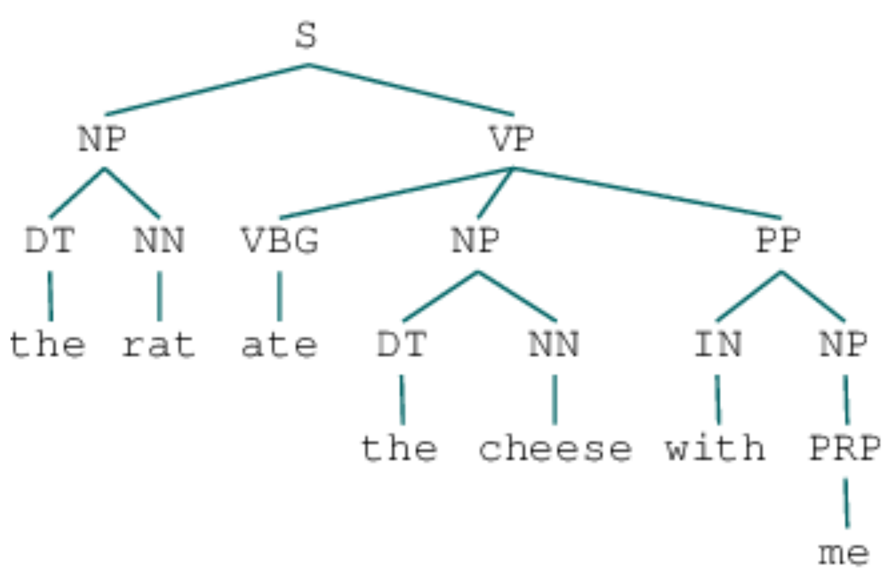
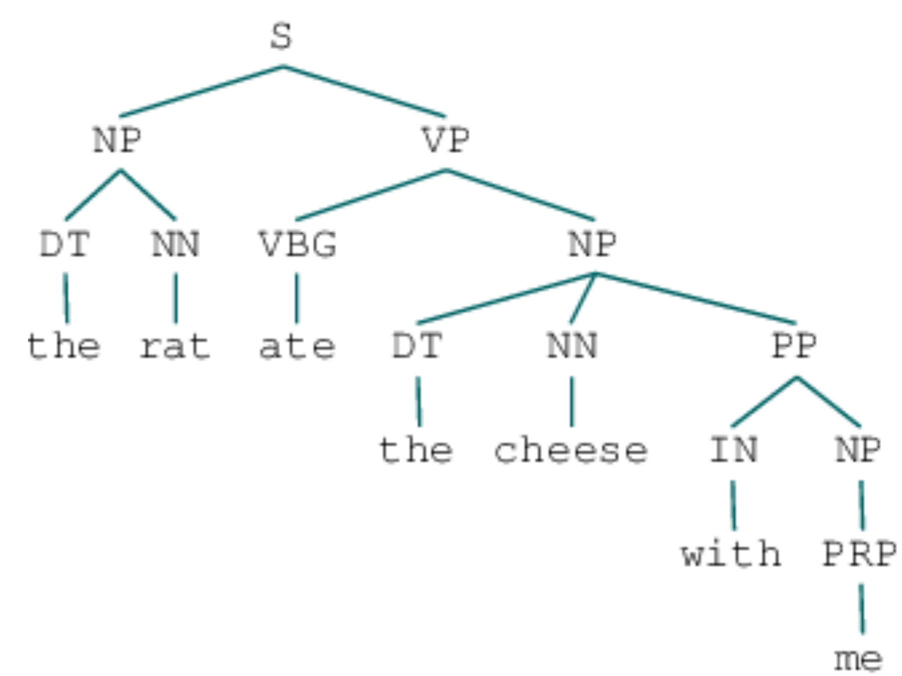
Probabilistic Parsing

COMP90042 LECTURE 18

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 *conditional*
  - Pr(RHS | LHS)
- Consequently they:
  - must be positive values, between 0 and 1
  - must sum to one for given LHS
- E.g.,
  - NN → aadvark [p = 0.0003]
  - NN → cat [p = 0.02]
  - NN → leprechaun [p = 0.0001]
  - $\sum_x \Pr(NN \rightarrow x \,/\, NN) = 1$

A Probabilistic grammar

Grammar		Lexicon
$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.10] | a [.30] | the [.60]
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$ [.10] | flights [.30]
$S \rightarrow VP$	[.05]	| $meal$ [.015] | $money$ [.05]
$NP \rightarrow Pronoun$	[.35]	| $flight$ [.40] | $dinner$ [.10]
$NP \rightarrow Proper\text{-}Noun$	[.30]	$Verb \rightarrow book$ [.30] | $include$ [.30]
$NP \rightarrow Det\ Nominal$	[.20]	| $prefer$ [.40]
$NP \rightarrow Nominal$	[.15]	$Pronoun \rightarrow I$ [.40] | she [.05]
$Nominal \rightarrow Noun$	[.75]	| me [.15] | you [.40]
$Nominal \rightarrow Nominal\ Noun$	[.20]	$Proper\text{-}Noun \rightarrow Houston$ [.60]
$Nominal \rightarrow Nominal\ PP$	[.05]	| NWA [.40]
$VP \rightarrow Verb$	[.35]	$Aux \rightarrow does$ [.60] | can [40]
$VP \rightarrow Verb\ NP$	[.20]	$Preposition \rightarrow from$ [.30] | to [.30]
$VP \rightarrow Verb\ NP\ PP$	[.10]	| on [.20] | $near$ [.15]
$VP \rightarrow Verb\ PP$	[.15]	| $through$ [.05]
$VP \rightarrow Verb\ NP\ NP$	[.05]	
$VP \rightarrow VP\ PP$	[.15]	
$PP \rightarrow Preposition\ NP$	[1.0]	

Source JM3, Fig 12.1

Stochastic Generation with PCFGs

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

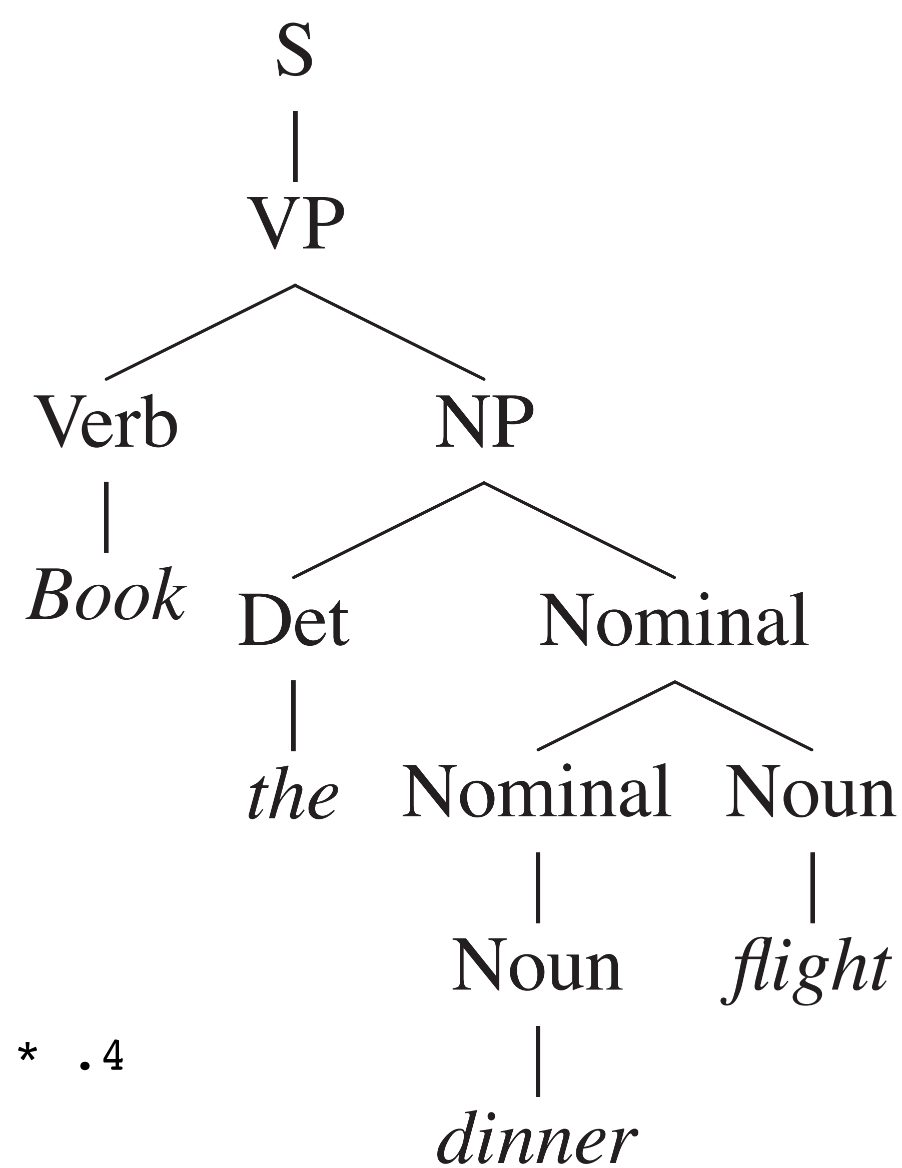
1. Start with S, the sentence symbol
2. Choose a rule with S as the LHS
   * **Randomly select a RHS** according to Pr(RHS | LHS) e.g., S → VP
   * Apply this rule, e.g., substitute VP for S
3. Repeat step 2 for each non-terminal in the string (here, VP)
4. Stop when no non-terminals remain

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
- E.g., for tree on right,
  - P(tree) =

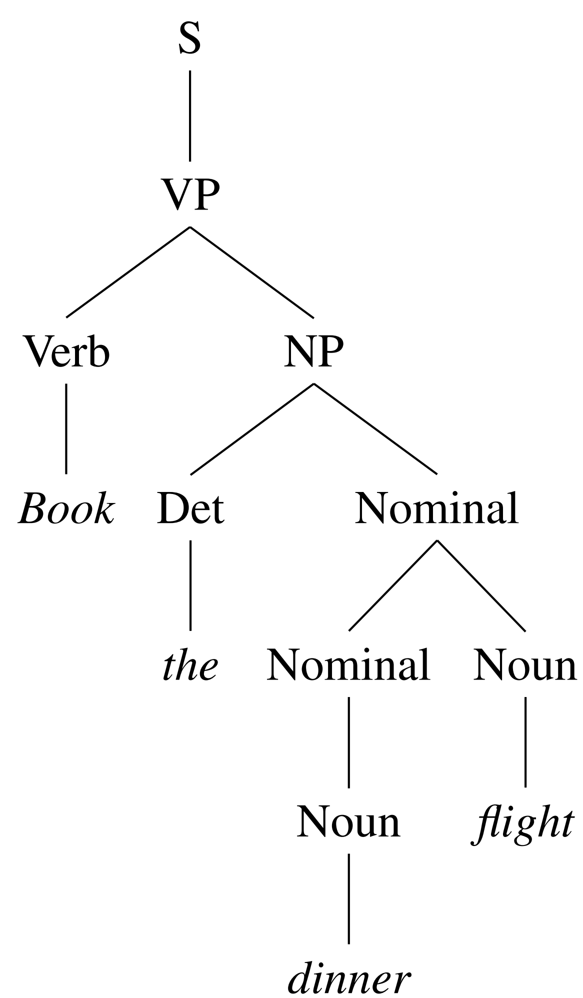
    P(S → VP) ×
    P(VP → Verb NP) ×
    P(Verb → *Book*) ×
    P(NP → Det Nominal) ×
    P(Det → *the*) ×
    P(Nominal → Nominal Noun) ×
    P(Nominal → Noun) ×
    P(Noun → *dinner*) ×
    P(Noun → *flight*) = 2.16×10^{-6}

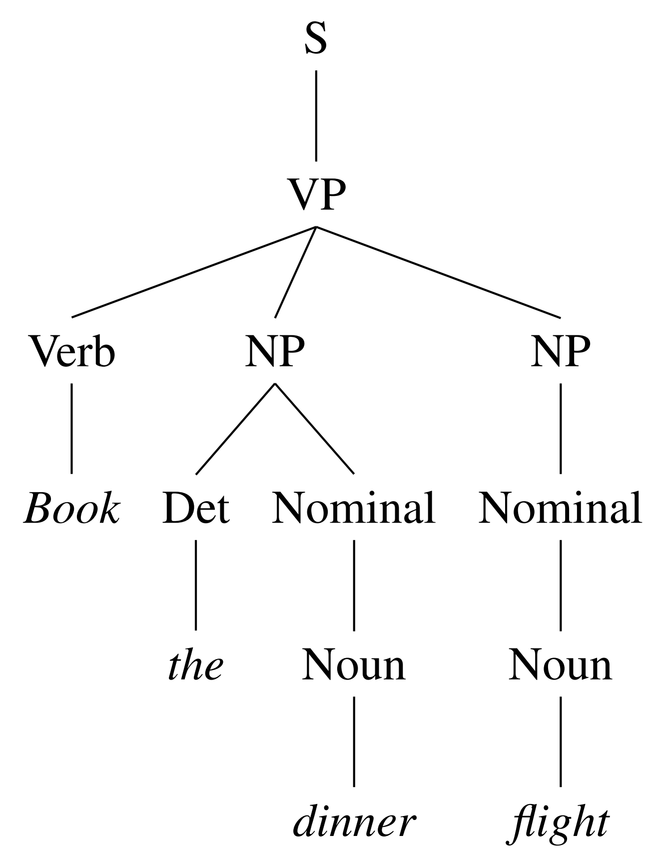



```
I.e., .05 * .2 * .3 * .2 * .6 * .2 * .75 * .1 * .4
```

Resolving parse ambiguity

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





Source: JM3
Fig 12.2

Mistakes in textbook

.05*.05*.20*.15*.75*.75*.30*.60*.10*.40

- $P = 2.16 \times 10^{-6}$ $P = 3.04 \times 10^{-7}$

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

$$\arg\max_{T \text{ s.t. } \mathrm{yield}(T)=\mathbf{w}} P(T)$$

- Can we solve for the **most probable tree**

CYK for PCFGS

- CYK finds *all trees* for a sentence; we want best tree
- Prob. CYK follows similar process to standard CYK
- Convert grammar to Chomsky Normal Form (CNF)
  - E.g., VP → Verb NP NP [0.05]

    becomes VP → Verb NP+NP [_____]

    NP+NP → NP NP [_____]

    where NP+NP is a new symbol.
- Issues with unary productions *(see ipython notebook)*

Prob. CYK

function PROBABILISTIC-CKY(*words,grammar*) **returns** most probable parse and its probability

for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
  for all $\{ A \mid A \rightarrow words[j] \in grammar\}$
    $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
  for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
    for $k \leftarrow i+1$ **to** $j-1$ **do**
      for all $\{ A \mid A \rightarrow BC \in grammar,$ **and** $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$
        if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**
          $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
          $back[i,j,A] \leftarrow \{k, B, C\}$
  return BUILD_TREE(*back*[1, LENGTH(*words*), *S*]), *table*[1, LENGTH(*words*), *S*]

Figure 12.3 The probabilistic CKY algorithm for finding the maximum probability parse

Source: JM3
Ch 12

chart now stores probabilities for each span and symbol

CYK can be thought of as storing all events with probability = 1

function CKY-PARSE(*words, grammar*) **returns** *table*

  for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{A \mid A \rightarrow words[j] \in grammar\}$
      $table[j-1, j] \leftarrow table[j-1, j] \cup A$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
      for $k \leftarrow i+1$ **to** $j-1$ **do**
        for all $\{A \mid A \rightarrow BC \in grammar$ **and** $B \in table[i,k]$ **and** $C \in table[k,j]\}$
          $table[i,j] \leftarrow table[i,j] \cup A$

Figure 11.5 The CKY algorithm.

validity test now looks to see that the child chart cells have non-zero probability

function PROBABILISTIC-CKY(*words,grammar*) **returns** most probable parse and its probability

  for $j \leftarrow$ **from** 1 **to** LENGTH(*words*) **do**
    for all $\{ A \mid A \rightarrow words[j] \in grammar\}$
      $table[j-1, j, A] \leftarrow P(A \rightarrow words[j])$
    for $i \leftarrow$ **from** $j-2$ **downto** 0 **do**
      for $k \leftarrow i+1$ **to** $j-1$ **do**
        for all $\{ A \mid A \rightarrow BC \in grammar,$
            and $table[i,k,B] > 0$ **and** $table[k,j,C] > 0 \}$
          if $(table[i,j,A] < P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C])$ **then**
            $table[i,j,A] \leftarrow P(A \rightarrow BC) \times table[i,k,B] \times table[k,j,C]$
            $back[i,j,A] \leftarrow \{k,B,C\}$
  return BUILD_TREE(*back*[1, LENGTH(*words*), *S*]), *table*[1, LENGTH(*words*), *S*]

Figure 12.3 The probabilistic CKY algorithm for finding the maximum probability parse

Instead of storing set of symbols, store the probability of best scoring tree fragment covering span [i,j] with root symbol A

Overwrite lower scoring analysis if this one is better, and record the best production.

Illustration

we	eat	sushi	with	chopsticks

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				
	V 1			

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

Illustration

	we	eat	sushi	with	chopsticks
1	NP 1/4				
2		V 1			
3			NP 1/8		
4					
5					

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				
	V 1	VP 1/8 * 1 * ½ = 1/16		
		NP 1/8		

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				
	V 1	VP 1/16		
		NP 1/8		
			IN 1	

S	→ NP VP		1
NP	→ NP PP		½
	→ we		¼
	→ sushi		1/8
	→ chopsticks		1/8
PP	→ IN NP		1
IN	→ with		1
VP	→ V NP		½
	→ VP PP		¼
	→ MD V		¼
V	→ eat		1

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				
	V 1	VP 1/16		
		NP 1/8		
			IN 1	
				NP 1/8

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				
	V 1	VP 1/16		
		NP 1/8		
			IN 1	PP 1/8
				NP 1/8

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				
	V 1	VP 1/16		
		NP 1/8		NP 1/128
			IN 1	PP 1/8
				NP 1/8

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				
	V 1	VP 1/16		VP ½ * 1 * 1/128 = 1/256
		NP 1/8		NP 1/128
			IN 1	PP 1/8
				NP 1/8

S → NP VP 1

NP → NP PP ½

→ we ¼

→ sushi 1/8

→ chopsticks 1/8

PP → IN NP 1

IN → with 1

VP → V NP ½

→ VP PP ¼

→ MD V ¼

V → eat 1

1/256 > 1/512
→ this is a better analysis, so replace old value

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				
	V 1	VP 1/16		VP 1/256
		NP 1/8		NP 1/128
			IN 1	PP 1/8
				NP 1/8

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

Illustration

we	eat	sushi	with	chopsticks
NP 1/4				S 1/4096
	V 1	VP 1/16		VP 1/256
		NP 1/8		NP 1/128
			IN 1	PP 1/8
				NP 1/8

S	→ NP VP	1
NP	→ NP PP	½
	→ we	¼
	→ sushi	1/8
	→ chopsticks	1/8
PP	→ IN NP	1
IN	→ with	1
VP	→ V NP	½
	→ VP PP	¼
	→ MD V	¼
V	→ eat	1

Example & grammar from E18 Chapter 10

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 → 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

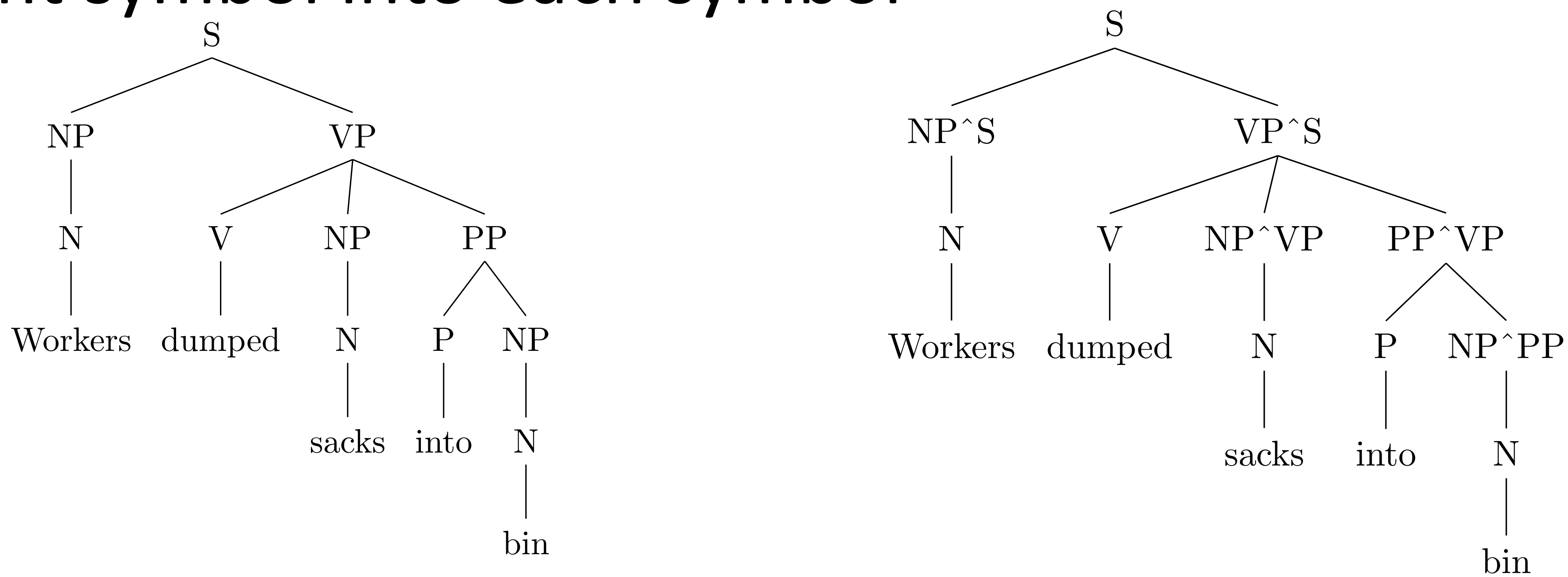
- Consider sentences (PP shown bracketed)
  (1) *Workers dumped sacks [into bin].*
  (2) *Fishermen caught tons [of herring].*

- 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 comes down to rules:
    - 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

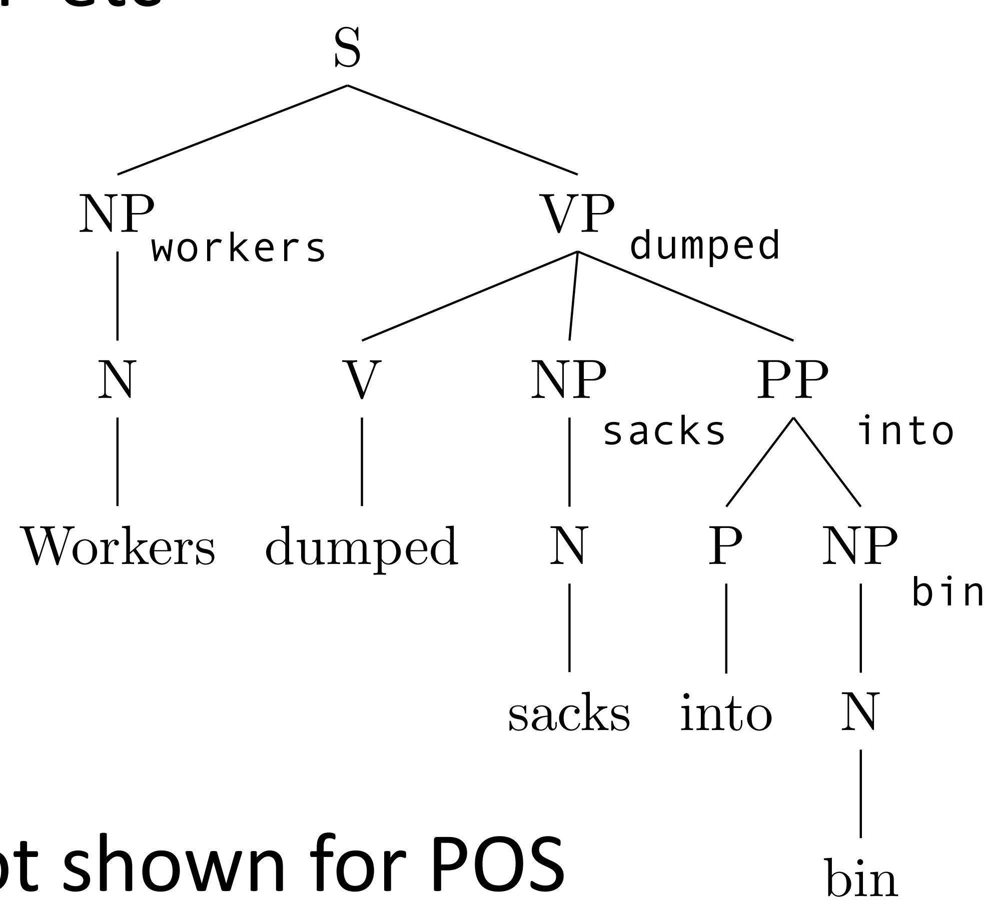
- Make non-terminals more explicit by incorporating parent symbol into each symbol



- NP^S represents subject position (left); NP^VP denotes object position (right); PP^VP is different to PP^NP
- Helps to make general tags more specific, used for a number of different purposes, e.g., *He said **that** I saw …*

Another solution: Head Lexicalisation

- Record **head word** with parent symbols
  - the most salient child of a constituent, usually the noun in a NP, verb in a VP etc



  - head words not shown for POS
  - *VP → V NP PP* ⇒
    VP(dumped) → V(dumped) NP(sacks) PP(into)

Head lexicalisation

- Incorporate head words into productions, such that the most important links between words is captured
  - rule captures correlations between head tokens of phrases
- Grammar symbol inventory expands massively!
  - Many of the productions much too specific, seen very rarely
  - Learning more involved to avoid sparsity problems (e.g., zero probabilities)

A final word

- PCFGs widely used, and are 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
- But not used universally, a competing method is to treat parsing as a sequential process of “transitions”
  - next week, dependency parsing

Required Reading

- J&M3 Ch. 12 – 12.6
  - Warning: several errors in the computations, and grammar used for PCYK is not in CNF