Syntactic Parsing Introduction

Niranjan Balasubramanian
Stony Brook University

Aug 27, 2015

Some slides adapted from these *fantastic* researchers:
Ray Mooney, Michael Collins, and Chris Manning.

This class.

- Syntactic Parsing
  - What is it?
  - A simple formalism (PCFG)
  - Issues w/ Vanilla PCFG
  - Hints at how these are typically addressed.

Syntactic Parsing – Refresher Quiz

- What is syntactic parsing?
  - Identifying *syntactic* structure underlying a sentence.
  - Assumes that there is a set of rules that underlie language.

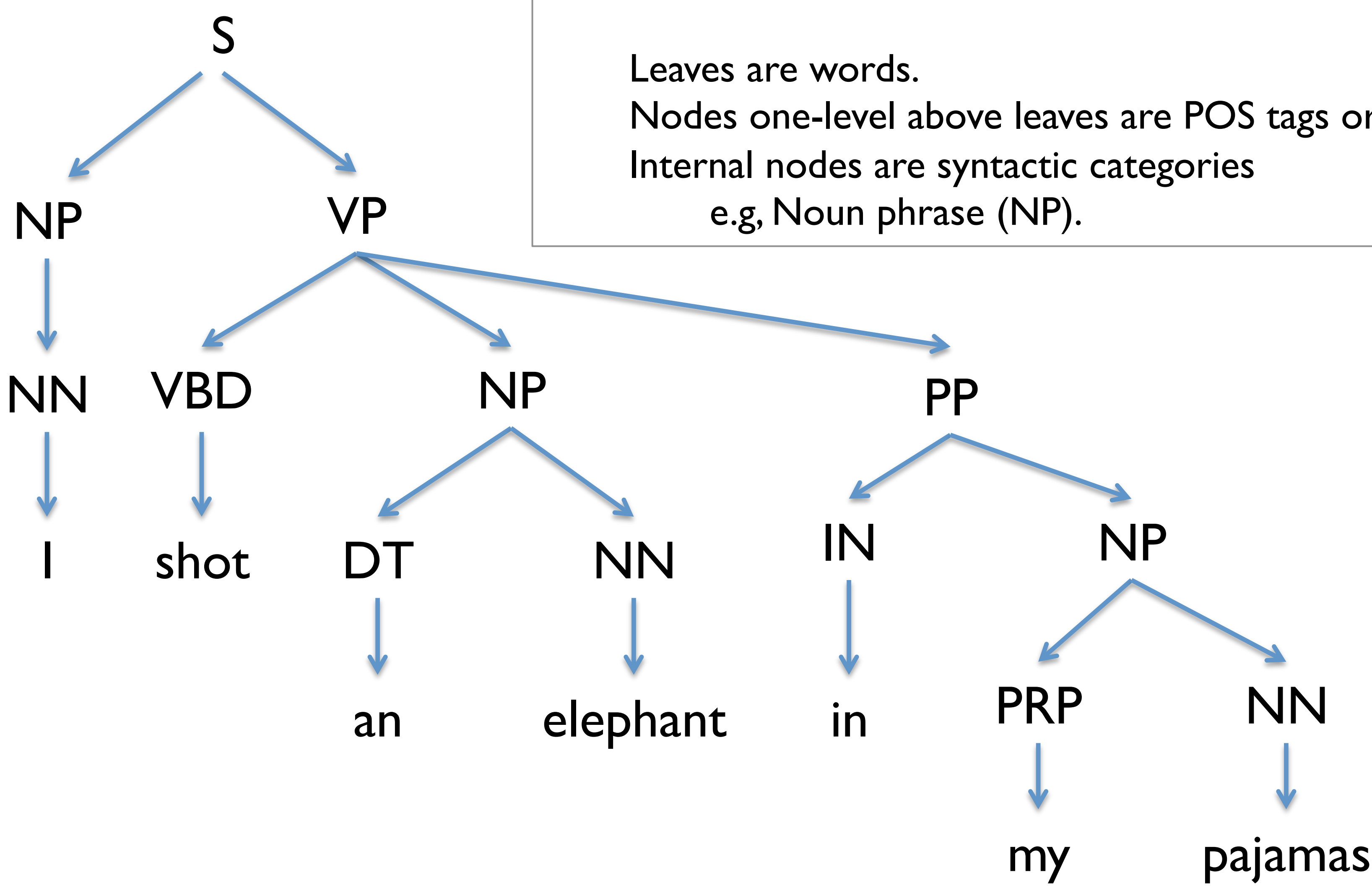
- Why is it useful?
  - Serves as a model that explains the observed language string.
  - Use it to:
    - Predict or complete sentences.
    - Re-organize, simplify sentences.
    - Learn semantic phenomenon identifiable via syntactic patterns.

- What are the big issues in syntactic parsing?
  - Dependence on semantics
  - Lexicalization helps but hurts generalization.
  - Speed. Parsing is a $O(N^3)$ business w/ a constant the size of grammar.

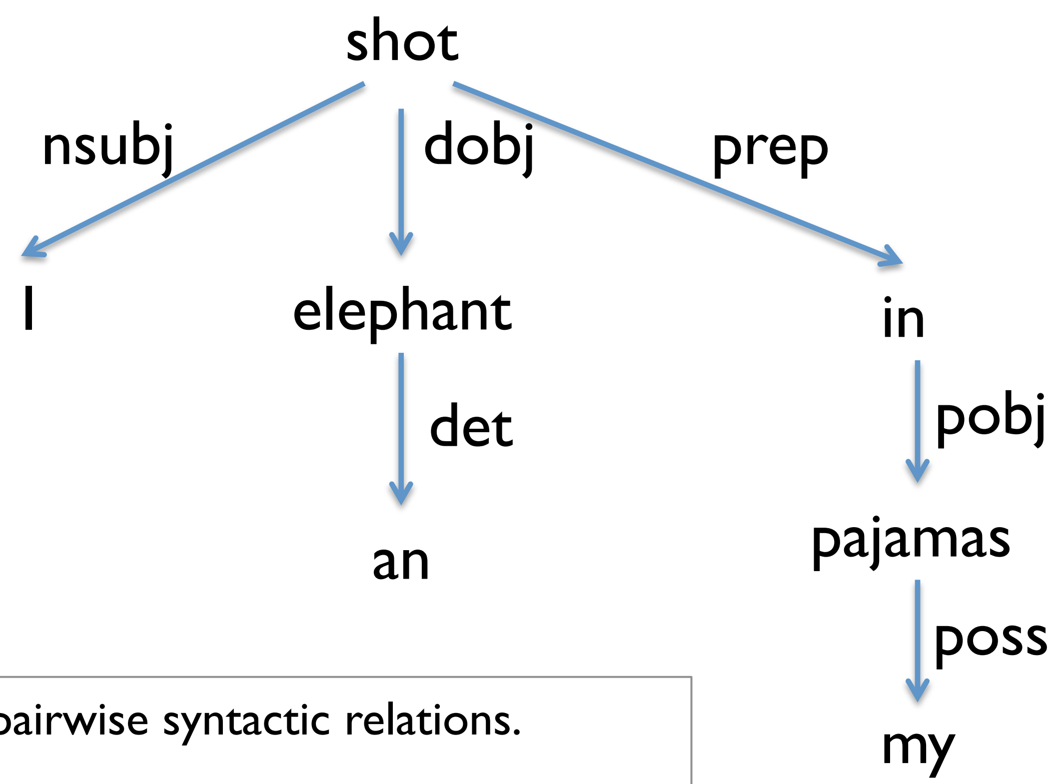
Constituency Parse

A tree representing nested compositional structure.

Leaves are words.
Nodes one-level above leaves are POS tags or
Internal nodes are syntactic categories
e.g, Noun phrase (NP).



Dependency Parse



A tree representing pairwise syntactic relations.

Nodes are words.
Edges are grammatical relations.

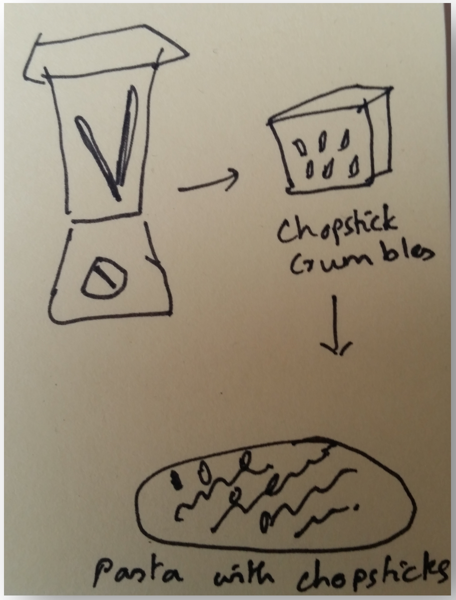
Constituency vs. Dependency Grammars

- What is the difference?
  - Constituency parse groups words that act as a unit.
  - Constituents are typically "headed" by a particular type of word.
    - E.g., noun phrases are headed by a noun, verb phrases by a verb.
  - Dependency parse directly specifies relations between heads and their dependents.

- A deterministic procedure can transform constituency into dependency.

- Dependency parses are more compact than constituency parses.
  - **[How does this impact the automatic parsing?]**

Why is syntactic parsing hard?



Ambiguity:

PP attachment

Noun pre-modifiers

What is so hard about this?

>> Write down a grammar in a formal language that has sufficient representation power.

Noam Chomsky tried this in his thesis!

Turns out this is not simple.

Ambiguity and coverage make it hard.

Fundamental trade-off:

1) Smaller grammars have limited coverage. No parses for many sentences.
2) Large grammars improve coverage but are ambiguous and yield more parses.

Main Questions in Parsing

- What is the formalism for the grammar?
  - Constituency (Phrase-structure) vs. Dependency Grammar

- How does one get the grammar?
  - We are not writing it down!
  - To address ambiguity, we need probabilities attached to the grammar.

- How does one parse sentences given a grammar?

Parsers done three ways!

- Probabilistic Context Free Grammars
- Transition-based Parsing (Next class)
- Graph-based methods (Next week)

What are context-free grammars?

- Grammars are a way to encode rules that can generate strings in a language.
  - Recall formal languages such as regular, context free, context-sensitive etc.

- What grammar generates the following strings?
  - A, AA, AAA, AAA, …?
  - AB, ABA, ABBA, …?
  - ABCD, AABBCCDD, …?

- What kind of language is English?
  - Not regular.
  - Not necessarily context-free.

  [Can you come up with sentences that show this?]

Context-Free Grammars -- Formally

- *N* a set of ***non-terminal symbols*** (or ***variables***)
- Σ a set of ***terminal symbols*** (disjoint from *N*)
- *R* a set of ***productions*** or ***rules*** of the form:

  $$A \rightarrow \beta,$$

  where A is a non-terminal and

  β is a string of symbols from $(\Sigma \cup N)^*$

- S, a designated non-terminal called the ***start symbol***

Strings that can be generated by applying a sequence of rules from R are said to be in the language of the grammar.

Parsing becomes the task of identifying if a string is generated by the grammar (and recovering the sequence of rules that generated it).

Probabilistic Context Free Grammars (PCFGs)

Probabilistic Context Free Grammars are CFGs + Probabilities

S	$\Rightarrow$	NP	VP	1.0
VP	$\Rightarrow$	Vi		0.4
VP	$\Rightarrow$	Vt	NP	0.4
VP	$\Rightarrow$	VP	PP	0.2
NP	$\Rightarrow$	DT	NN	0.3
NP	$\Rightarrow$	NP	PP	0.7
PP	$\Rightarrow$	P	NP	1.0

Vi	$\Rightarrow$	sleeps	1.0
Vt	$\Rightarrow$	saw	1.0
NN	$\Rightarrow$	man	0.7
NN	$\Rightarrow$	woman	0.2
NN	$\Rightarrow$	telescope	0.1
DT	$\Rightarrow$	the	1.0
IN	$\Rightarrow$	with	0.5
IN	$\Rightarrow$	in	0.5

Parsing using PCFGs

Modeling Assumption:

PCFG rules applied recursively derive sentences.

What about ambiguity?

The product of the probabilities of the rules scores each parse.

What independence assumptions justify this factoring?

DERIVATION	RULES USED	PROBABILITY
S	$S \rightarrow NP\ VP$	1.0
NP VP	$NP \rightarrow DT\ N$	0.3
DT N VP	$DT \rightarrow$ the	1.0
the N VP	$N \rightarrow$ dog	0.1
the dog VP	$VP \rightarrow VB$	0.4
the dog VB	$VB \rightarrow$ laughs	0.5
the dog laughs		

TOTAL PROBABILITY = $1.0 \times 0.3 \times 1.0 \times 0.1 \times 0.4 \times 0.5$

How to parse given a PCFG?

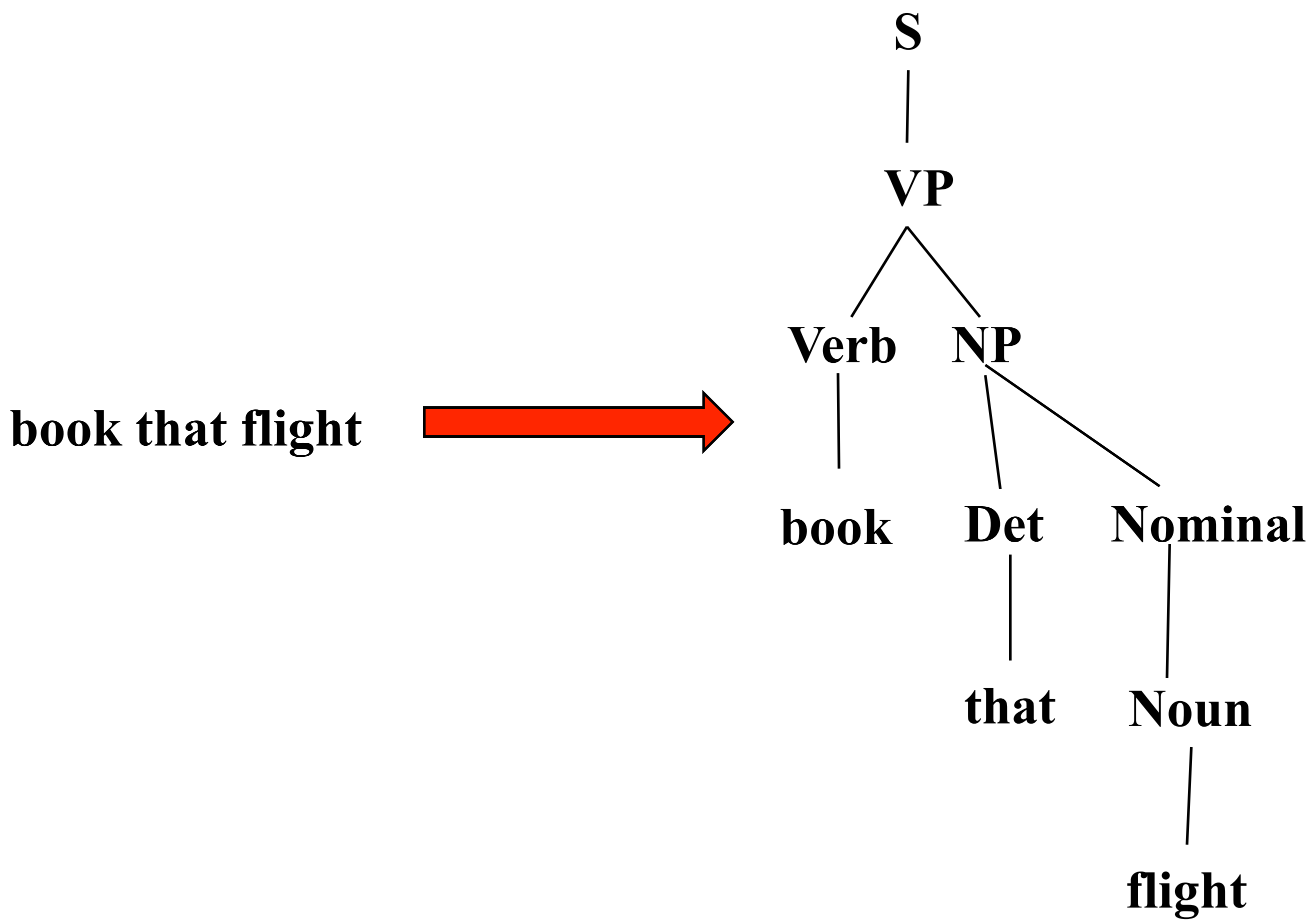
- Exhaustive search of the space of derivations that can produce the input sentence is *infeasible*.

- **Why? What is inefficient about this approach?**

- **So how can we make it better?**

Top Down Parsing

<u>Idea:</u> Do not explore paths that cannot lead to the sentence.



Top Down Parsing

book that flight

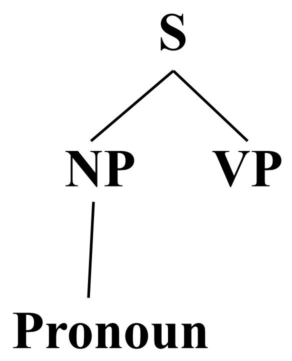
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

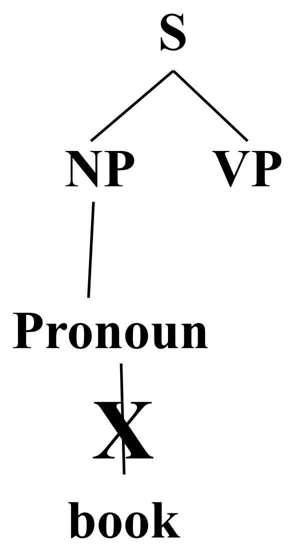
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

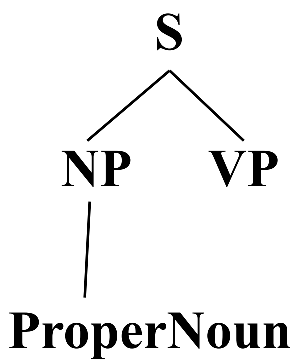
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

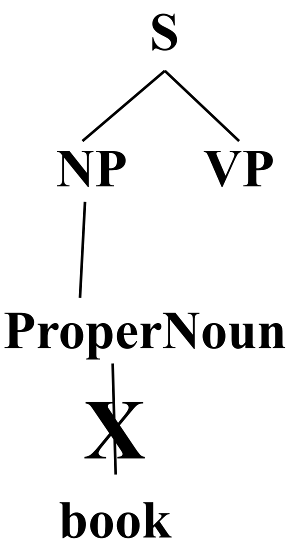
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

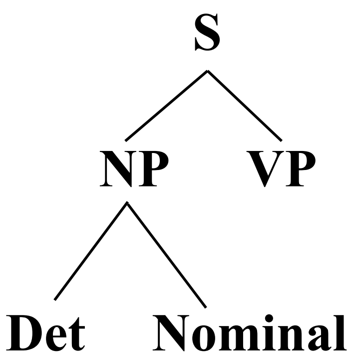
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

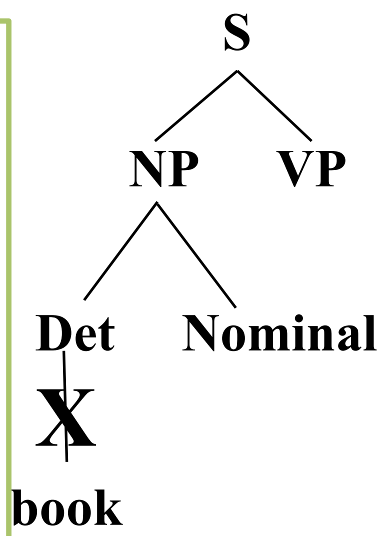
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

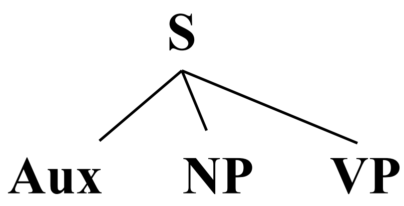
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

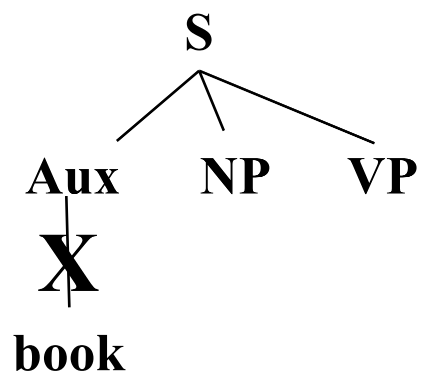
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

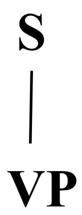
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

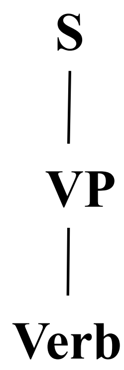
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

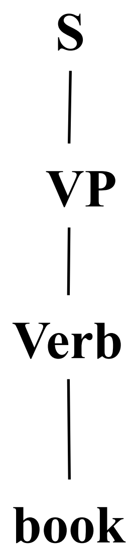
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

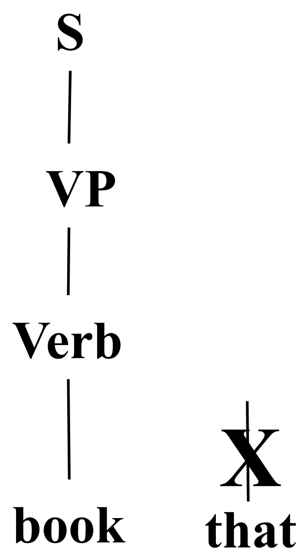
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

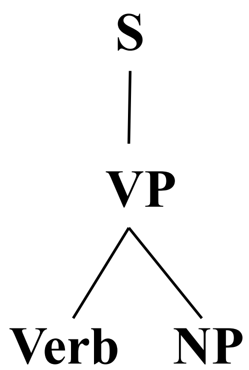
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

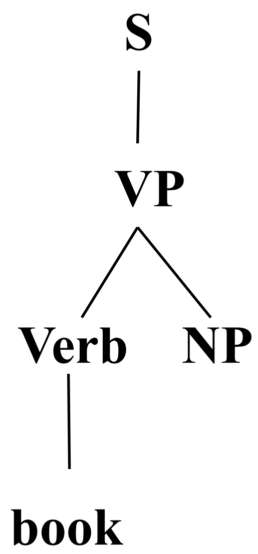
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP



Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

book that flight

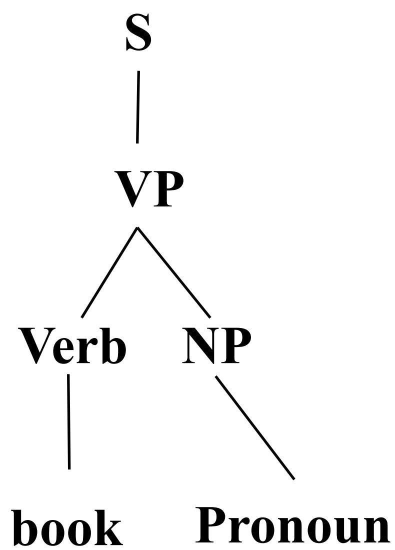
S → NP VP
S → Aux NP VP
S → VP

NP → Pronoun
NP → Proper-Noun
NP → Det Nominal

Nominal → Noun
Nominal → Nominal Noun
Nominal → Nominal PP

VP → Verb
VP → Verb NP
VP → VP PP

PP → Prep NP

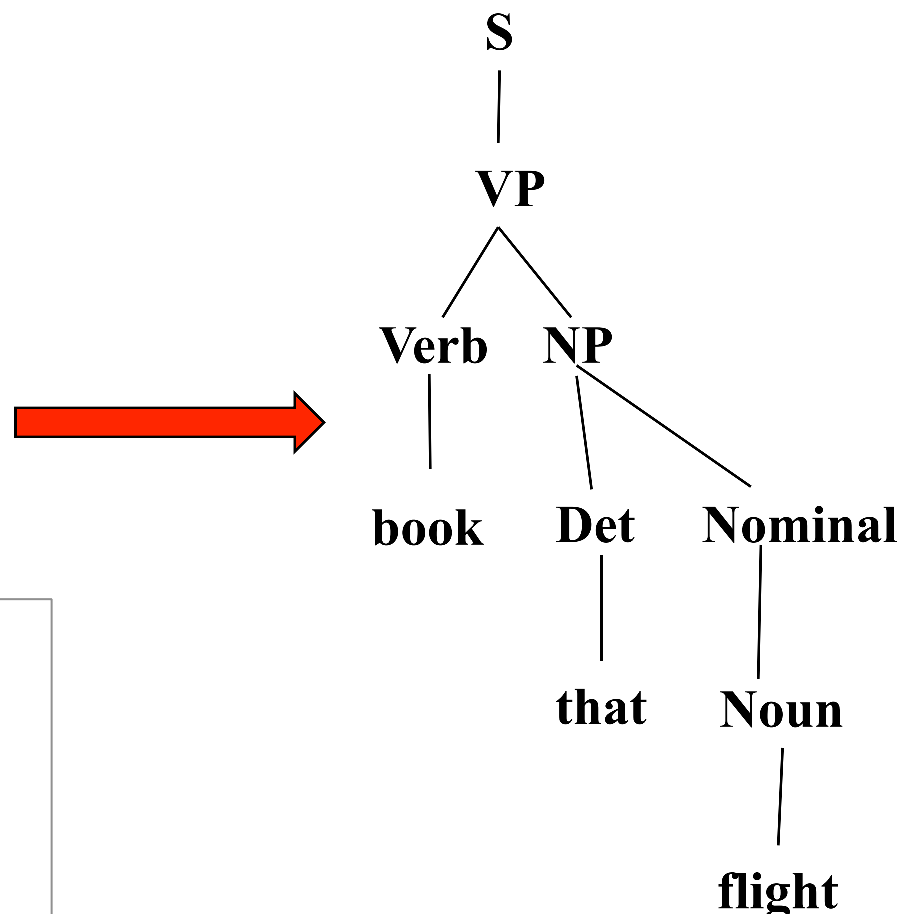


Det → the | a | that | this
Noun → book | flight | meal
Verb → book | prefer
Pronoun → I | he | she | me
Aux → does
….

Top Down Parsing

And on it goes until ....

book that flight



Rules Used:

S -> VP
VP -> Verb NP
Verb -> book
NP -> Det Nominal
Det -> that
Nominal -> Noun
Noun -> flight

Efficiency of Parsing

- Top down (and bottom up) are quite bad.
  - Asymptotic complexity is exponential in the length of the sentence (N).
- Dynamic programming approaches bring the complexity down to $O(N^3)$
  - E.g. Cocke-Young-Kasami algorithm
- Remember grammar size also affects runtime by a constant factor.

[Learn CYK algorithm and understand the impact of grammar size]

How to learn a PCFG?

- Assume you are given example sentences and their parses (generated by humans).

- You can get the rules by inspecting the parses.

- Obtain probabilities by simple maximum Likelihood estimates:

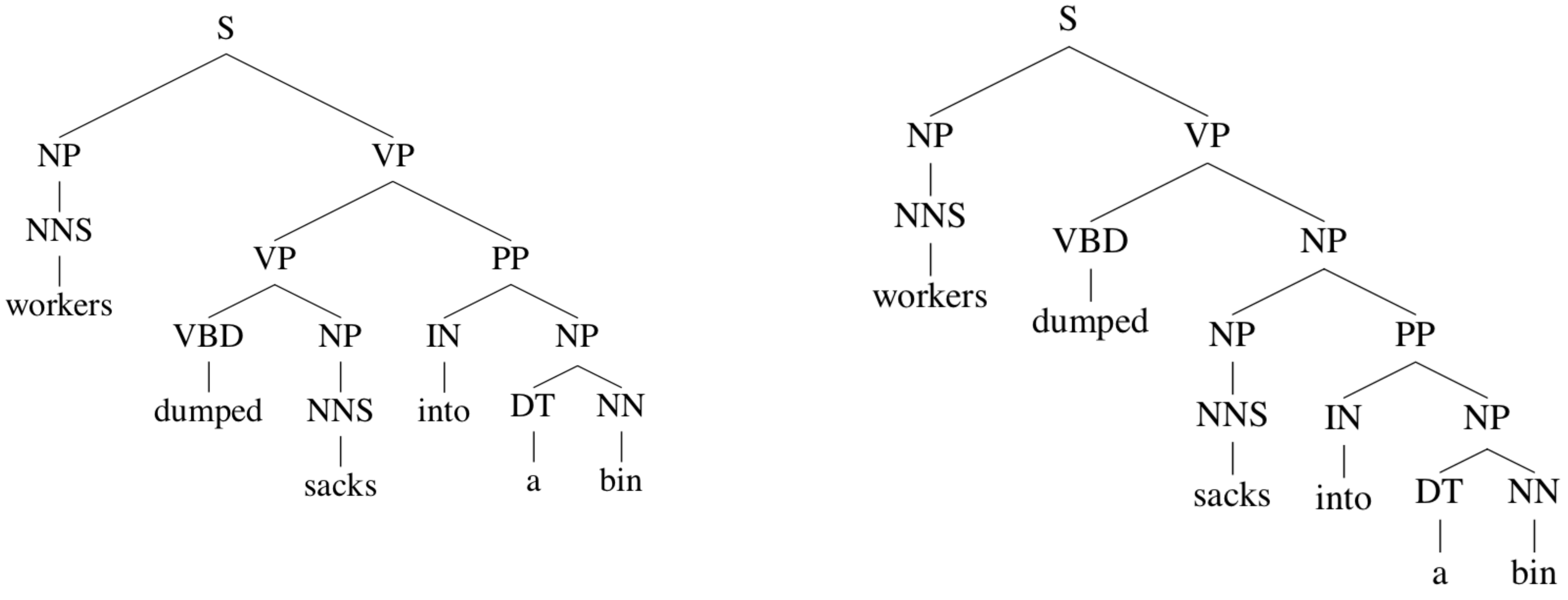
$$P_{ML}(\alpha \to \beta \mid \alpha) = \frac{\text{Count}(\alpha \to \beta)}{\text{Count}(\alpha)}$$

- What could be potential pitfalls with this estimation approach?
  - Unseen words.
  - Unseen constructions.
  - Infrequent combinations.

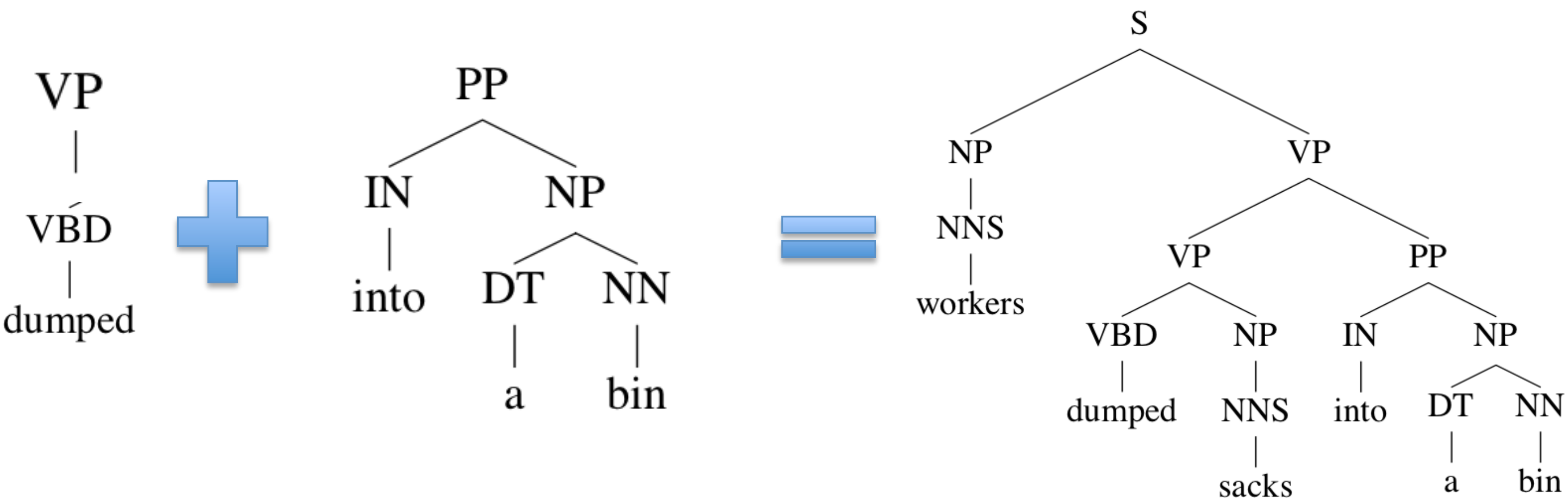
Issues with PCFGs

- Makes strong independence assumptions about language
  - Lexical independence
  - Structural independence

Lexical Dependence: PP Attachment Ambiguity

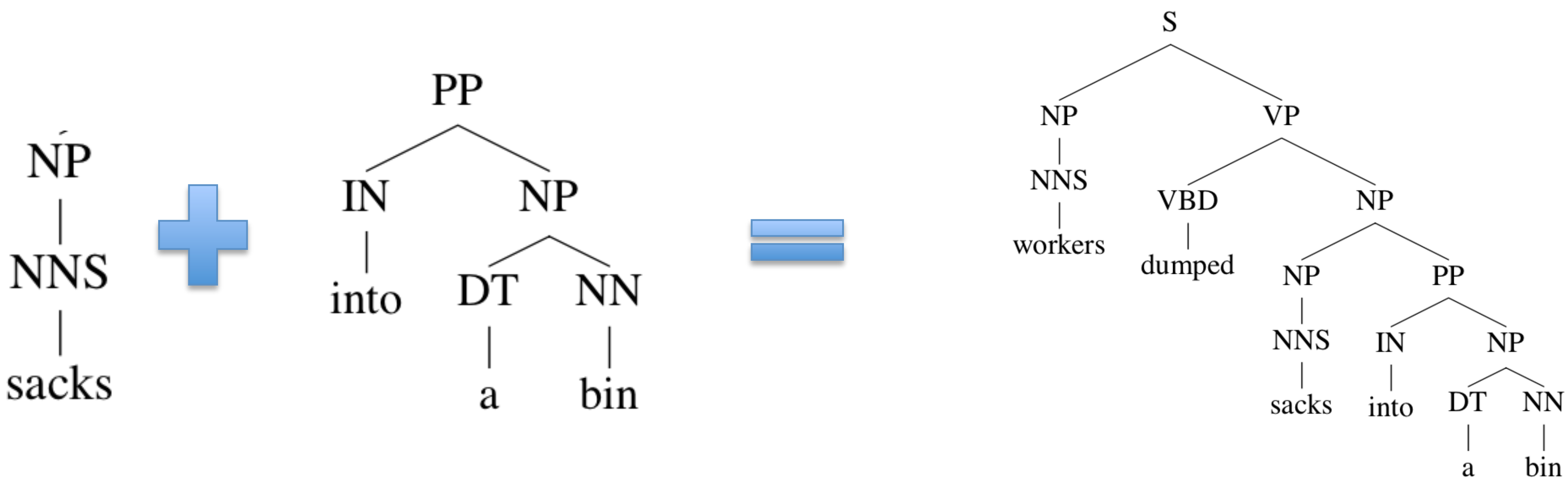


Lexical Dependence: PP Attachment Ambiguity



Prob(Tree 1, Sentence) = …. x Prob(VP -> VP PP | VP) x …

Lexical Dependence: PP Attachment Ambiguity



Prob(Tree 2, Sentence) = …. x Prob(NP -> NP PP | NP) x …

Lexical Dependence: PP Attachment Ambiguity

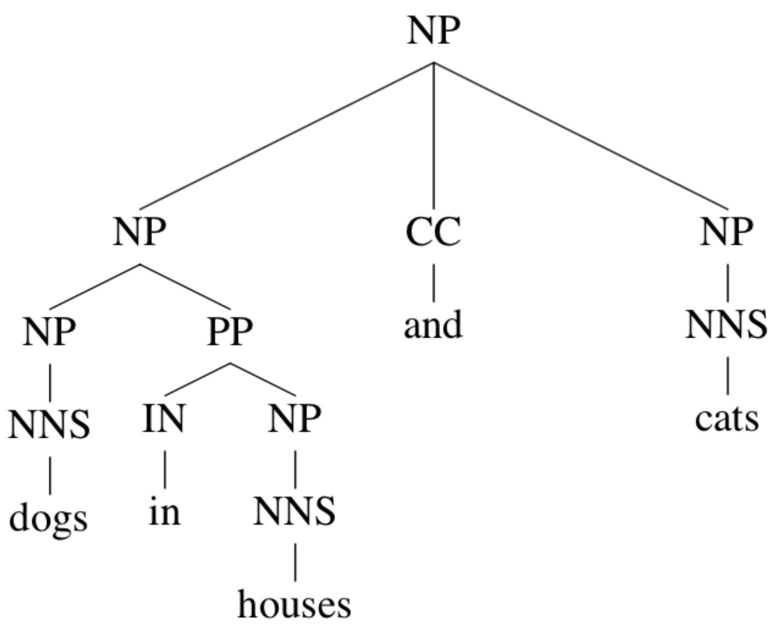
Rules
S $\rightarrow$ NP VP
NP $\rightarrow$ NNS
VP $\rightarrow$ VP PP
VP $\rightarrow$ VBD NP
NP $\rightarrow$ NNS
PP $\rightarrow$ IN NP
NP $\rightarrow$ DT NN
NNS $\rightarrow$ workers
VBD $\rightarrow$ dumped
NNS $\rightarrow$ sacks
IN $\rightarrow$ into
DT $\rightarrow$ a
NN $\rightarrow$ bin

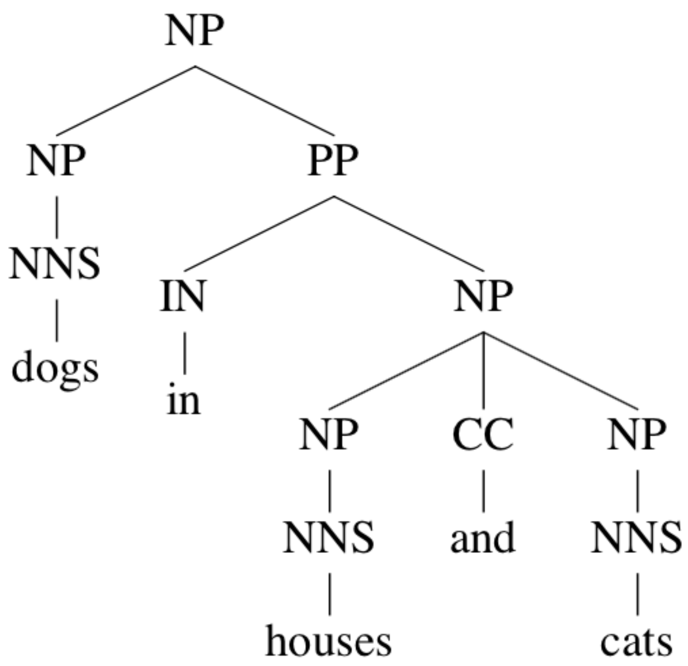
Rules
S $\rightarrow$ NP VP
NP $\rightarrow$ NNS
NP $\rightarrow$ NP PP
VP $\rightarrow$ VBD NP
NP $\rightarrow$ NNS
PP $\rightarrow$ IN NP
NP $\rightarrow$ DT NN
NNS $\rightarrow$ workers
VBD $\rightarrow$ dumped
NNS $\rightarrow$ sacks
IN $\rightarrow$ into
DT $\rightarrow$ a
NN $\rightarrow$ bin

Prob(Tree 1, Sentence) > Prob(Tree 2, Sentence)

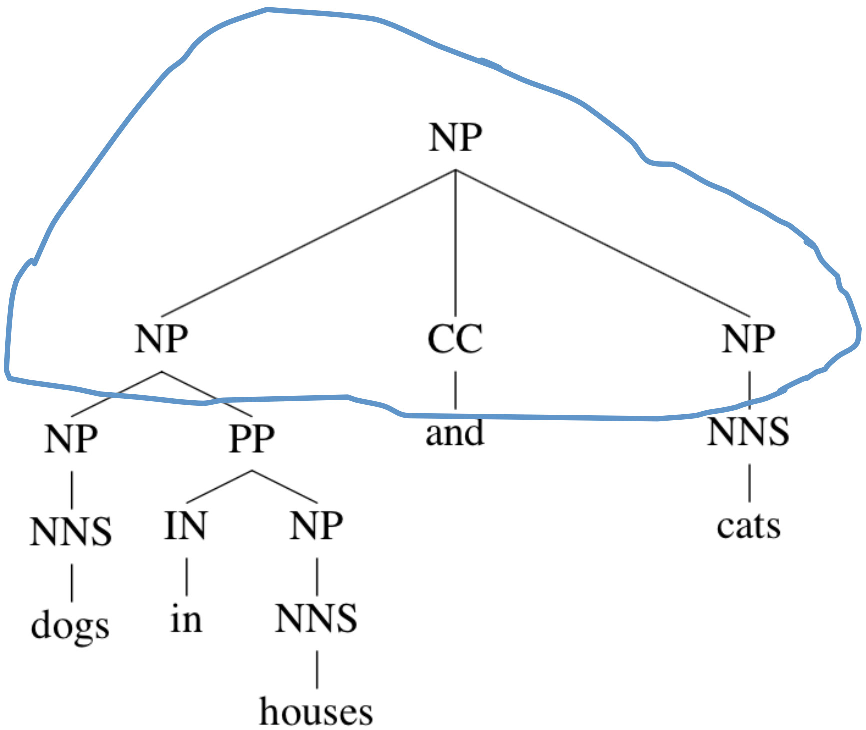
If Prob(VP -> VP PP | VP) > Prob(NP -> NP PP | NP)

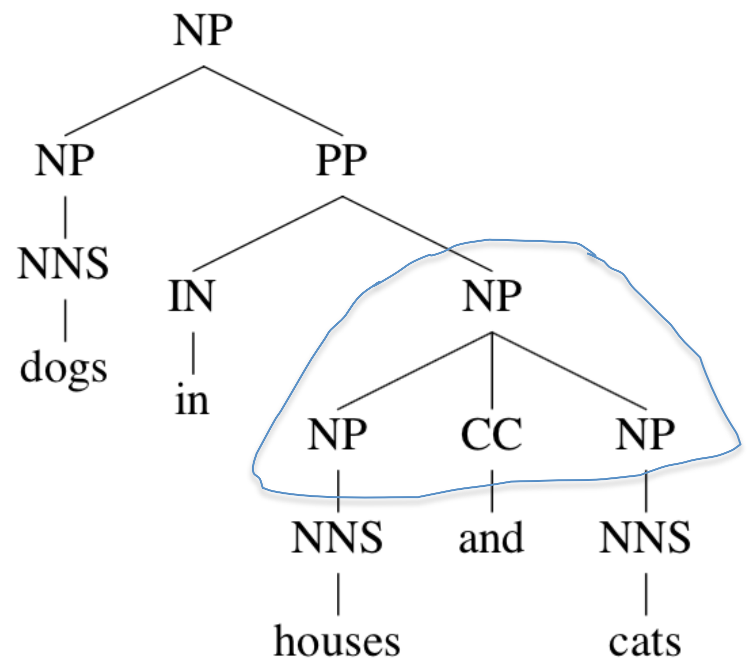
Lexical Dependence: Co-ordination Ambiguity



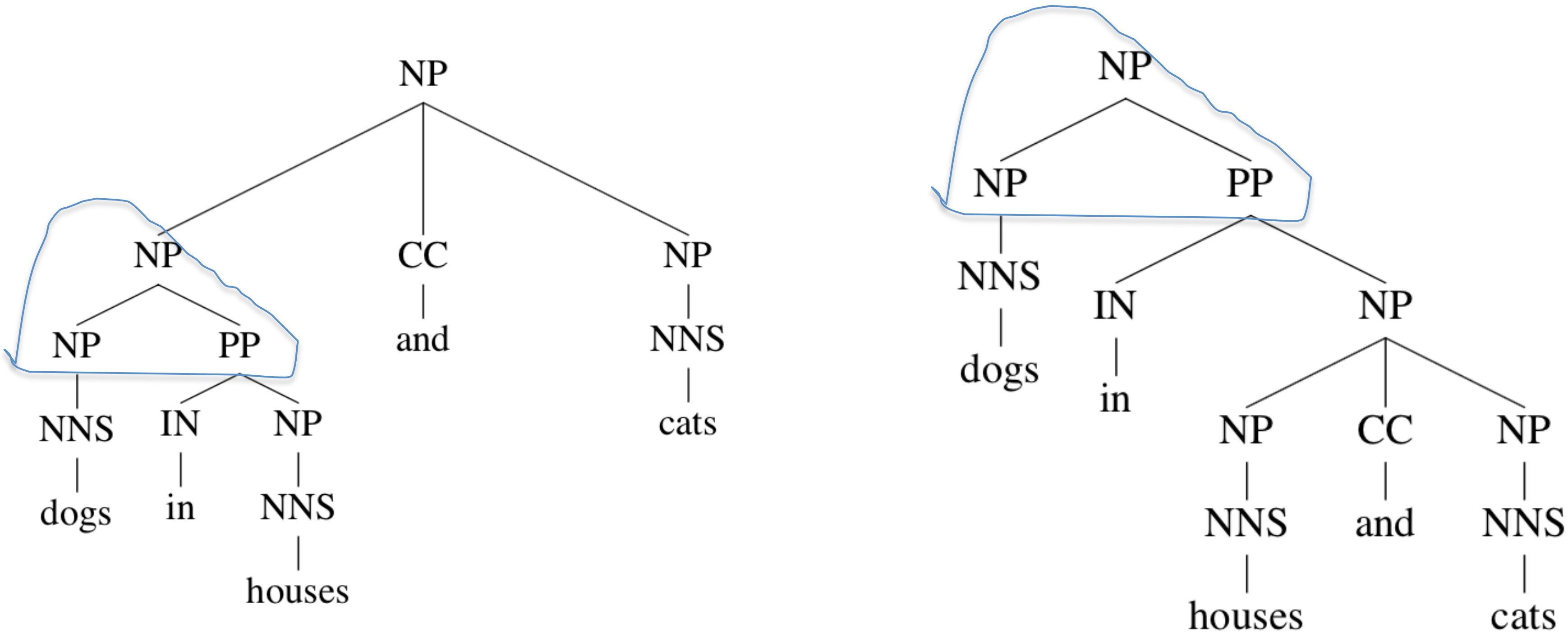


Lexical Dependence: Co-ordination Ambiguity

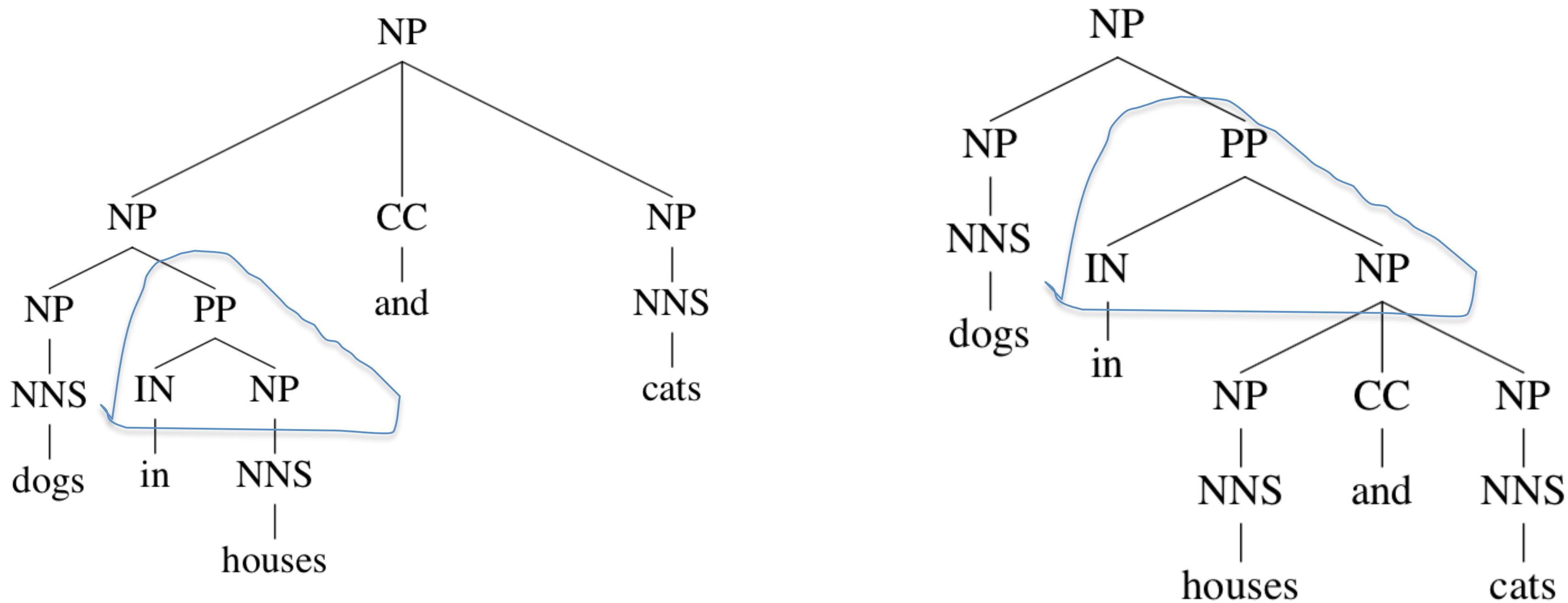




Lexical Dependence: Co-ordination Ambiguity



Lexical Dependence:
Co-ordination Ambiguity



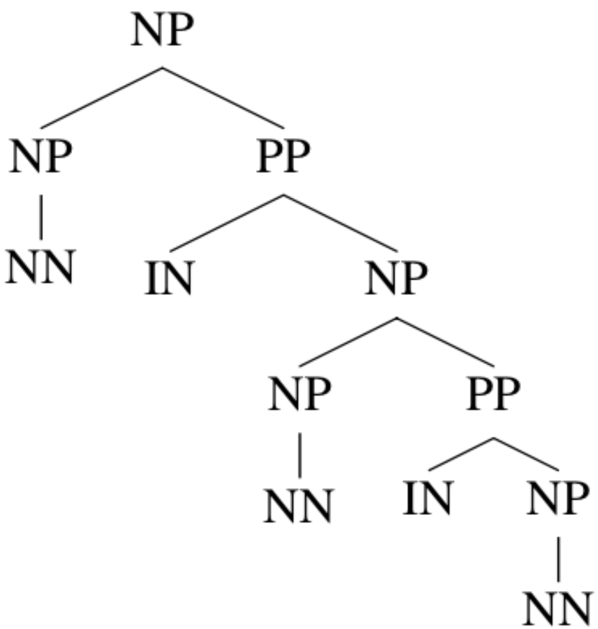
Lexical Dependence: Co-ordination Ambiguity

Rules
NP → NP CC NP
NP → NP PP
NP → NNS
PP → IN NP
NP → NNS
NP → NNS
NNS → dogs
IN → in
NNS → houses
CC → and
NNS → cats

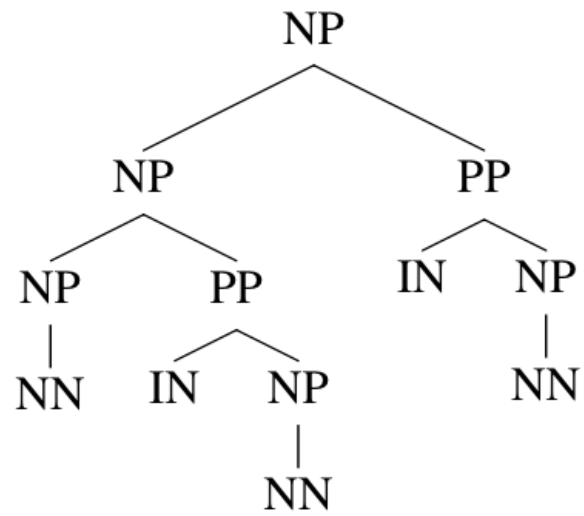
=

Rules
NP → NP CC NP
NP → NP PP
NP → NNS
PP → IN NP
NP → NNS
NP → NNS
NNS → dogs
IN → in
NNS → houses
CC → and
NNS → cats

Structural Preferences



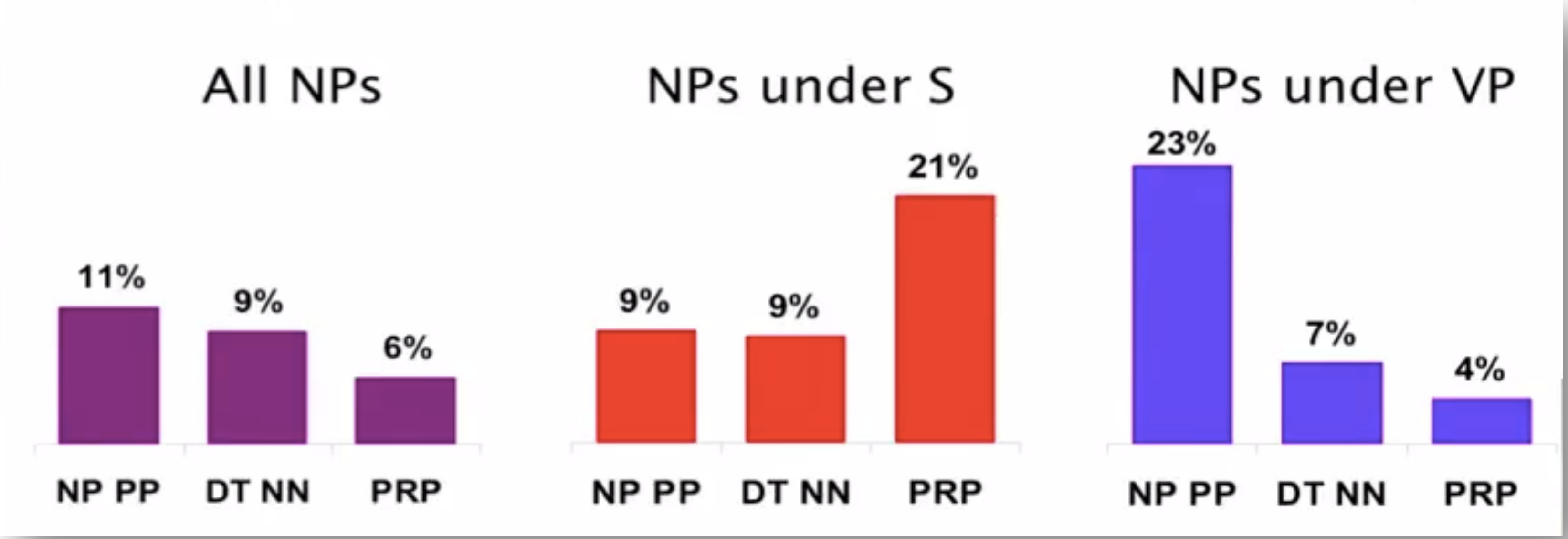
president of [company in Africa]



[president of company] in Africa

- Same rules applied in both cases. Tree probability is the same
- Left structure is twice as likely in Wall Street Journal.
- There are similar issues when PPs can attach to multiple verbs.

Structural Dependence



Structural Dependence: Sub-categorization

- Specific verbs take some types of arguments but not others.
  - Intransitive, transitive, and di-transitive
  - Finite vs. Non-finite verbs.

- A generic VP label hides the different argument preferences of the various sub-categories.

How to address these issues?

- Lexical dependence
  - Introduce lexical items into the tree.
  - Use headwords as part of the node-label. [Charniak 1997]
- Structural Dependence
  - Add more information to non-terminal categories [state splitting]
    - Include information about parents. [Johnson 1998]
    - Include fine-grained information (mark possessives for example)
- Sub-categorization
  - Add information to the non-terminal categories (state splitting)
  - E.g., S -> NP_firstpersonsingular VP_firstpersonsingular

Trade-off: Adding lexical information and fine-grained categories:

a) Increases sparsity -- Need appropriate smoothing.
b) Adds more rules – Can affect parsing speed.

A Summary of the Issues

So what are all these issues essentially pointing out?

- Syntactic categories have different attachment preferences depending on their context.
  - Lexical or otherwise.
- Adding this context results in estimation issues due to sparsity.

This is a central challenge in NLP. Many phenomena have this characteristic.

Typical remedial actions include:

1) Using lexicalization but with generalization or dimensionality reduction.
2) Using carefully constructed features that leverages "expert intuitions" thereby avoiding sparsity issues.

Lets stop here …

- Next class.
  - Research summary and presentation templates.
  - An un-lexicalized parser.
  - Switch a bit for transition based parsing.
  - Annotate some sentences in class.

- Starting next week
  - Student presentations
  - Research reports due before class on Thursday.

- Will add “additional readings”.