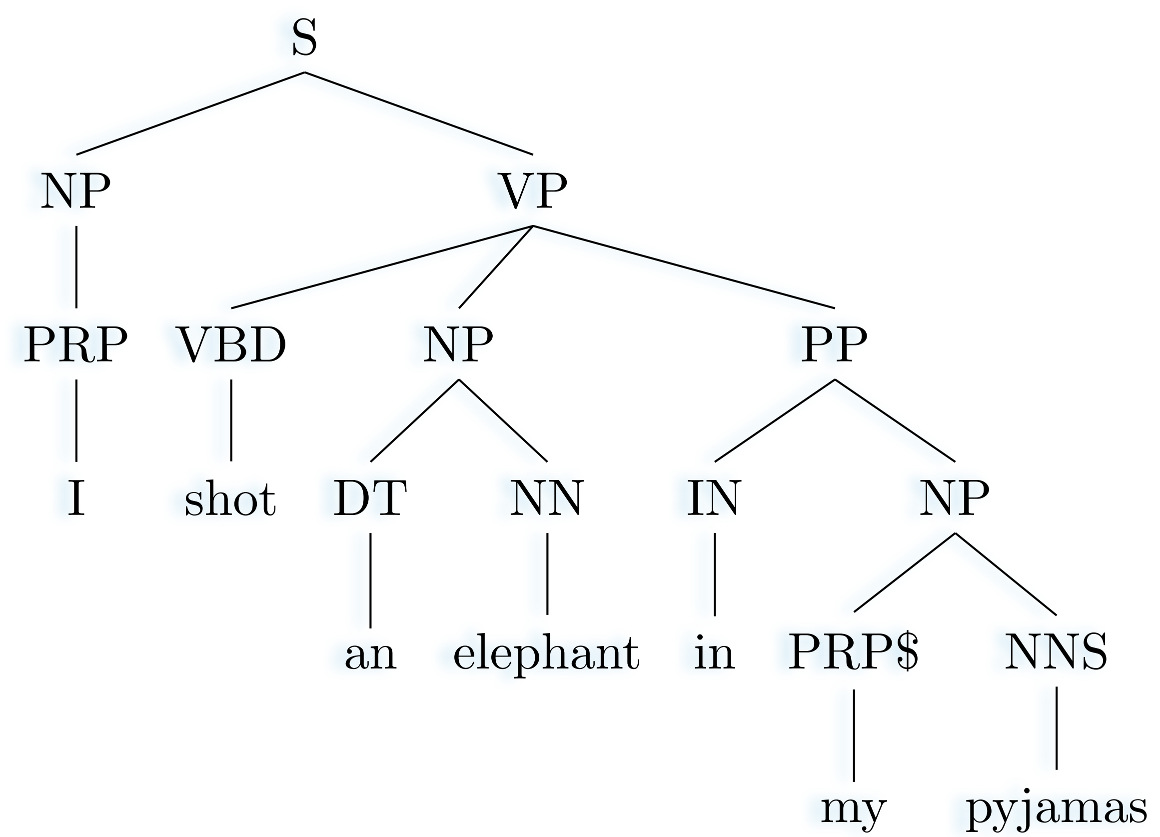
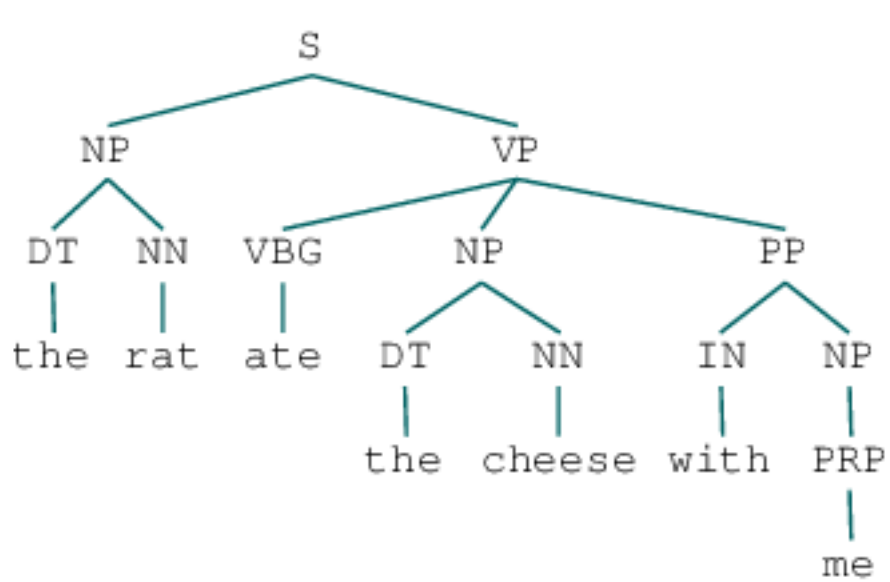


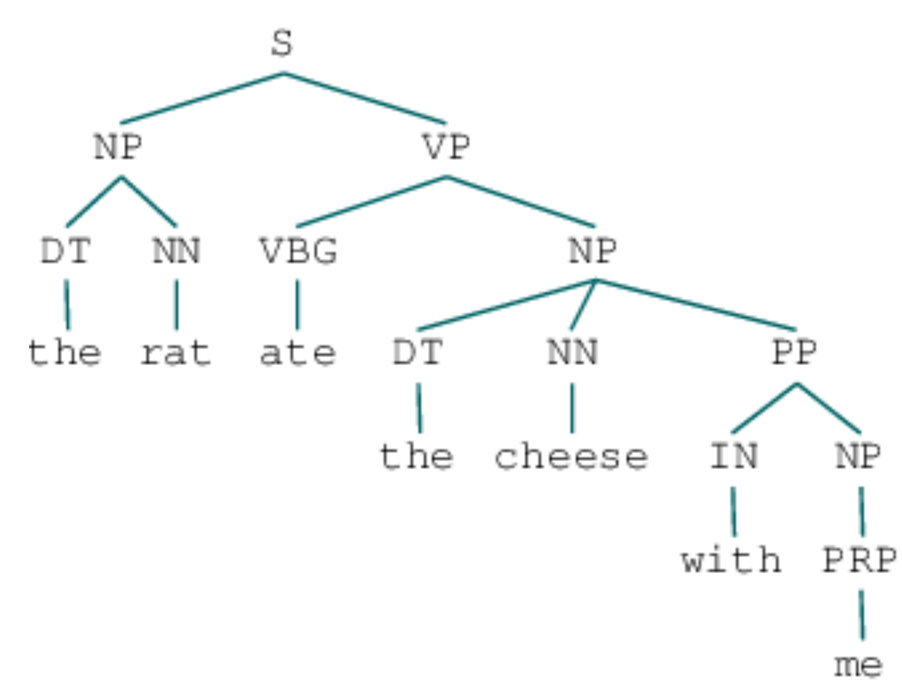
COMP90042 LECTURE 6

PROBABILISTIC PARSING

AMBIGUITY IN PARSING

- Context-free grammars assign hierarchical structure to language
  - Linguistic notion of a '*syntactic constituent*'
  - Formulated as generating all strings in the language; or
  - Predicting the structure(s) for a given string
- Raises problem of ambiguity, e.g., which is better?





OUTLINE

- Probabilistic context-free grammars (PCFGs)
- Parsing using dynamic programming
- Limitations of ‘context-free’ assumption and some solutions:
  - parent annotation
  - head lexicalisation

BASICS OF PROBABILISTIC CFGS

- As for CFGs, same symbol set:
  - Terminals: words such as *book*
  - Non-terminal: syntactic labels such as NP or NN
- Same productions (rules)
  - LHS non-terminal → ordered list of RHS symbols
- In addition, store a **probability** with each production
  - NP → DT NN [p = 0.45]
  - NN → cat [p = 0.02]
  - NN → leprechaun [p = 0.00001]
  - …

PROBABILISTIC CFGS

- Probability values denote
  - Pr(RHS | LHS)
- Consequently they:
  - must be positive values, between 0 and 1
  - must sum to one for a given LHS
- E.g.,
  - NN → aadvark [p = 0.0003]
  - NN → leprechaun [p = 0.0001]
  - NN → Zanzibar [p = 0.0025]
  - $\sum_x \Pr(\text{NN} \rightarrow x \mid \text{NN}) = 1$

A PROBABILISTIC GRAMMAR

S → NP VP [0.8]	Verb → *book* [0.3]
S → VP [0.05]	Verb → *include* [0.3]
S → Aux NP VP [0.15]	Verb → *prefer* [0.4]
NP → Pronoun [0.35]	Noun → *book* [0.1]
…	Noun → *dinner* [0.1]
NP → Nominal [0.15]	Noun → *flight* [0.3]
Nominal → Noun [0.75]	Noun → *meal* [0.15]
Nominal → Nomial Noun [0.20]	Noun → *money* [0.05]
Nominal → Nominal PP [0.05]	Noun → *flights* [0.40]
VP → Verb [0.35]	…
…	

Extract from JM2 Fig. 14.1

GENERATING SENTENCES WITH PCFGS

Déjà vu, it's almost the same as for CFG, with one twist:

1. Start with S, the sentence symbol
2. Choose a rule with S as the LHS
   - **Randomly select a RHS** according to Pr(RHS | LHS) e.g., $S \rightarrow VP$
   - Apply this rule, e.g., substitute VP for S
3. Repeat step 2 for each non-terminal in the string (here, VP)
4. Stop when no non-terminals remain

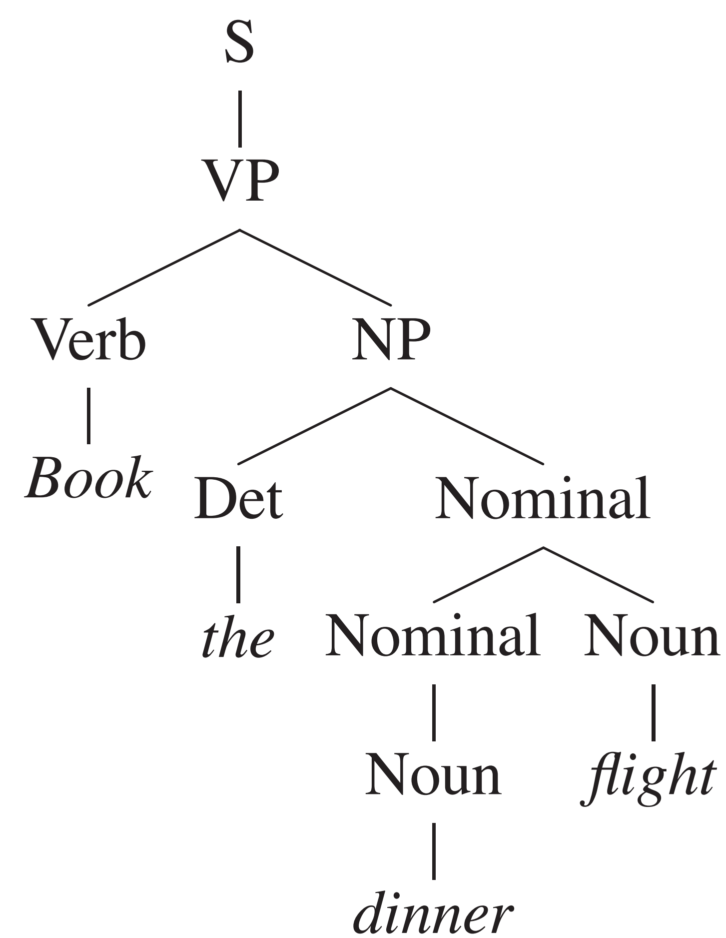
Gives us a tree, as before, with a sentence as the yield

HOW LIKELY IS A TREE?

- Given a tree, we can compute its probability
  - Decomposes into probability of each production

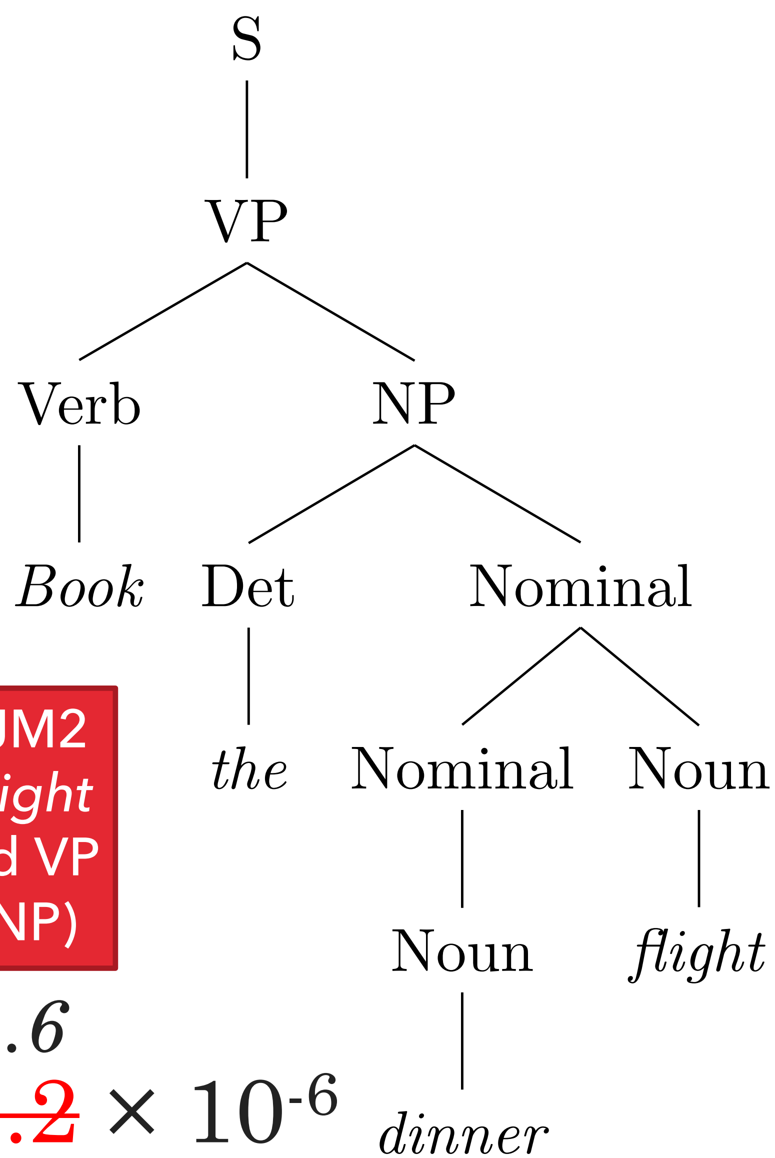
$$P(T) = \prod_{i=1}^{n} P(\mathrm{RHS}_i | \mathrm{LHS}_i)$$

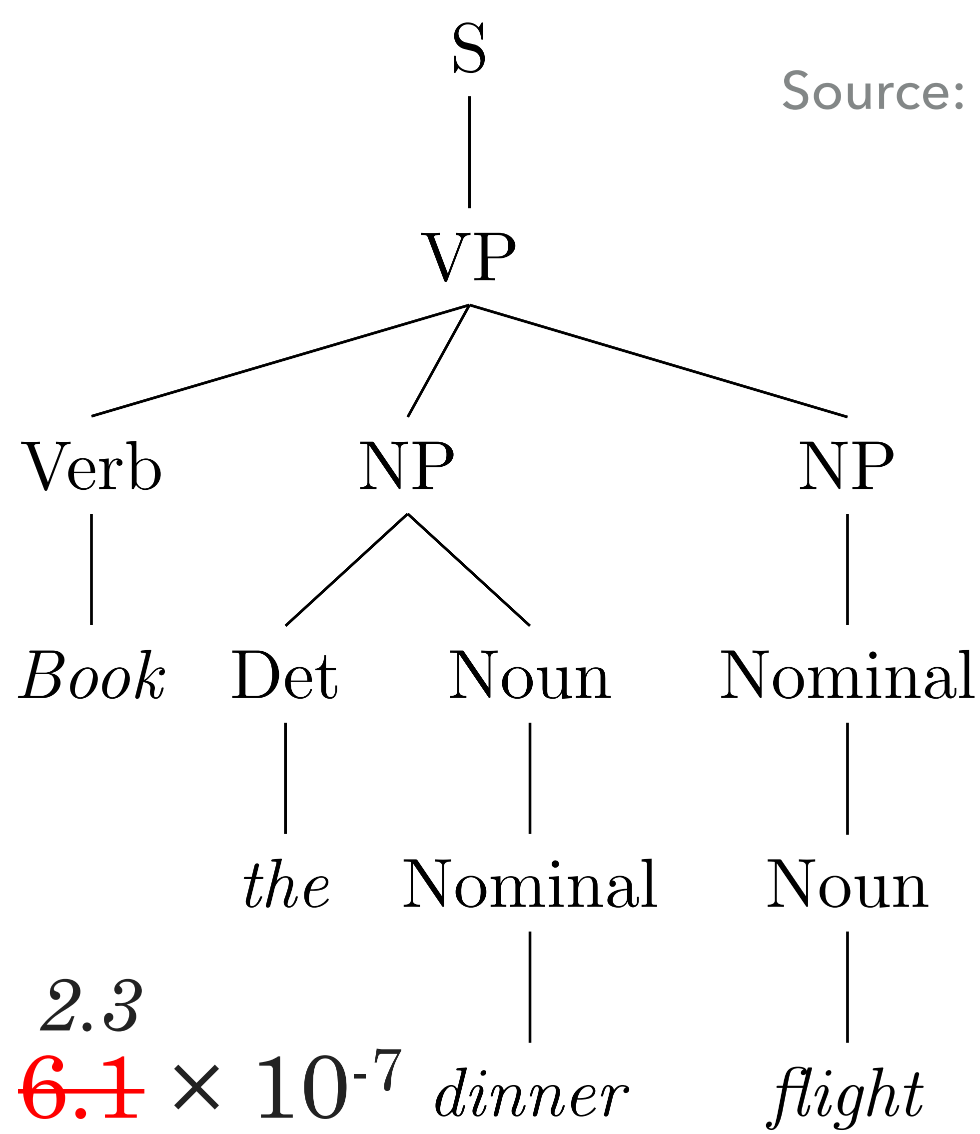
- E.g., for tree on right,
  - P(T) = P(S → VP) ×
    P(VP → Verb NP) ×
    P(Verb → *Book*) ×
    P(NP → Det Nominal) ×
    P(Det → *the*) ×
    P(Nominal → Nominal Noun) ×
    P(Noun → *dinner*) ×
    P(Noun → *flight*) = 2.2×10^{-6}



RESOLVING PARSE AMBIGUITY

- Can select between different trees based on P(T)





Source: JM2 Ch 14

Mistakes in JM2 (regarding *flight* vs *flights*, and VP → Verb NP NP)

- $P = \cancel{2.2}\; 1.6 \times 10^{-6}$ $\qquad$ $P = \cancel{6.1}\; 2.3 \times 10^{-7}$

- Note that some structures are the same (S → VP, Verb → *Book*…)

PARSING PCFGS

- Instead of selecting between two trees, can we select a tree from the set of all possible trees?
- Before we looked at
  - CYK and Early
  - for unweighted grammars (CFGs)
  - finds **all possible trees**
- But there are often 1000s, many completely nonsensical
- Can we solve for the **most probable tree**?

$$T^* = \arg\max_{T \text{ s.t. } yield(T)=S} P(T)$$

CYK FOR PCFGS

- Similar process to standard CYK
- Convert grammar to Chomsky Normal Form (CNF)
  - E.g., $\quad \mathrm{VP} \rightarrow \mathrm{Verb\ NP\ NP} \quad [0.05]$

    becomes $\quad \mathrm{VP} \rightarrow \mathrm{Verb\ X} \quad [??]$

    $\mathrm{X} \rightarrow \mathrm{NP\ NP} \quad [??]$

    where X is a new symbol.
  - But what happens to the probability?
- Issues with unary productions *(see ipython notebook)*

CYK FOR PCFGS

Source: JM2 Ch 14

function parse-CYK(w, G):

- **for** j **in** 1 … |**w**|
  - **for all** A → w_j **in** grammar
    - **set** chart[j-1,j,A] = P(A → w_j)
  - **for** i **in** j-1 … 0 (descending)
    - **for** k **in** i+1 .. j-1
      - **for all** A → B C **in** grammar
        such that chart[i,k,B] > 0 **and** chart[k,j,C] > 0
        - prob = P(A → B C) chart[i,k,B] chart[k,j,C]
        - **if** prob > chart[i,j,A] **then**
          - chart[i,j,A] = prob
          - back[i,j,a] = (k, B, C)
- **return build-tree**(back, |**w**|, S)

Initialise the table with preterminal expansions

i = left, k = middle, j = right

Find maximum scoring decomposition of span [i, j] split into i < k < j

Build tree by tracing backpointers

ILLUSTRATION

- Insert preterminal productions of type POS → *word*

Book	*the*	*dinner*	*flight*
Verb [0.3] Noun [0.1] VP [0.105] Nominal [0.075] NP [0.01125] S [0.00525] [0,1]	[0,2]	[0,3]	[0,4]
	Det [0.6] [1,2]	[1,3]	[1,4]
		Noun [0.1] Nominal [0.075] NP [0.075] [2,3]	[2,4]
			Noun [0.3] Nominal [0.225] NP [0.03375] [3,4]

Verb → *book* [0.3]
Noun → *book* [0.1]
Noun → *dinner* [0.1]
Noun → *flight* [0.3]

VP → Verb [0.35]
S → VP [0.05]
Nominal → Noun [0.75]
NP → Nominal [0.15]

ILLUSTRATION

NP → Det Nominal [0.20]

score $= 0.6 \times 0.075 \times 0.2$

$= 0.09$

Book	*the*	*dinner*	*flight*
Verb [0.3] Noun [0.1] VP [0.105] Nominal [0.075] NP [0.01125] S [0.00525] [0,1]	[0,2]	[0,3]	[0,4]
	Det **[0.6]** [1,2]	NP [0.09] [1,3]	[1,4]
		Noun [0.1] **Nominal** **[0.075]** NP [0.01125] [2,3]	[2,4]
			Noun [0.3] Nominal [0.225] NP [0.03375] [3,4]

ILLUSTRATION

Nominal → Nominal Noun [0.20]
score = 0.075 x 0.3 x 0.2
= 0.0045

Book	*the*	*dinner*	*flight*
Verb [0.3] Noun [0.1] VP [0.105] Nominal [0.075] NP [0.01125] S [0.00525] [0,1]	[0,2]	[0,3]	[0,4]
	Det [0.6] [1,2]	NP [0.09] [1,3]	[1,4]
		Noun [0.1] **Nominal [0.075]** NP [0.01125] [2,3]	Nominal [0.0045] [2,4]
			Noun [0.3] Nominal [0.225] NP [0.03375] [3,4]

ILLUSTRATION

NP → Det Nominal [0.20]
score = 0.6 x 0.0045 x 0.2
= 0.00054

Book	*the*	*dinner*	*flight*
Verb [0.3] Noun [0.1] VP [0.105] Nominal [0.075] NP [0.01125] S [0.00525] [0,1]	[0,2]	[0,3]	[0,4]
	Det [0.6] [1,2]	NP [0.09] [1,3]	NP [0.00054] [1,4]
		Noun [0.1] Nominal [0.075] NP [0.01125] [2,3]	**Nominal [0.0045]** [2,4]
			Noun [0.3] Nominal [0.225] NP [0.03375] [3,4]

ILLUSTRATION

VP → Verb NP [0.20]
score = 0.3 x 0.00054 x 0.2
= 0.000032

Book	*the*	*dinner*	*flight*
Verb [0.3] Noun [0.1] VP [0.105] Nominal [0.075] NP [0.01125] S [0.00525] [0,1]	[0,2]	[0,3]	**VP [0.000032]** [0,4]
	Det [0.6] [1,2]	NP [0.09] [1,3]	**NP [0.00054]** [1,4]
		Noun [0.1] Nominal [0.075] NP [0.01125] [2,3]	**Nominal [0.0045]** [2,4]
			Noun [0.3] Nominal [0.225] NP [0.03375] [3,4]

ILLUSTRATION: COMPETING ANALYSIS

- Length j = 4

X → NP NP [1]
VP → Verb X [0.05]

Book	*the*	*dinner*	*flight*
Verb [0.3] Noun [0.1] VP [0.105] Nominal [0.075] NP [0.01125] S [0.00525] [0,1]	[0,2]	[0,3]	**VP [0.00015]** [0,4]
	Det [0.6] [1,2]	**NP [0.09]** [1,3]	NP [0.00054] X [0.003] [1,4]
		Noun [0.1] Nominal [0.075] NP [0.01125] [2,3]	Nominal [0.0045] [2,4]
			Noun [0.3] Nominal [0.225] **NP [0.03375]** [3,4]

outscores existing analysis for [0,4; VP]

PROB CYK: RETRIEVING THE PARSES

- S in the top-right corner of parse table indicates success
- Retain back-pointer to best analysis
  - for each chart cell, store the split point and the non-terminal for the left and right children
- To get parse(s), follow pointers back for each match
- Convert back from CNF by removing new non-terminals

COMPLEXITY OF CYK

- What's the space and time complexity of this algorithm?
  - in terms of n the length of the input sentence

PROBLEMS WITH (P)CFGS

- **poor independence assumptions:** rewrite decisions made independently, whereas inter-dependence is often needed to capture global structure.
    - E.g., NP $\rightarrow$ PRP used often as subject (first NP), much less often as object (second NP)
- **lack of lexical conditioning:** non-terminals representation behaviour of the actual words, but are much too coarse. Problems with
    - preposition attachment ambiguity;
    - subcategorisation ([*forgot NP*] vs [*forgot S*]);
    - coordinate structure ambiguities (*dogs in houses and cats)*

PP ATTACHMENT

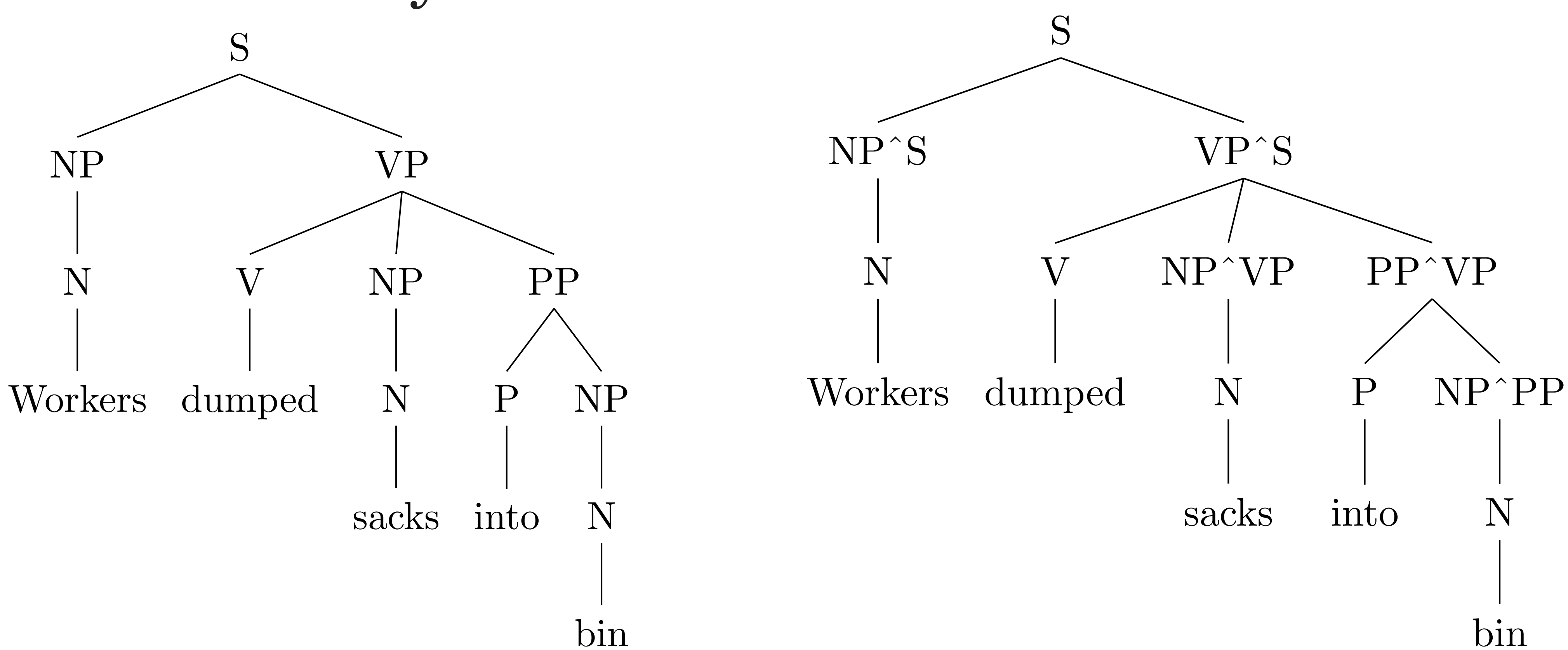
- Consider sentences (PP shown bracketed)

  (1) *Workers dumped sacks [into bin].*
  (2) *Fishermen caught tons [of herring].*

- Both have same POS tag sequence, but different structure
  - PP attaches either high (to the verb) or low (to the noun)
  - how to make this attachment decision? Difference between the two analyses minor:
    - VP → Verb NP PP vs. VP → Verb NP; NP → NP PP
- The probabilities of these three rules drive attachment, *irrespective of the verb, preposition and noun*

ONE SOLUTION: PARENT CONDITIONING

- Make non-terminals more explicit by incorporating parent symbol into each symbol



- NP^S represents subject position; while NP^VP denoting object position

- Helps to specify general tags, used for a number of very different purposes, e.g., *He said **that** I saw* …

ANOTHER SOLUTION: LEXICALISATION

- Uses notion of **head word**
  - the most salient child of a constituent, usually the noun in a NP, verb in a VP etc
- Incorporate head words into productions, such that the most important links between words is captured
  - E.g., $VP \rightarrow VBD\ NP\ PP \quad \Rightarrow$
    $VP(dumped) \rightarrow VBD(dumped)\ NP(sacks)\ PP(into)$
  - rule captures correlations between head tokens of phrases
- Learning probabilities somewhat more involved, to avoid sparsity problems (e.g., zero probabilities)

A FINAL WORD

- PCFGs widely used, and some of the best performing parsers available. E.g.,
  - Collins parser, Berkeley parser, Stanford parser
  - all use some form of lexicalisation or change to non-terminal set with CFGs

REQUIRED READING

- J&M2 Ch. 14, $\leq$ 14.6 (skip 14.1.2)