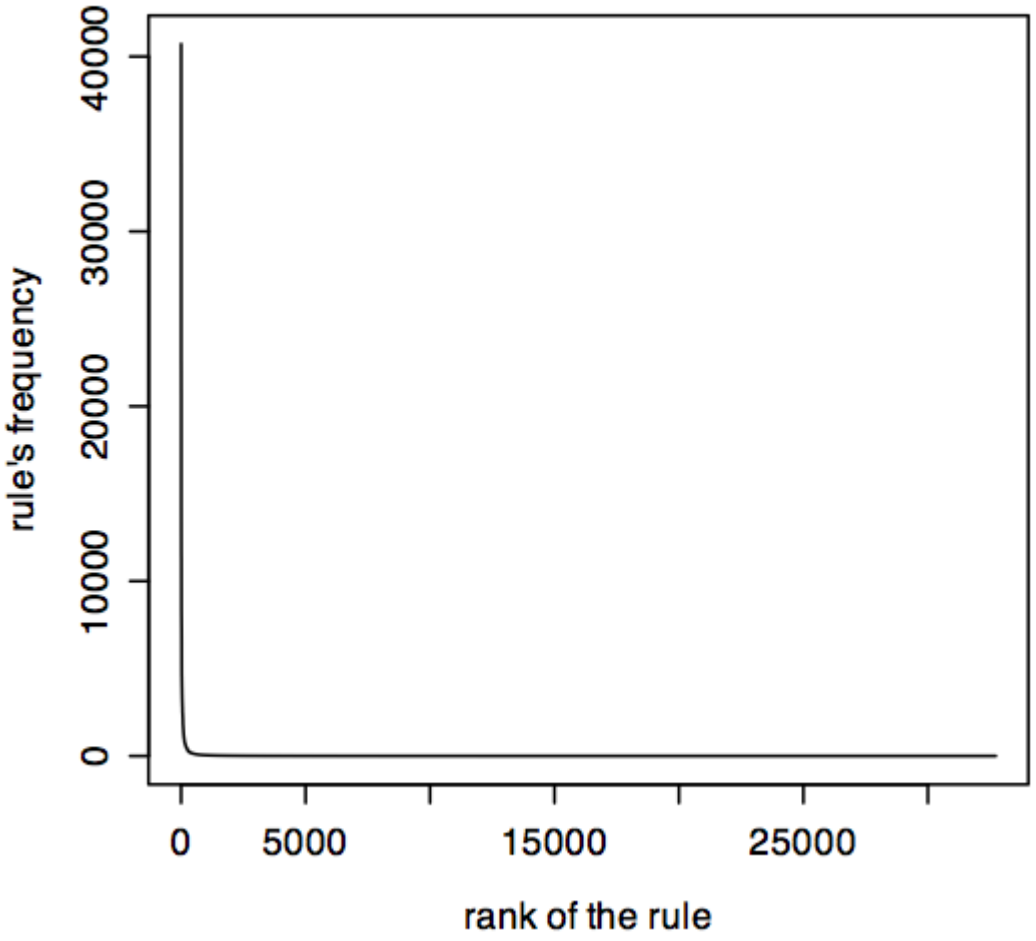
40717 PP → IN NP
33803 S → NP-SBJ VP
22513 NP-SBJ → -NONE-
21877 NP → NP PP
20740 NP → DT NN
14153 S → NP-SBJ VP .
12922 VP → TO VP
11881 PP-LOC → IN NP
11467 NP-SBJ → PRP
11378 NP → -NONE-
11291 NP → NN
...
989 VP → VBG S
985 NP-SBJ → NN
983 PP-MNR → IN NP
983 NP-SBJ → DT
969 VP → VBN VP
...

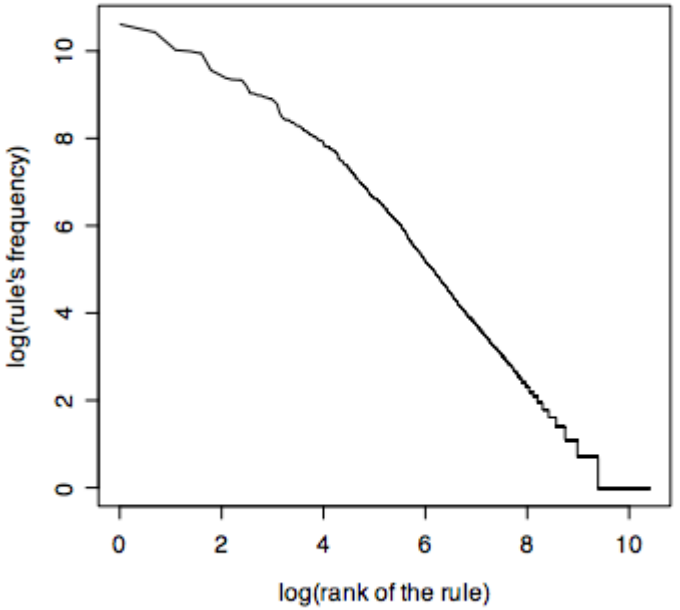
100 VP → VBD PP-PRD
100 PRN → : NP :
100 NP → DT JJS
100 NP-CLR → NN
99 NP-SBJ-1 → DT NNP
98 VP → VBN NP PP-DIR
98 VP → VBD PP-TMP
98 PP-TMP → VBG NP
97 VP → VBD ADVP-TMP VP
...
10 WHNP-1 → WRB JJ
10 VP → VP CC VP PP-TMP
10 VP → VP CC VP ADVP-MNR
10 VP → VBZ S , SBAR-ADV
10 VP → VBZ S ADVP-TMP

Rules in the Treebank



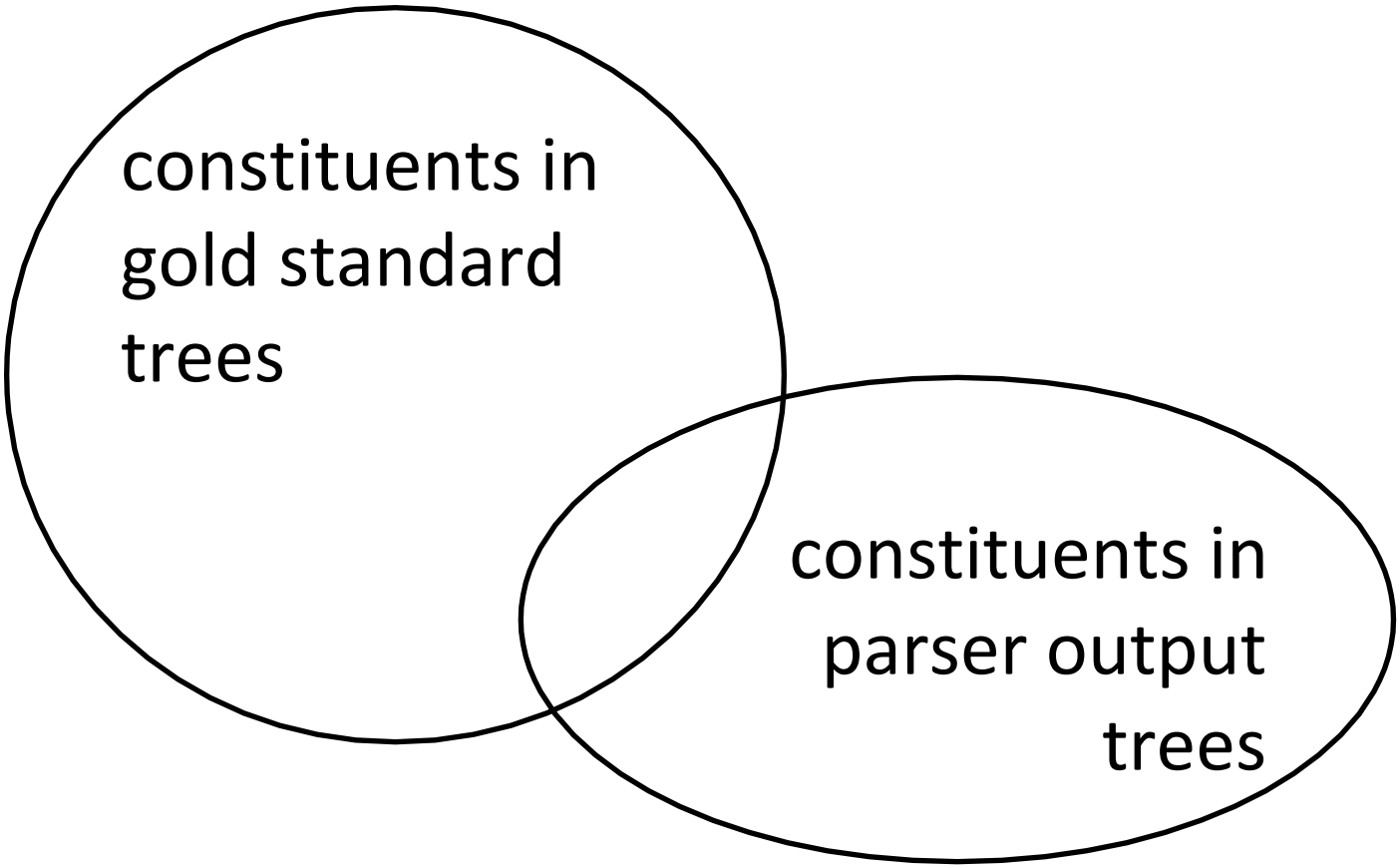
Rule Distribution (Training Set)





EVALUATION OF PARSING

Evaluation for Parsing: Parseval



Parseval

labeled recall: $= \frac{\text{\# of correct constituents in candidate parse of } s}{\text{\# of correct constituents in treebank parse of } s}$

labeled precision: $= \frac{\text{\# of correct constituents in candidate parse of } s}{\text{\# of total constituents in candidate parse of } s}$

cross-brackets: the number of crossed brackets (e.g. the number of constituents for which the treebank has a bracketing such as ((A B) C) but the candidate parse has a bracketing such as (A (B C))).

The F-Measure

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

$$F_1 = \frac{2PR}{P + R}$$

PROBABILISTIC CONTEXT-FREE GRAMMARS

Two Related Problems

- Input: sentence $\boldsymbol{w} = (w_1, \ldots, w_n)$ and CFG G
- Output (recognition): true iff $\boldsymbol{w} \in$ Language(G)
- Output (parsing): one or more derivations for $\boldsymbol{w}$, under G

Probabilistic Context-Free Grammars

- Vocabulary of terminal symbols, Σ
- Set of nonterminal symbols, N
- Special start symbol $S \in N$
- Production rules of the form $X \rightarrow \alpha$, each with a positive weight,

  where

  $X \in N$

  $\alpha \in (N \cup \Sigma)^*$ (in CNF: $\alpha \in N^2 \cup \Sigma$)

  $\forall X \in N, \sum_\alpha p(X \rightarrow \alpha) = 1$

A Sample PCFG

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.05] | the [.80] | a [.15]	
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.35]	$Verb \rightarrow book$	[.30]
$NP \rightarrow Nom$	[.05]	$Verb \rightarrow include$	[.30]
$NP \rightarrow Pronoun$	[.40]	$Verb \rightarrow want$	[.40]
$Nom \rightarrow Noun$	[.75]	$Aux \rightarrow can$	[.40]
$Nom \rightarrow Noun\ Nom$	[.20]	$Aux \rightarrow does$	[.30]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[.40]
$VP \rightarrow Verb\ NP\ NP$	[.05]	$Pronoun \rightarrow you$ [.40] | I [.60]	

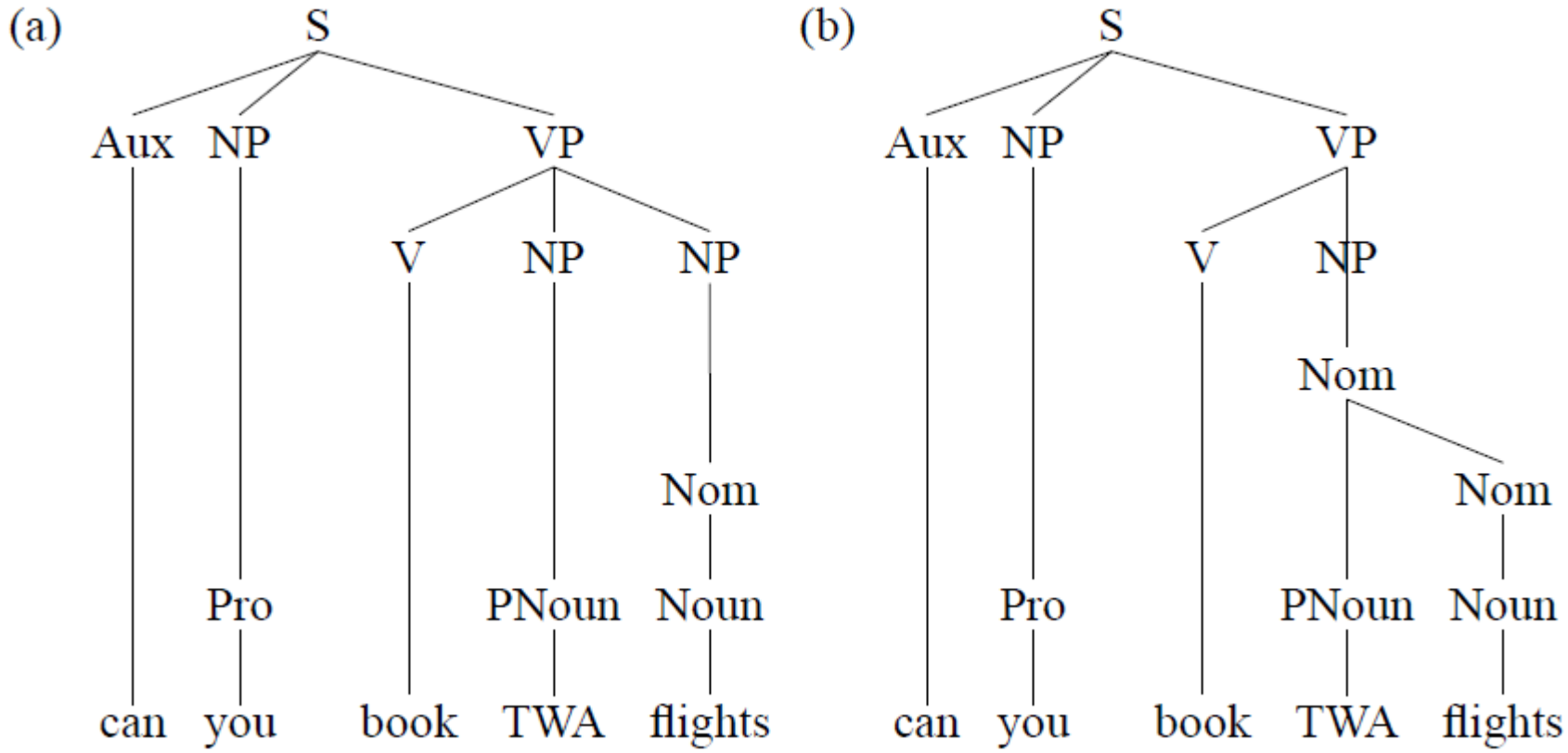
Figure 12.1 A PCFG; a probabilistic augmentation of the miniature English grammar and lexicon in Figure 10.2. These probabilities are not based on a corpus; they were made up merely for expository purposes.

The Probability of a Parse Tree

The joint probability of a particular parse *T* and sentence *S*, is defined as the product of the probabilities of all the rules *r* used to expand each node *n* in the parse tree:

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

An Example—Disambiguation



An Example—Disambiguation

- Consider the productions for each parse:

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

Probabilities

$$P(T_l) = .15 * .40 * .05 * .05 * .35 * .75 * .40 * .40 * .40 * .30 * .40 * .50$$
$$= 1.5 \times 10^{-6}$$

book flights for (on behalf of) TWA

$$P(T_r) = .15 * .40 * .40 * .05 * .05 * .75 * .40 * .40 * .40 * .30 * .40 * .50$$
$$= 1.7 \times 10^{-6}$$

book flights that are on TWA

We favor the tree on the right in disambiguation because it has a higher probability.

What Can You Do With a PCFG?

- Just as with CFGs, PCFGs can be used for both parsing and generation, but they have advantages in both areas:
  - **Parsing**
    - CFGs are good for “precision” parsers that reject ungrammatical sentences
    - PCFGs are good for robust parsers that provide a parse for every sentence (no matter how improbable) but assign the highest probabilities to good sentences
    - CFGs have no built-in capacity for disambiguation—one parse is as good as another, but PCFGs assign different probabilities to “good” parses and “better” parses that can be used in disambiguation
  - **Generation**
    - If a properly-trained PCFG is allowed to generate sentences, it will tend to generate many plausible sentences and a few implausible sentences
    - A well-constructed CFG will generate only grammatical sentences, but many of them will be strange; they will be less representative of the content of a corpus than a properly-trained PCFG

Where Do the Probabilities in PCFGs Come From?

- From a **tree bank**

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

- From a **corpus**
  - Parse the corpus with your CFG
  - Count the rules for each parse
  - Normalize
  - But wait, most sentences are ambiguous!
    - "Keep a separate count for each parse of a sentence and weigh each partial count by the probability of the parse it appears in."

Random Generation Toy Example

V → leaves	0.02
V → leave	0.01
V → snacks	0.02
V → snack	0.01
V → table	0.04
V → tables	0.02
N → snack	0.08
N → snacks	0.02
N → table	0.03
N → tables	0.01
N → leaf	0.01
N → leaves	0.01
Dt → the	0.6
P → on	0.3

S → NP VP	0.8
S → VP	0.2
NP → Dt N'	0.5
NP → N'	0.4
N' → N	0.7
N' → N' PP	0.2
PP → P NP	0.8
VP → V NP	0.4
VP →VP PP	0.4
VP → V	0.2

Randomly generated 10000 sentences with the grammar at left.

5634 unique sentences generated.

Random Generation Toy Example

V → leaves	0.02
V → leave	0.01
V → snacks	0.02
V → snack	0.01
V → table	0.04
V → tables	0.02
N → snack	0.08
N → snacks	0.02
N → table	0.03
N → tables	0.01
N → leaf	0.01
N → leaves	0.01
Dt → the	0.6
P → on	0.3

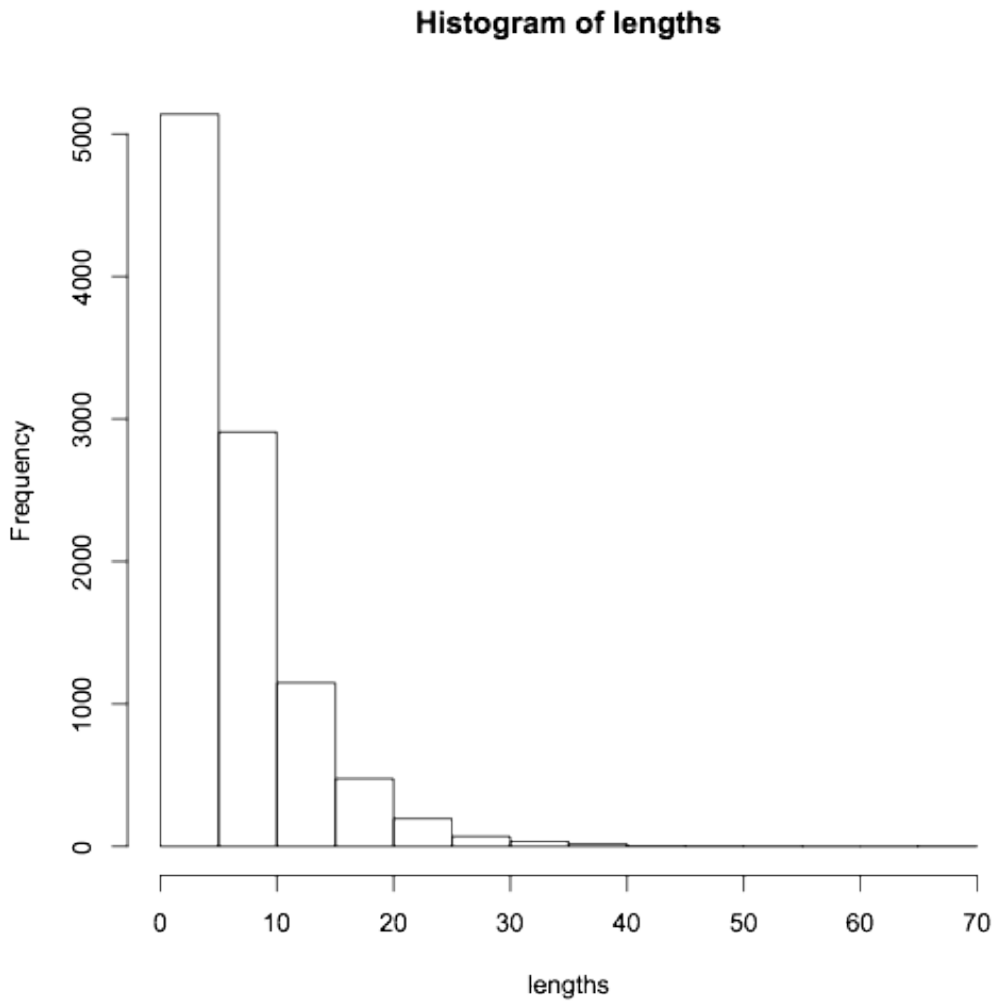
S → NP VP	0.8
S → VP	0.2
NP → Dt N'	0.5
NP → N'	0.4
N' → N	0.7
N' → N' PP	0.2
PP → P NP	0.8
VP → V NP	0.4
VP →VP PP	0.4
VP → V	0.2

135 table
125 the snack table
93 snack table
75 tables
72 snack snacks
64 table the snack
63 the snack snacks
62 leaves
59 the snack leaves
59 table snack
…
1 leaf leave snacks on table on the table on snack on the snack
1 leaf leave snack on table
1 leaf leave snack on snack
1 leaf leaves leaf
1 leaf leaves
1 leaf leave on the tables on the table on snack on table on the snack on snack

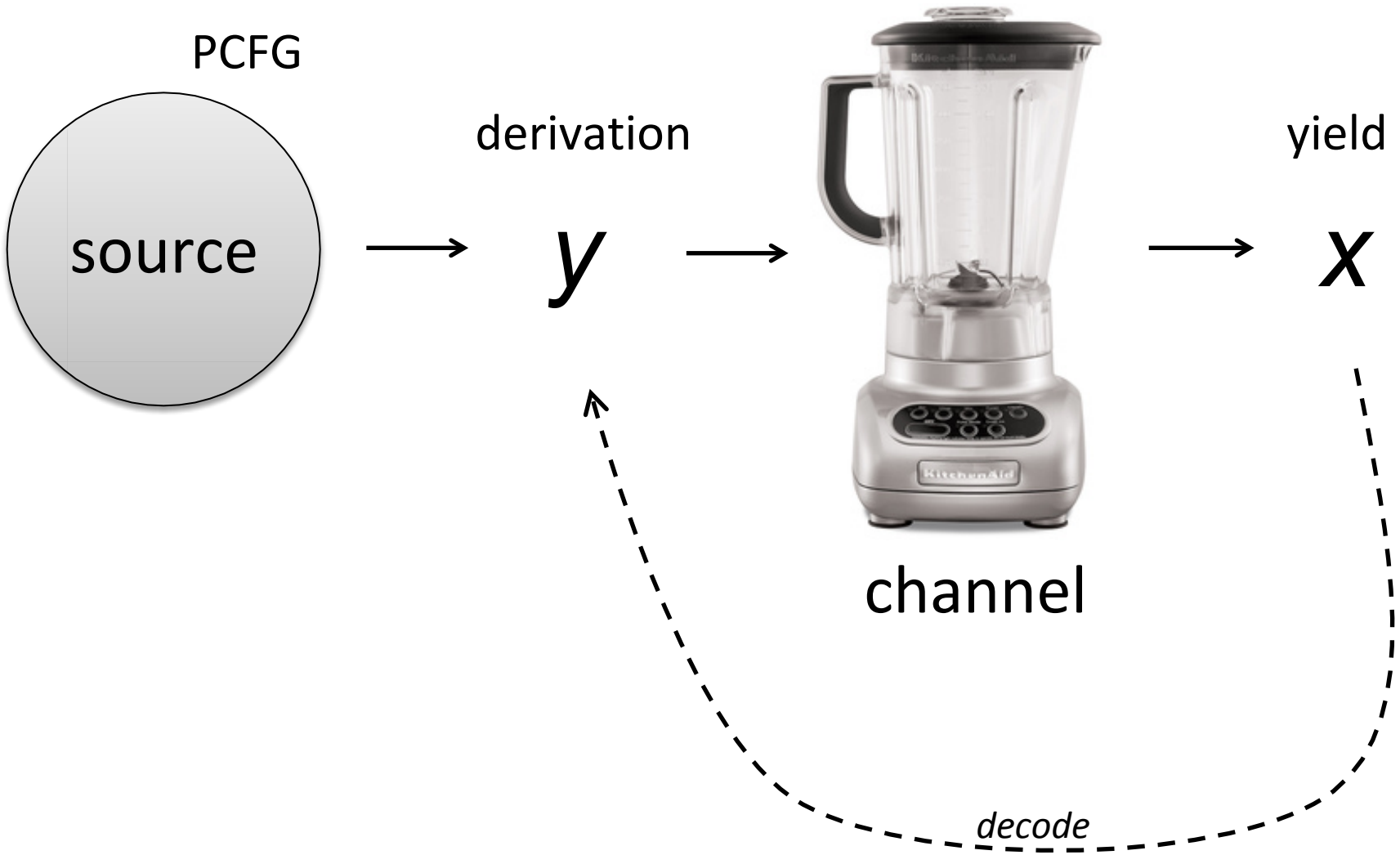
Random Generation Toy Example

V → leaves	0.02
V → leave	0.01
V → snacks	0.02
V → snack	0.01
V → table	0.04
V → tables	0.02
N → snack	0.08
N → snacks	0.02
N → table	0.03
N → tables	0.01
N → leaf	0.01
N → leaves	0.01
Dt → the	0.6
P → on	0.3

S → NP VP	0.8
S → VP	0.2
NP → Dt N’	0.5
NP → N’	0.4
N’ → N	0.7
N’ → N’ PP	0.2
PP → P NP	0.8
VP → V NP	0.4
VP →VP PP	0.4
VP → V	0.2



PCFGs as a Noisy Channel



PROBLEMS WITH PCFGS

Structural Dependencies

- In CFGs, each rule is independent of each other rule
- This carries over into PCFGs, and can be a problem
  - Take the rules
    - S → NP VP
    - VP → V NP
  - In actual treebanks, the first NP (subject) is far more likely to be rewritten as a pronoun than the second NP (object)
  - There is no way to capture this in vanilla PCFGs

Lexical Dependencies

- Vanilla PCFGs are not sensitive to words
  - Words only enter the picture when you rewrite preterminals as terminals (words)
  - Higher up the tree, PCFGs have no way of "knowing" what words will appear below
- However, lexical information is important (as in selecting the correct parse with ambiguous prepositional phrase attachment)
  - *Moscow [sent [more than 100,000 soldiers [into Afghanistan]]]*
  - *Moscow [sent [more than 100,000 soldiers] [into Afghanistan]]*
  - *Sent* subcategorizes for a destination, favoring VP attachment, but a vanilla PCFG has no way of knowing this
- How can we solve this problem?

PROBABILISTIC LEXICALIZED CONTEXT FREE GRAMMARS

The Concept of **Head**

- Headedness is an important concept in syntax
- For our purposes, the most “important” word in a constituent is the head
  - It is the word that determines the type (label) of the constituent
    - For example, a noun phrase is headed by a noun
    - A verb phrase, and a sentence, is headed by a verb
  - Linguists argue over exactly which words are most important, leading to somewhat different schemes for headedness, but there is broad agreement
- In lexicalized grammars/trees we augment the label with the head

A Lexicalized Tree

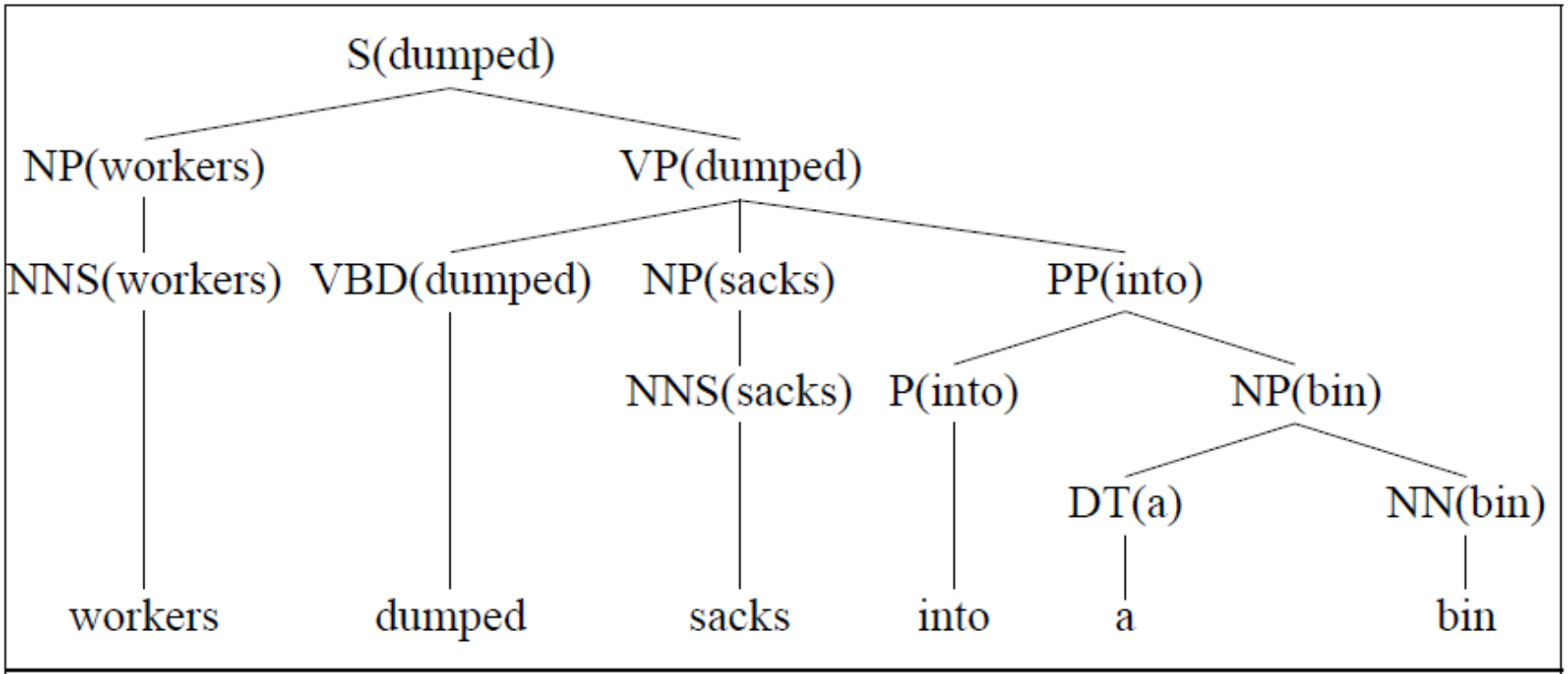


Figure 12.5 A lexicalized tree from Collins (1999).

Lexicalized Grammars

- There are fancy ways of looking at rules in a lexicalized grammar (involving feature structures) but we are going to look at them in a simple way:
  - A lexicalized grammar is like a vanilla PCFG only with a lot more rules
  - It is as if you took your treebank and added a new rule for each combination of heads that you observed.
    - S(dumped) → NP(workers) VP(dumped)
    - VP(dumped) → VBD(dumped) PP(into)
    - …

Lexicalized Grammars

- Viewed in this way...
  - Lexicalized grammars are huge (perhaps impractically huge)
  - However, they can be used with the same algorithms we have already learned
- Lexicalized grammars require more training data than vanilla PCFGs, but they can capture probabilistic patterns that PCFGs could never capture
- In most contemporary applications, lexicalized grammars are chosen over vanilla PCFGs

Questions?