Probabilistic Context-Free Grammars (PCFGs)

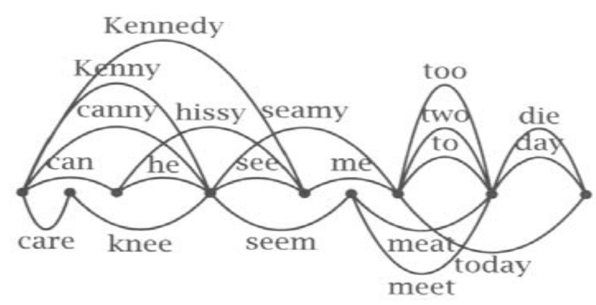
Berlin Chen 2003

References:

1. Speech and Language Processing, chapter 12
2. Foundations of Statistical Natural Language Processing, chapters 11, 12

Parsing for Disambiguation

- At least three ways to use probabilities in a parser
  - **Probabilities for choosing between parses**
    - Choose from among the many parses of the input sentence which ones are most likely
  - **Probabilities for speedier parsing** Parsing as Search
    - Use probabilities to order or prune the search space of a parser for finding the best parse more quickly
  - **Probabilities for determining the sentence**
    - Use a parser as a language model over a word lattice in order to determine a sequence of words that has the highest probability



Parsing for Disambiguation

- The integration of sophisticated structural and probabilistic models of syntax is at the very cutting edge of the field
  - For the non-probabilistic syntax analysis
    - The context-free grammar (CFG) is the standard
  - For the probabilistic syntax analysis
    - No single model has become a standard
    - A number of probabilistic augmentations to context-free grammars
      - Probabilistic CFG with the CYK algorithm
      - Probabilistic lexicalized CFG
      - Dependency grammars
      - …….

Definition of the PCFG

Booth, 1969

- A PCFG G has five parameters

syntactic categories
lexical categories

1. A set of non-terminal symbols (or "variables") N
2. A set of terminal symbols $\sum$ (disjoint from N) words
3. A set of productions P, each of the form $A \rightarrow \beta$, where A is a non-terminal symbol and β is a string of symbols from the infinite set of strings $(\sum \cup N)^*$
4. A designated start symbol S (or N^1)
5. Each rule in P is augmented with a conditional probability assigned by a function D

$$A \rightarrow \beta \quad [prob.]$$

$P(A \rightarrow \beta)$ or $P(A \rightarrow \beta \mid A)$ $\Rightarrow$ $\forall A \quad \sum_{\beta} P(A \rightarrow \beta) = 1$

- A PCFG $G=(N, \sum, P, S, D)$

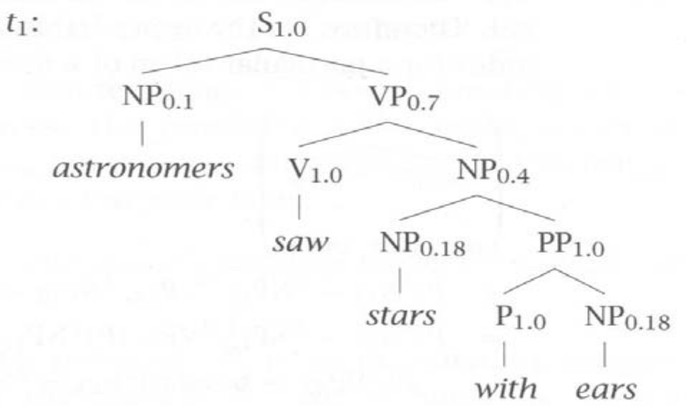
An Example Grammar

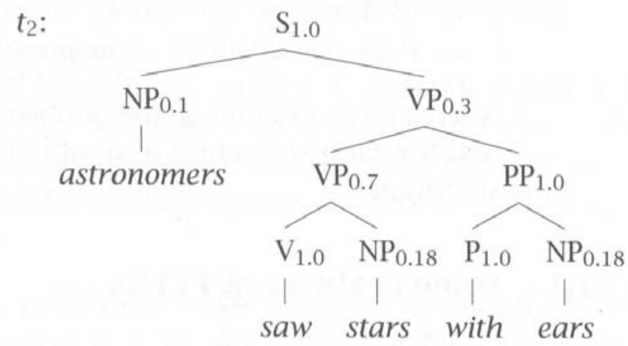
S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1

Table 11.2 A simple Probabilistic Context Free Grammar (PCFG). The nonterminals are S, NP, PP, VP, P, V. We adopt the common convention whereby the start symbol N^1 is denoted by S. The terminals are the words in italics. The table shows the grammar rules and their probabilities. The slightly unusual NP rules have been chosen so that this grammar is in Chomsky Normal Form, for use as an example later in the section.

Parse Trees

- Input: ***astronomers saw stars with ears***





$$
\begin{aligned}
P(t_1) &= 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0009072 \\
P(t_2) &= 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0006804 \\
P(w_{15}) &= P(t_1) + P(t_2) = 0.0015876
\end{aligned}
$$

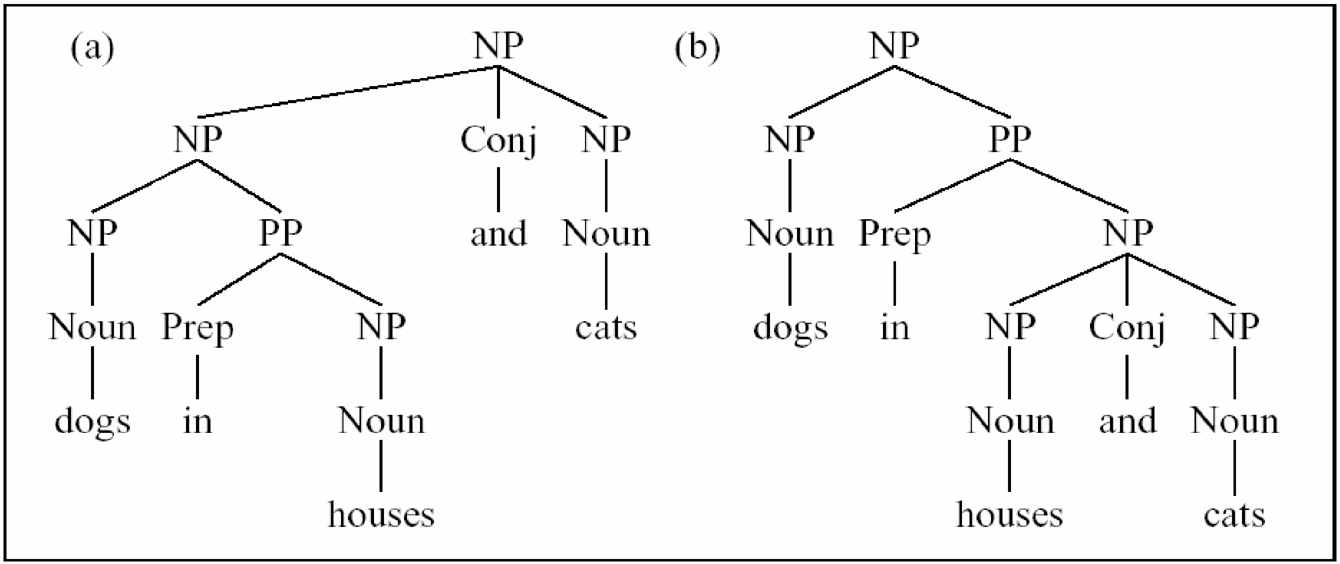
Figure 11.1 The two parse trees, their probabilities, and the sentence probability. This is for the sentence *astronomers saw stars with ears*, according to the grammar in table 11.2. Nonterminal nodes in the trees have been subscripted with the probability of the local tree that they head.

The probability of a particular parse is defined as the product of the probabilities of all the rules used to expand each node in the parse tree

– An instance of **PP-attachment ambiguity**

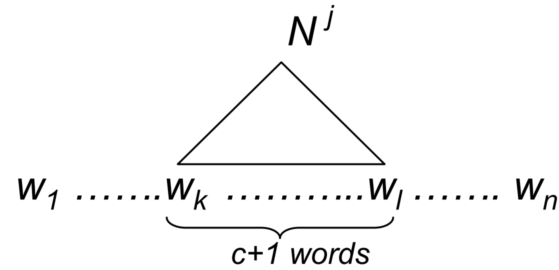
Parse Trees

- Input: *dogs in houses and cats*



  - An instance of **coordination ambiguity**
    - Which one is correct ?
    - However, the PCFG will assign the identical probabilities to the two parses

Basic Assumptions



- Place Invariance
  - The probability of a subtree does not depend on where in the string the words it dominates are

$$\forall k \quad P\left(N^j_{k(k+c)} \rightarrow \zeta\right) = P\left(N^j \rightarrow \zeta\right)$$

word positions in the input string

- Context free
  - The probability of a subtree does not depend on words not dominated by the subtree

$$P\left(N^j_{kl} \rightarrow \zeta \middle| \text{anything outside } k \text{ through } l\right) = P\left(N^j_{kl} \rightarrow \zeta\right)$$

- Ancestor free
  - The probability of a subtree does not depend on nodes in the derivation outside the subtree

$$P\left(N^j_{kl} \rightarrow \zeta \middle| \text{any ancestor outside } N^j_{kl}\right) = P\left(N^j_{kl} \rightarrow \zeta\right)$$

Basic Assumptions

- Example

$$P\left(\begin{array}{ccc} & {}^{1}\mathrm{S} & \\ {}^{2}\mathrm{NP} & & {}^{3}\mathrm{VP} \\ \textit{the}\quad\textit{man} & & \textit{snores} \end{array}\right)$$

chain rule

$$= P({}^{1}\mathrm{S}_{13} \rightarrow {}^{2}\mathrm{NP}_{12}\ {}^{3}\mathrm{VP}_{33}, {}^{2}\mathrm{NP}_{12} \rightarrow \textit{the}_1\ \textit{man}_2, {}^{3}\mathrm{VP}_{33} \rightarrow \textit{snores}_3)$$

context-free & ancestor-free assumptions

$$= P({}^{1}\mathrm{S}_{13} \rightarrow {}^{2}\mathrm{NP}_{12}\ {}^{3}\mathrm{VP}_{33}) P({}^{2}\mathrm{NP}_{12} \rightarrow \textit{the}_1\ \textit{man}_2 | {}^{1}\mathrm{S}_{13} \rightarrow {}^{2}\mathrm{NP}_{12}\ {}^{3}\mathrm{VP}_{33})$$
$$P({}^{3}\mathrm{VP}_{33} \rightarrow \textit{snores}_3 | {}^{1}\mathrm{S}_{13} \rightarrow {}^{2}\mathrm{NP}_{12}\ {}^{3}\mathrm{VP}_{33}, {}^{2}\mathrm{NP}_{12} \rightarrow \textit{the}_1\ \textit{man}_2)$$

Place-invariant assumption

$$= P({}^{1}\mathrm{S}_{13} \rightarrow {}^{2}\mathrm{NP}_{12}\ {}^{3}\mathrm{VP}_{33}) P({}^{2}\mathrm{NP}_{12} \rightarrow \textit{the}_1\ \textit{man}_2) P({}^{3}\mathrm{VP}_{33} \rightarrow \textit{snores}_3)$$
$$= P(\mathrm{S} \rightarrow \mathrm{NP\ VP}) P(\mathrm{NP} \rightarrow \textit{the man}) P(\mathrm{VP} \rightarrow \textit{snores})$$

Some Features of PCFGs

- PCFGs give some idea (probabilities) of the plausibility of different parses
  – But the probability estimates are based purely on **structural factors** and not **lexical factors**
- PCFGs are good for grammar induction
  – PCFG can be learned from data, e.g. from bracketed (labeled) corpora
- PCFGs are robust
  – Tackle grammatical mistakes, disfluencies, and errors by ruling out nothing in the grammar, but by just giving implausible sentences a lower probability

Chomsky Normal Form

- Chomsky Normal Form (CNF) grammars only have **unary** and **binary** rules of the form

$$N^j \rightarrow N^r N^s$$ For syntactic categories

$$N^j \rightarrow w^k$$ For lexical categories

- The parameters of a PCFG in CNF are

$$P\left(N^i \rightarrow N^r N^s \middle| G\right)$$

n^3 matrix of parameters (when n nonterminals)

$$P\left(N^i \rightarrow w^k \middle| G\right)$$

nV matrix of parameters (when n nonterminals and V terminals)

n^3+nV parameters

$$\sum_{r,s} P\left(N^j \rightarrow N^r N^s\right) + \sum_k P\left(N^i \rightarrow w^k\right) = 1$$

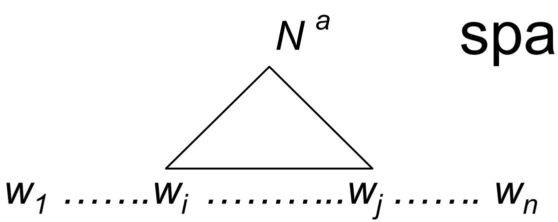
- Any CFG can be represented by a weakly equivalent CFG in CNF
  - "*weakly equivalent*" : "*generating the same language*"
    - But do not assign the same phrase structure to each sentence

CYK Algorithm

Ney, 1991

Collins, 1999

- CYK (Cocke-Younger-Kasami) algorithm
  - A **bottom-up** parser using the dynamic programming table
  - Assume the PCFG is in Chomsky normal form (CNF)
- Definition
  - $w_1 \ldots w_n$: an input string composed of *n* words
  - w_{ij}: a string of words from words *i* to *j*
  - π [*i*, *j*, *a*]: a table entry holds the maximum probability for a constituent with non-terminal index *a* spaning words $w_i \ldots w_j$



CYK Algorithm

- Fill out the table entries by induction
  - **Base case**
    - Consider the input strings of length one (i.e., each individual word w_i) $P(A \rightarrow w_i)$
    - Since the grammar is in CNF, $A \overset{*}{\Rightarrow} w_i \quad \text{iff} \quad A \rightarrow w_i$

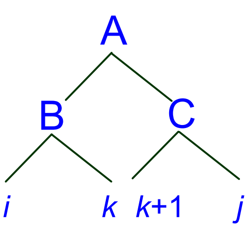
A must be a lexical category

  - **Recursive case**
    - For strings of words of length > 1,

$$A \overset{*}{\Rightarrow} w_{ij} \quad \text{iff there is at least one rule } A \rightarrow BC$$
$$\text{where } B \text{ derives the first } k\text{-}i+1 \text{ symbols and}$$
$$C \text{ derives the last } j\text{-}k \text{ symbols and}$$

A must be a syntactic category

Choose the maximum among all possibilities



    - Compute the probability by multiplying together the probabilities of these two pieces (note that they have been calculated in the recursion)

CYK Algorithm

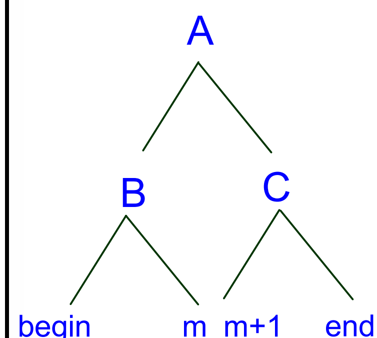
Finding the most Likely parse for a sentence

```
function CYK(words,grammar) returns The most probable parse
                                   and its probability

  Create and clear π[num_words, num_words, num_nonterminals]   ← set to zero

  # base case
  for i ← 1 to num_words
     for A ← 1 to num_nonterminals
        if (A → w_i) is in grammar then
           π[i, i, A] ← P(A → w_i)

  # recursive case   ← on the word-span
  for span ← 2 to num_words
     for begin ← 1 to num_words − span + 1
        end ← begin + span − 1
        for m = begin to end − 1
           for A = 1 to num_nonterminals
           for B = 1 to num_nonterminals
           for C = 1 to num_nonterminals
              prob = π[begin, m, B] × π[m + 1, end, C] × P(A→B C)
              if (prob > π[begin, end, A]) then
                 π[begin, end, A] = prob
                 back[begin, end, A] = {m, B, C}   ← bookkeeping
  return build_tree(back[1, num_words, 1]), π[1, num_words, 1]
```

Figure 12.3 The Probabilistic CYK algorithm for finding the maximum probability parse of a string of *num_words* words given a PCFG grammar with *num_rules* rules in Chomsky Normal Form (after Collins (1999) and Aho and Ullman (1972).) *back* is an array of back-pointers used to recover the best parse. The *build_tree* function is left as an exercise to the reader.

m-word input string
n non-terminals

$O(m^3n^3)$



Three Basic Problems for PCFGs

- What is the probability of a sentence w_{1m} according to a grammar

  G: $P(w_{1m}|G)$?

- What is the most likely parse for a sentence?

  $\text{argmax}_t\ P(t\ |w_{1m}, G)$

- How can we choose the rule probabilities for the grammar G that maximize the probability of a sentence?

  $\text{argmax}_G\ P(w_{1m}|G)$ Training the PCFG

The Inside-Outside Algorithm

Baker 1979

Young 1990

- A generalization of the forward-backward algorithm of HMMs
- A dynamic programming technique used to efficiently compute PCFG probabilities
  - Inside and outside probabilities in PCFG

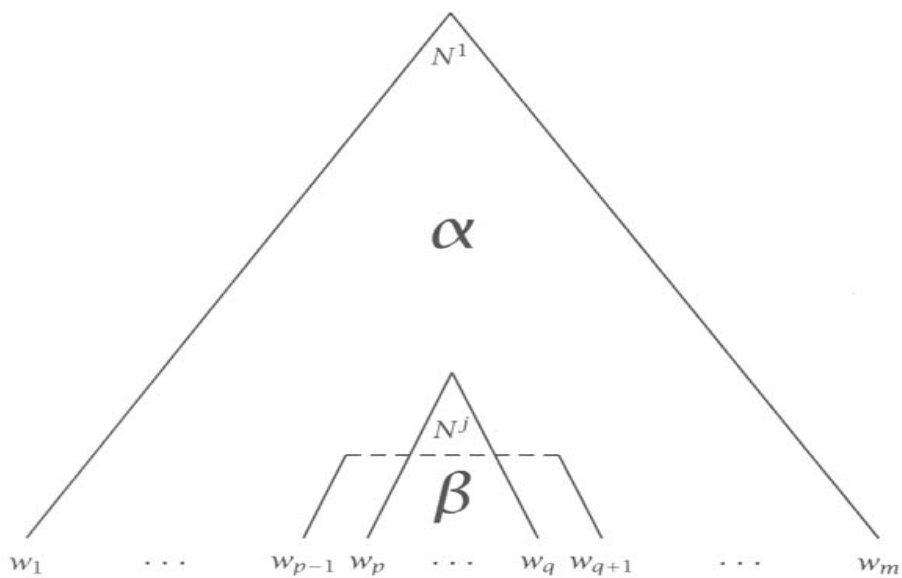


Figure 11.3 Inside and outside probabilities in PCFGs.

The Inside-Outside Algorithm

- Definition
  - Inside probability $\beta_j(p,q) = P\left(w_{pq} \middle| N^j_{pq}, G\right)$
    - The total probability of generating words $w_p \ldots w_q$ given that one is starting off with the nonterminal N^j
  - Outside probability $\alpha_j(p,q) = P\left(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} \middle| G\right)$
    - The total probability of beginning with the start symbol N_1 and generating the nonterminal N^j_{pq} and all the words outside $w_p \ldots w_q$

Problem 1: The Probability of a Sentence

- A PCFG with the **Chomsky Normal Form** was used here

- The total probability of a sentence expressed by the **inside algorithm**

$$P(w_{1m}|G) = P(N^1 \Rightarrow w_{1m}|G) = P(w_{1m}|N^1_{1m}, G) = \beta_1(1, m)$$

- The probability of **the base case**

word-span=1

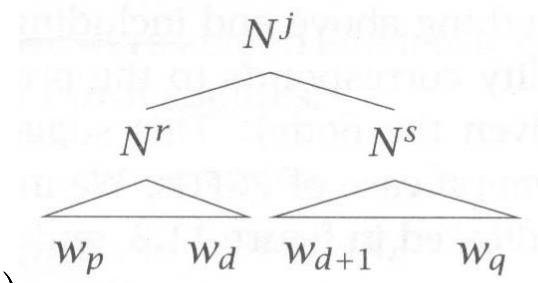
$$\beta_j(k,k) = P(w_k|N^j_{kk}, G) = P(N^j \rightarrow w_k|G) = P(w_{1m}|N^1_{1m}, G)$$

- Find the probabilities $\beta_j(p,q)$ **by induction** (or by recursion)

word-span > 1

Problem 1: The Probability of a Sentence

- Find the probabilities $\beta_j(p,q)$ by induction
  - A **bottom-up** version of calculation



$$\forall j,\ 1 \le p < q \le m$$

$$\beta_j(p,q) = P\left(N^j_{pq} \Rightarrow w_{pq} \middle| G\right) = P\left(w_{pq} \middle| N^j_{pq}, G\right)$$

$$= \sum_{r,s} \sum_{d=p}^{q-1} P\left(w_{pd}, N^r_{pd}, w_{(d+1)q}, N^s_{(d+1)q} \middle| N^j_{pq}, G\right)$$

$$= \sum_{r,s} \sum_{d=p}^{q-1} P\left(w_{pd}, N^r_{pd}, w_{(d+1)q}, N^s_{(d+1)q} \middle| N^j_{pq}, G\right)$$

chain rule

$$= \sum_{r,s} \sum_{d=p}^{q-1} P\left(N^r_{pd}, N^s_{(d+1)q} \middle| N^j_{pq}, G\right) \times P\left(w_{pd} \middle| \cancel{N^j_{pq}}, N^r_{pd}, \cancel{N^s_{(d+1)q}}, G\right)$$

$$\times P\left(w_{(d+1)q} \middle| \cancel{N^j_{pq}}, \cancel{N^r_{pd}}, N^s_{(d+1)q}, \cancel{w_{pd}}, G\right)$$

context-free & ancestor-free assumptions

$$= \sum_{r,s} \sum_{d=p}^{q-1} P\left(N^r_{pd}, N^s_{(d+1)q} \middle| N^j_{pq}, G\right) \times P\left(w_{pd} \middle| N^r_{pd}, G\right) \times P\left(w_{(d+1)q} \middle| N^s_{(d+1)q}, G\right)$$

Place-invariant assumption

$$= \sum_{r,s} \sum_{d=p}^{q-1} P\left(N^j \to N^r N^s\right) \times \beta_r(p,d) \times \beta_s(d+1,q)$$

the binary rule

Problem 1: The Probability of a Sentence

- Example

$S \rightarrow NP\ VP$	1.0	$NP \rightarrow NP\ PP$	0.4
$PP \rightarrow P\ NP$	1.0	$NP \rightarrow$ *astronomers*	0.1
$VP \rightarrow V\ NP$	0.7	$NP \rightarrow$ *ears*	0.18
$VP \rightarrow VP\ PP$	0.3	$NP \rightarrow$ *saw*	0.04
$P \rightarrow$ *with*	1.0	$NP \rightarrow$ *stars*	0.18
$V \rightarrow$ *saw*	1.0	$NP \rightarrow$ *telescopes*	0.1

end

begin

	1	2	3	4	5
1	$\beta_{NP} =\ 0.1$		$\beta_S =\ 0.0126$		$\beta_S =\ 0.0015876$
2		$\beta_{NP} =\ 0.04$ $\beta_V =\ 1.0$	$\beta_{VP} =\ 0.126$		$\beta_{VP} =\ 0.015876$
3			$\beta_{NP} =\ 0.18$		$\beta_{NP} =\ 0.01296$
4				$\beta_P =\ 1.0$	$\beta_{PP} =\ 0.18$
5					$\beta_{NP} =\ 0.18$
	astronomers	*saw*	*stars*	*with*	*ears*

$$\beta_{VP}(2,5) = P(VP \rightarrow V\ NP)\beta_V(2,2)\beta_{NP}(3,5) + P(VP \rightarrow VP\ PP)\beta_{VP}(2,3)\beta_{PP}(4,5)$$

0.015876 0.7 1.0 0.01296 0.3 0.126 0.18

$$\beta_S(1,5) = P(S \rightarrow NP\ VP)\beta_{NP}(1,1)\beta_{VP}(2,5)$$

0.0015867 1.0 0.1 0.015867

Problem 1: The Probability of a Sentence

- The total probability of a sentence expressed by the **outside algorithm**

chain rule

$$P(w_{1m}|G) = \sum_j P(w_{1m}, N^j_{kk}|G) = \sum_j P(w_{1(k-1)}, w_{kk}, w_{(k+1)m}, N^j_{kk}|G)$$

context-free & place-invariant assumptions

$$= \sum_j P(w_{1(k-1)}, N^j_{kk}, w_{(k+1)m}|G) P(w_{kk}|\cancel{w_{1(k-1)}}, N^j_{kk}, \cancel{w_{(k+1)m}}, G)$$

$$= \sum_j \alpha_j(k,k) P(N^j \rightarrow w_k|G)$$

N^j's are lexical categories

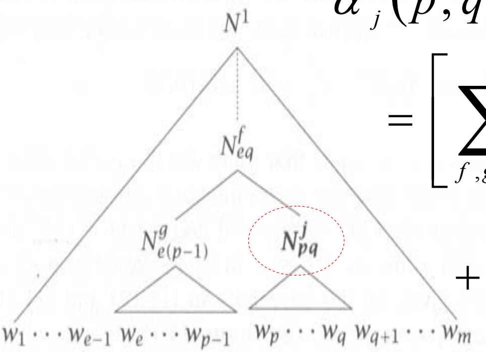
- The probabilities of **the base case**

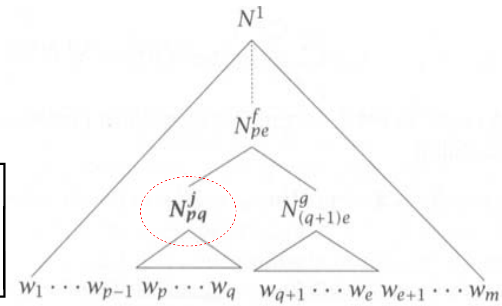
$$\alpha_1(1,m) = 1$$
$$\alpha_j(1,m) = 0 \text{ for } j \neq 1$$

- Find the probabilities $\alpha_j(p,q)$ **by induction**

Problem 1: The Probability of a Sentence

- Find the probabilities $\alpha_j(p,q)$ by induction
  - A **top-down** version of calculation





$$\alpha_j(p,q) = P\left(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} \middle| G\right)$$

$$= \left[\sum_{f,g\neq j}\sum_{e=q+1}^{m} P\left(w_{1(p-1)}, w_{(q+1)m}, N^f_{pe}, N^j_{pq}, N^g_{(q+1)e}\right)\right]$$

$$+ \left[\sum_{f,g}\sum_{e=1}^{p-1} P\left(w_{1(p-1)}, w_{(q+1)m}, N^f_{eq}, N^g_{e(p-1)}, N^j_{pq}\right)\right]$$

Chain rule & context-free & ancestor-free assumptions

$$= \left[\sum_{f,g\neq j}\sum_{e=q+1}^{m} P\left(w_{1(p-1)}, w_{(e+1)m}, N^f_{eq}\right) P\left(N^j_{pq}, N^g_{(q+1)e} \middle| N^f_{eq}\right) P\left(w_{(q+1)e} \middle| N^g_{(q+1)e}\right)\right]$$

$$+ \left[\sum_{f,g}\sum_{e=1}^{p-1} P\left(w_{1(e-1)}, w_{(q+1)m}, N^f_{eq}\right) P\left(N^g_{e(p-1)}, N^j_{pq} \middle| N^f_{eq}\right) P\left(w_{e(p-1)} \middle| N^g_{e(p-1)}\right)\right]$$

$$= \left[\sum_{f,g\neq j}\sum_{e=q+1}^{m} \alpha_f(p,e) P\left(N^f \rightarrow N^j N^g\right) \beta_g(q+1,e)\right]$$

$$+ \left[\sum_{f,g}\sum_{e=1}^{p-1} \alpha_f(e,q) P\left(N^f \rightarrow N^g N^j\right) \beta_g(e,p-1)\right]$$

Problem 1: The Probability of a Sentence

- Explanation

$$
\begin{aligned}
& P\left(w_{1(p-1)}, w_{(q+1)m}, N^f_{pe}, N^j_{pq}, N^g_{(q+1)e}\right) \\
&= P\left(w_{1(p-1)}, w_{(q+1)e}, w_{(e+1)m}, N^f_{pe}, N^j_{pq}, N^g_{(q+1)e}\right) \\
&= P\left(w_{1(p-1)}, w_{(e+1)m}, N^f_{pe}\right) P\left(w_{(q+1)e}, N^j_{pq}, N^g_{(q+1)e} \middle| \cancel{w_{1(p-1)}}, \cancel{w_{(e+1)m}}, N^f_{pe}\right) \\
&= P\left(w_{1(p-1)}, w_{(e+1)m}, N^f_{pe}\right) P\left(w_{(q+1)e}, N^j_{pq}, N^g_{(q+1)e} \middle| N^f_{pe}\right) \\
&= P\left(w_{1(p-1)}, w_{(e+1)m}, N^f_{pe}\right) P\left(N^j_{pq}, N^g_{(q+1)e} \middle| N^f_{pe}\right) P\left(w_{(q+1)e} \middle| \cancel{N^j_{pq}}, N^g_{(q+1)e}, \cancel{N^f_{pe}}\right) \\
&= P\left(w_{1(p-1)}, w_{(e+1)m}, N^f_{pe}\right) P\left(N^j_{pq}, N^g_{(q+1)e} \middle| N^f_{pe}\right) P\left(w_{(q+1)e} \middle| N^g_{(q+1)e}\right) \\
&= \alpha_f(p, e) P\left(N^f \rightarrow N^j N^g\right) \beta_g(q+1, e)
\end{aligned}
$$

Problem 1: The Probability of a Sentence

- The product of the inside and outside probabilities

$$
\begin{aligned}
\alpha_j(p,q)\beta_j(p,q) &= P\left(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m} \middle| G\right) P\left(w_{pq} \middle| N^j_{pq}, G\right) \\
&= P\left(N^j_{pq} \middle| G\right) P\left(w_{1(p-1)}, w_{(q+1)m} \middle| N^j_{pq}, G\right) P\left(w_{pq} \middle| N^j_{pq}, G\right) \\
&= P\left(N^j_{pq} \middle| G\right) P\left(w_{1(p-1)}, w_{pq}, w_{(q+1)m} \middle| N^j_{pq}, G\right) \\
&= P\left(w_{1m}, N^j_{pq} \middle| G\right)
\end{aligned}
$$

- The probability of a sentence having some constituent spanning from word *p* to *q*

$$
P\left(w_{1m}, N_{pq} \middle| G\right) = \sum_j \alpha_j(p,q)\beta_j(p,q)
$$

Problem 2: Find the Most Likely Parse

- A **Viterbi-style algorithm** adapted from the **inside algorithm** was used to find the most likely parse of a sentence
  - Similar to **the CYK algorithm** introduced previously
- Definition

$$\delta_i(p,q): \text{the highest inside probability parse of a subtree } N^i_{pq}$$

$$\psi_i(p,q): \text{store the backtrace information } (j, k, r) \text{ of a subtree } N^i_{pq}$$

Store the optimal setting

$$\left\{\begin{array}{l} N^i_{pq} \rightarrow N^{j_1}_{pr_1} N^{k_1}_{(r_1+1)q} \\ N^i_{pq} \rightarrow N^{j_2}_{pr_2} N^{k_2}_{(r_2+1)q} \\ N^i_{pq} \rightarrow N^{j_3}_{pr_3} N^{k_3}_{(r_3+1)q} \\ \quad\quad\ldots\ldots \end{array}\right.$$

Different combinations of constituents spanning different word ranges

Problem 2: Find the Most Likely Parse

1. Initialization

$$\delta_i(p,p) = P\left(N^i \rightarrow w_p\right)$$

2. Induction

$$\delta_i(p,q) = \max_{\substack{1\le j,k\le n \\ p\le r<q}} P\left(N^i \rightarrow N^j N^k\right)\delta_j(p,r)\delta_k(r+1,q)$$

$$\psi_i(p,q) = \arg\max_{\substack{1\le j,k\le n \\ p\le r<q}} P\left(N^i \rightarrow N^j N^k\right)\delta_j(p,r)\delta_k(r+1,q)$$

three elements stored (j,k,r)

3. Termination

The corresponding tree

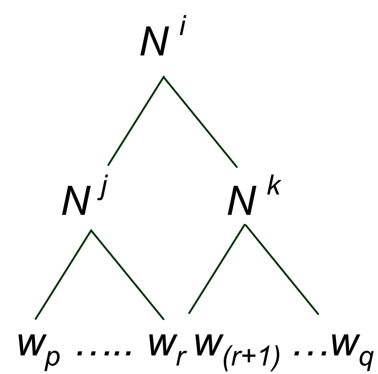
$$P\left(\hat{t}\right) = \delta_1(1,m) \Longrightarrow N^1_{1,m}$$

- Recursively construct the tree nodes

If $X_\chi = N^i_{pq}$, $\psi_i(p,q) = (j,k,r)$

$\text{left}\left(N^i_{pq}\right) = N^j_{pr}$

$\text{right}\left(N^i_{pq}\right) = N^k_{(r+1)q}$



Problem 3: Training a PCFG

- **If parsed training corpus are available**
  - Directly calculate the probabilities of rules via Maximum Likelihood Estimation (MLE)

$$\hat{P}\left(N^{j} \rightarrow \zeta\right)=\frac{C\left(N^{j} \rightarrow \zeta\right)}{\sum_{\gamma} C\left(N^{j} \rightarrow \gamma\right)}$$

The count of number of times a particular rule is used

The new probability of the rule

  - But, more commonly, a pared training corpus is not available (or a sentence may have many parses)
    - *A hidden data problem !*
    - We wish to determine probability function on rules, but can only directly see the probabilities of sentences

Problem 3: Training a PCFG

- **If parsed training corpus are not available**
  - An iterative algorithm is used to determine improving estimates of the probability of the corpus *W*

$$P(W|G_{i+1}) \geq P(W|G_i) \quad ?$$

  - Algorithm started with a certain grammar topology
    - The number of terminals and noterminals (determined)
    - The initial probability estimates for rules (randomly chosen)
  - According to this grammar
    - The probability of each parse of a training sentence are accumulated
    - The probabilities of each rule being used in each place are accumulated as an *expectation* of how often each rule are used

Problem 3: Training a PCFG

- **If parsed training corpus are not available**
  - Refine the probability estimates on rules in regarding to the expectations achieved previously
    - The likelihood of the training corpus given the grammar is increased $P(W|G_{i+1}) \geq P(W|G_i)$
  - Consider $\alpha_j(p,q)\beta_j(p,q) = P(w_{1m}, N^j_{pq}|G)$

$$= P\left(N^1 \overset{*}{\Rightarrow} w_{1m}, N^j \overset{*}{\Rightarrow} w_{pq} \middle| G\right)$$

$$= P\left(N^1 \overset{*}{\Rightarrow} w_{1m} \middle| G\right) P\left(N^j \overset{*}{\Rightarrow} w_{pq} \middle| N^1 \overset{*}{\Rightarrow} w_{1m}, G\right)$$

The probability of all possible parses

$\beta_1(1,m)$

- $P\left(N^1 \overset{*}{\Rightarrow} w_{1m} \middle| G\right)$ is calculated previously and is set as π

$$\Rightarrow P\left(N^j \overset{*}{\Rightarrow} w_{pq} \middle| N^1 \overset{*}{\Rightarrow} w_{1m}, G\right) = \frac{\alpha_j(p,q)\beta_j(p,q)}{\pi}$$

  - The estimate for how many times N^j is used

$$E(N^j \text{ is used in the derivation}) = \sum_{p=1}^{m}\sum_{q=p}^{m} \frac{\alpha_j(p,q)\beta_j(p,q)}{\pi}$$

Sum over all regions of words that the node could dominate in a sentence

Problem 3: Training a PCFG

- **If parsed training corpus are not available**
  - The estimate for how many times $N^j \rightarrow N^r N^s$ is used

$$E\left(N^j \rightarrow N^r N^s \text{ used}\right) =$$

$$\sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \sum_{d=p}^{q\text{-}1} \frac{\alpha_j(p,q)\, P\left(N^j \rightarrow N^r N^s\right) \beta_r(p,d) \beta_s(d+1,q)}{\pi}$$

  - The new probability for $N^j \rightarrow N^r N^s$ will be

$$\hat{P}\left(N^j \rightarrow N^r N^s\right) = \frac{\sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \sum_{d=p}^{q\text{-}1} \alpha_j(p,q)\, P\left(N^j \rightarrow N^r N^s\right) \beta_r(p,d) \beta_s(d+1,q)}{\sum_{p=1}^{m} \sum_{q=p}^{m} \alpha_j(p,q)\, \beta_j(p,q)}$$

The training formulas for a single sentence.

Problem 3: Training a PCFG

- **If parsed training corpus are not available**
  - The estimate for how many times $N^j \rightarrow w^k$ is used

$$E\left(N^j \rightarrow w^k \text{ used}\right)=\frac{\sum_{h=1}^{m} \alpha_j(h,h)\, P\left(N^j \rightarrow w_h, w_h = w^k\right)}{\pi}$$

$$=\frac{\sum_{h=1}^{m} \alpha_j(h,h) \boxed{P\left(w_h = w^k\right)} \beta_j(h,h)}{\pi}$$

Acts like a indicating function

  - The new probability for $N^j \rightarrow w^k$ will be

$$\hat{P}\left(N^j \rightarrow w^k\right)=\frac{\sum_{h=1}^{m} \alpha_j(h,h) P\left(w_h = w^k\right)\beta_j(h,h)}{\sum_{p=1}^{m}\sum_{q=p}^{m} \alpha_j(p,q)\beta_j(p,q)}$$

The training formulas for a single sentence.

Problem 3: Training a PCFG

- **If parsed training corpus are not available**
  - Assume the sentences in the corpus are independent
  - The likelihood of the corpus is just the product of the probabilities of sentences in it according to the grammar
  - Define common subterms for training sentences $W = (W_1, \ldots, W_\omega)$

$$f_i(p,q,j,r,s) = \frac{\sum_{d=p}^{q-1} \alpha_j(p,q)\, P(N^j \rightarrow N^r N^s)\, \beta_r(p,d)\, \beta_s(d+1,q)}{P\left(N^1 \stackrel{*}{\Rightarrow} W_i \middle| G\right)}$$

$$g_i(h,j,k) = \frac{\alpha_j(h,h)\, P(w_h = w^k)\, \beta_j(h,h)}{P\left(N^1 \stackrel{*}{\Rightarrow} W_i \middle| G\right)}$$

$$h_i(p,q,j) = \frac{\alpha_j(p,q)\, \beta_j(p,q)}{P\left(N^1 \stackrel{*}{\Rightarrow} W_i \middle| G\right)}$$

Problem 3: Training a PCFG

- **If parsed training corpus are not available**
  - The new probability for $N^j \rightarrow N^r N^s$ will be

$$\hat{P}\left(N^j \rightarrow N^r N^s\right) = \frac{\sum_{i=1}^{\omega} \sum_{p=1}^{m_i - 1} \sum_{q=p+1}^{m_i} f_i(p, q, j, r, s)}{\sum_{i=1}^{\omega} \sum_{p=1}^{m_i} \sum_{q=p}^{m_i} h_i(p, q, j)}$$

  - The new probability for $N^j \rightarrow w^k$ will be

$$\hat{P}\left(N^j \rightarrow w^k\right) = \frac{\sum_{i=1}^{\omega} \sum_{h=1}^{m_i} g_i(h, j, k)}{\sum_{i=1}^{\omega} \sum_{p=1}^{m_i} \sum_{q=p}^{m_i} h_i(p, q, j)}$$

The training formulas using all sentences.

Problems with the Inside-Outside Algorithm

- The whole training procedure is slow: $O(m^3 n^3)$ for each iteration
  - *m*: the length of the sentence
  - *n*: the number of nonterminals
- Local maxima are much more of a problem
- Satisfactory learning requires many more nonterminals than are theoretically needed to describe the language at hand
- No guarantee that the nonterminals learned will have any satisfactory resemblance to the kinds of non-terminals normally motivated in linguistic analysis

Problems with PCFGs

- The problems with PCFGs come from the fundamental independence assumptions
  - **Structural Independency**: the expansion of any one non-terminal is independent of any other non-terminal
    - Each rule is independent of each other rule
    - But the choice of how a node expands is dependent on the location of the node in the parse tree, e.g.,

NP $\rightarrow$ Pronoun or NP $\rightarrow$ Det Noun

NP is a subject in a sentence?
Talk about topic or old information

NP is an object in a sentence?
Introduce new referents

Switchboard: (for declarative sentences)
91% subjects are pronouns (9%: lexical nouns)
66% objects are lexical nouns (34% pronouns)

Problems with PCFGs

- The problems with PCFGs come from their fundamental independence (cont.)
  - **Lexical independency**: lack of sensitivity to words
    - **Lexical information** in PCFGs can only be represented via the probability of pre-terminal nodes (Verb, Noun, Det) to expanded lexically
    - But the **lexical information** plays an important role in selecting the correct parsing, e.g., **the ambiguous prepositional phrase attachment**

Moscow sent more than 100,000 soldiers into Afghanistan

$NP \rightarrow NP\ PP$ or $VP \rightarrow VP\ PP$

Problems with PCFGs

- **Lexical independency** (cont.)
  - **Attachment ambiguities**
    - Hindle and Rooth (13M words from the AP newswire 1991)
      - » 67% NP-attachment vs. 33% VP-attachment
    - Collins (WSJ and IBM computer manual, 1999)
      - » 52% NP-attachment
  - **Coordination ambiguities**
    - E.g., " *dogs in house and cats*"

A model keeping separate lexical dependency statistics for different verbs would be helpful for disambiguate these attachment problems !

Structural Dependency

- Examples

Pronouns, proper names, and definite NPs appear more commonly in subject position

NPs containing post-head modifiers and bare nouns occur more commonly in object position

Expansion	% as Subj	% as Obj
NP → PRP	13.7%	2.1%
NP → NNP	3.5%	0.9%
NP → DT NN	5.6%	4.6%
NP → NN	1.4%	2.8%
NP → NP SBAR	0.5%	2.6%
NP → NP PP	5.6%	14.1%

Table 12.3 Selected common expansions of NP as Subject vs. Object, ordered by log odds ratio. The data show that the rule used to expand NP is highly dependent on its parent node(s), which corresponds to either a subject or an object.

Expansion	% as 1st Obj	% as 2nd Obj
NP → NNS	7.5%	0.2%
NP → PRP	13.4%	0.9%
NP → NP PP	12.2%	14.4%
NP → DT NN	10.4%	13.3%
NP → NNP	4.5%	5.9%
NP → NN	3.9%	9.2%
NP → JJ NN	1.1%	10.4%
NP → NP SBAR	0.3%	5.1%

Table 12.4 Selected common expansions of NP as first and second object inside VP. The data are another example of the importance of structural context for nonterminal expansions.

Lexical Dependency

- Example

Local tree	Verb *come*	*take*	*think*	*want*
VP → V	9.5%	2.6%	4.6%	5.7%
VP → V NP	1.1%	32.1%	0.2%	13.9%
VP → V PP	34.5%	3.1%	7.1%	0.3%
VP → V SBAR	6.6%	0.3%	73.0%	0.2%
VP → V S	2.2%	1.3%	4.8%	70.8%
VP → V NP S	0.1%	5.7%	0.0%	0.3%
VP → V PRT NP	0.3%	5.8%	0.0%	0.0%
VP → V PRT PP	6.1%	1.5%	0.2%	0.0%

Table 12.2 Frequency of common subcategorization frames (local trees expanding VP) for selected verbs. The data show that the rule used to expand VP is highly dependent on the lexical identity of the verb. The counts ignore distinctions in verbal form tags. Phrase names are as in table 12.1, and tags are Penn Treebank tags (tables 4.5 and 4.6).

  – We should include more information about what the actual words in the sentence are when making decisions about the structure of the parse tree
    - Lexical dependencies between words

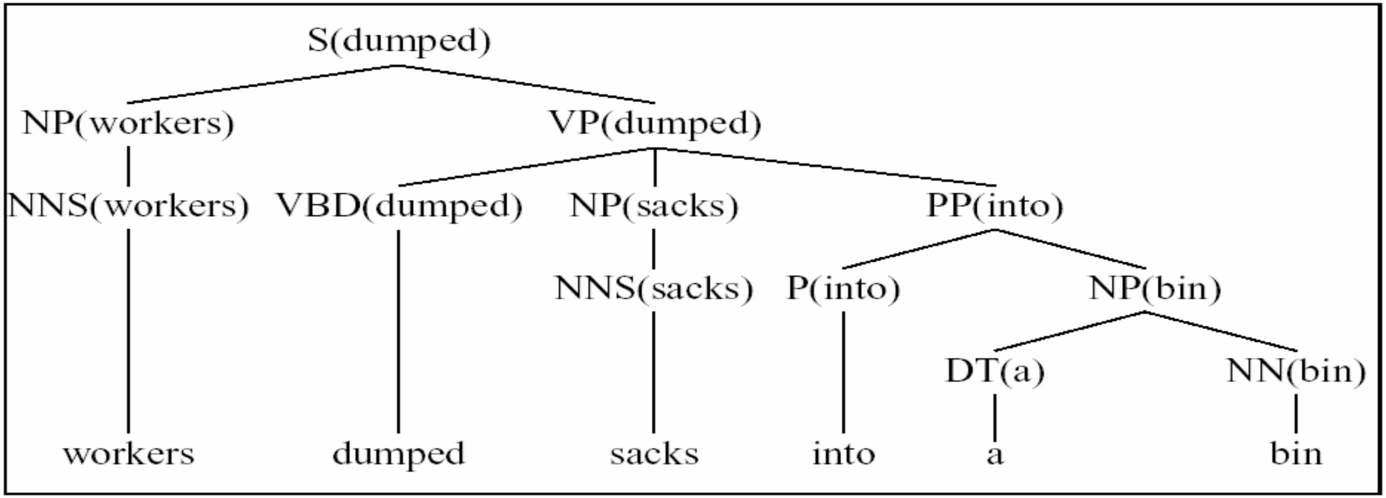
Problems with PCFGs

- Upshot
  - We should build a much better probabilistic parser than by taking into account lexical and structural context
- Challenge
  - How to find factors that give us a lot of extra discrimination while not defeating us with a multiplicity of parameters (or the sparse data problem)

Probabilistic Lexicalized CFGs

Black et al., 1992

- The syntactic constituents are associated with a **lexical head**
  - Each non-terminal is a parse tree is annotated with a single word which is its lexical head
  - Each rule is augmented to identify one right-hand-side constituent to be the head daughter



  - But how to choose is controversial !

Probabilistic Lexicalized CFGs

- How to select a head for a constituent ?
  - E.g., finding the head of a NP
    - Return the very last word if it is tagged POS (i.e., possessive)
    - Else to search from right to left for the first child that is an NN, NNP, etc.
    - Else to search from left to right for the first child that is an NP

$$NP \rightarrow NP\ PP$$

Probabilistic Lexicalized CFGs

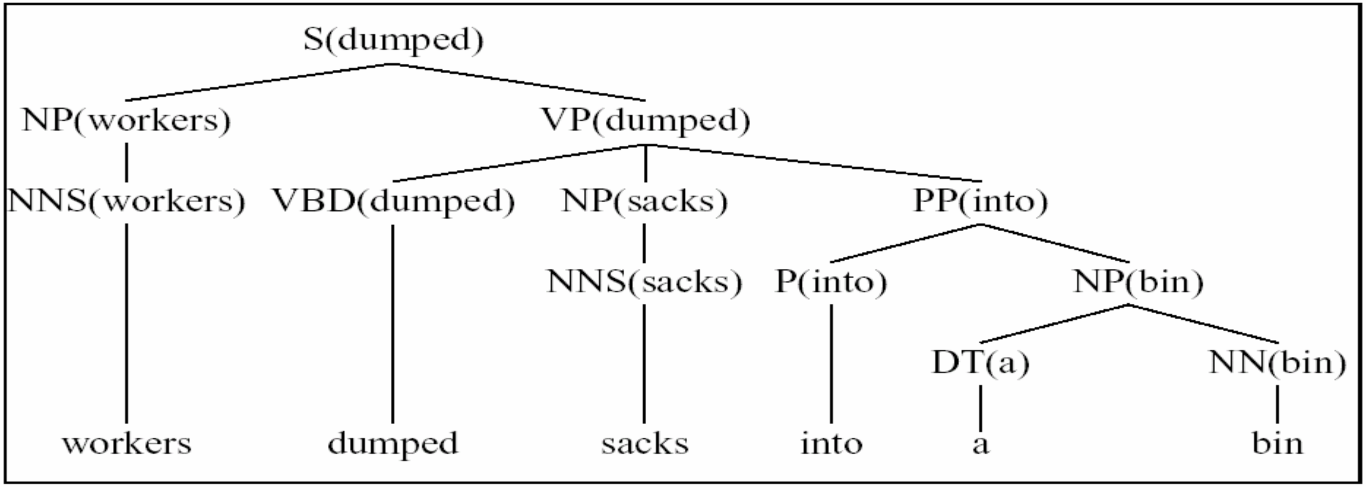
- A simple way to think of a lexicalized grammar
  - E.g., creating many copies of each rule, one copy for each possible head word for each constituent

VP (dumped) → VBD (dumped) NP (sacks) PP (into)	[$3\text{x}10^{-10}$]
VP (dumped) → VBD (dumped) NP (cats) PP (into)	[$8\text{x}10^{-11}$]
VP (dumped) → VBD (dumped) NP (hats) PP (into)	[$4\text{x}10^{-10}$]
VP (dumped) → VBD (dumped) NP (sacks) PP (above)	[$1\text{x}10^{-12}$]
……..	

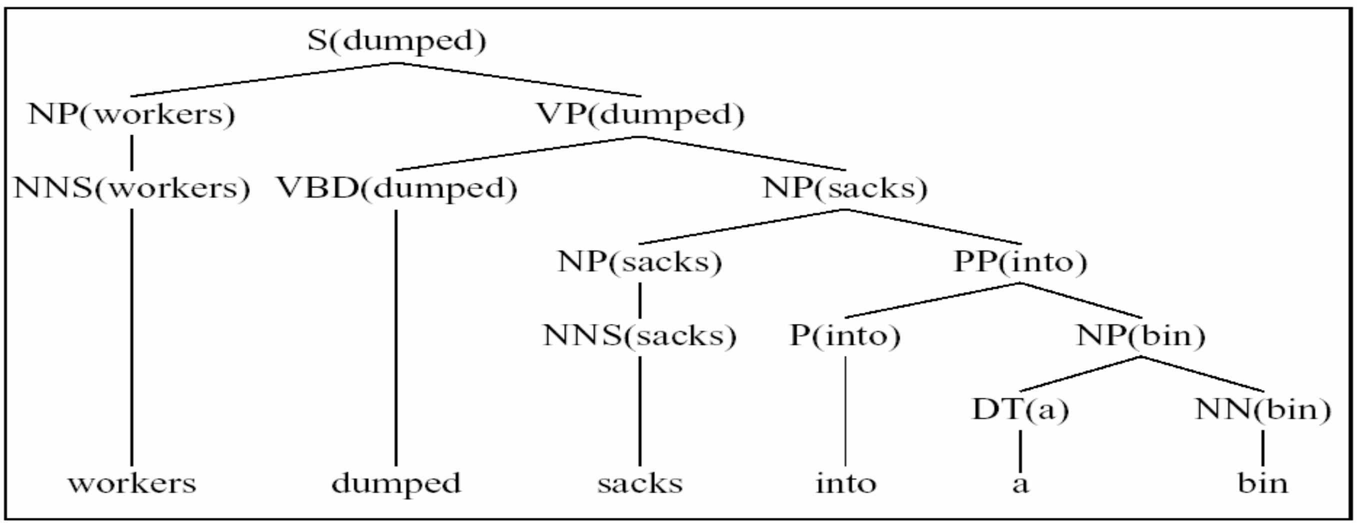
  - Problem
    - No corpus big enough to train such probabilities
      - Should make some simplifying independence assumptions in order to cluster some of the counts

Probabilistic Lexicalized CFGs

- Example



correct



incorrect

Probabilistic Lexicalized CFGs

- Take **Charniak's Parser** (1997) for example
  - Incorporate lexical dependency information by relating the heads of phrases to the heads of their constituents
  - Recall: the vanilla PCFG

$$P(r(n)|n)$$

n: the syntactic category of a parse-tree node

  - **Heard-rule probability** of the Probabilistic lexicalized CFG

$$P(r(n)|n, h(n))$$

h(*n*): the headword of a parse-tree node

- E.g.,

$VP \rightarrow VBD\ NP\ PP$

P(*r*|*VP, dumped*): the prob. of the rule

P(*r*|*VP, slept*): the prob. of the rule

Probabilistic Lexicalized CFGs

The prior probability of the head words

- **Further decide the probability of a head**
  - Null assumption: all head are equally likely
    - The probability that the head of a node would be *sacks* would be the same as the probability the head would be *racks*
    - Doesn't seem very useful
  - **Condition the probability of the head *h* of node *n* on two factors**
    - Syntactic category of the node *n*
    - The head of the node's mother

$$P\left(h(n) = word_i \mid n, \boxed{h(m(n))}\right)$$

$$\begin{array}{c} \text{X(dumped)} \\ \downarrow \\ \text{NP(?sacks?)} \end{array}$$

P(*head*(*n*)=sacks|*n*=*VP*, *h*(*m*(*n*))=*dumped*)

Probabilistic Lexicalized CFGs

– The probability of a parse *T* of a sentence *S*

$$P(T,S)=\prod_{n\in T} P\big(r(n)\,\big|\,n,h(n)\big)P\big(h(n)\,\big|\,n,h(m(n))\big)$$

head-rule probability

head-head probability

Counting from Brown corpus

$$P(VP \to VBD\ \ NP\ \ PP \mid VP, dumped)$$
$$=\frac{C(VP(dumped)\to VBD\ \ NP\ \ PP)}{\sum_{\beta} C(VP(dumped)\to \beta)}=\frac{6}{9}=0.67$$

$$P(into \mid PP, dumped)$$
$$=\frac{C(X(dumped)\to \ldots PP(into)\ldots)}{\sum C(X(dumped)\to \ldots PP\ldots)}=\frac{2}{9}=0.22$$

$$P(VP \to VBD\ \ NP \mid VP, dumped)$$
$$=\frac{C(VP(dumped)\to VBD\ \ NP)}{\sum_{\beta} C(VP(dumped)\to \beta)}=\frac{0}{9}=0$$

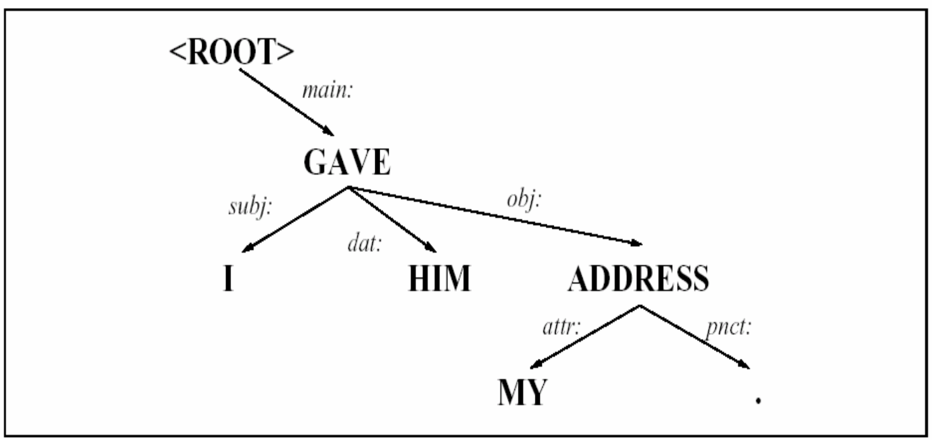
$$P(into \mid PP, sacks)$$
$$=\frac{C(X(sacks)\to \ldots PP(into)\ldots)}{\sum C(X(sacks)\to \ldots PP\ldots)}=\frac{0}{0}\Rightarrow 0$$

Probabilistic Lexicalized CFGs

- The original version of **Charniak's parser** adds additional conditional factors
  - The rule-expansion probability depends on the node's grandparent (trigram or second-order)
  - Use various backoff and smoothing algorithm

Dependency Grammars

- The grammar formulation is based purely on the lexical dependency information
  - The syntactic structure of a sentence is described purely in terms of words and binary semantic or syntactic relations between words



Dependency	Description
subj	syntactic subject
obj	direct object (incl. sentential complements)
dat	indirect object
pcomp	complement of a preposition
comp	predicate nominals (complements of copulas)
tmp	temporal adverbials
loc	location adverbials
attr	premodifying (attributive) nominals (genitives, etc.)
mod	nominal postmodifiers (prepositional phrases, etc.)

Dependency Grammars

- One of the main advantages of dependency grammars is their ability to handle languages with relatively **free word order**
  – Abstract away from word-order variation, representing only information that is necessary for the parse

Categorial Grammars

- The combinatory categorial grammar has two components
  - The categorial lexicon
    - Associate each word with a syntactic and semantic category
    - Two categories
      - Augments: Ns
      - Factors : verbs, determiners
  - The combination rules
    - Allow functions and arguments to be combined, e.g.,
      - X/Y: something combines with a Y on its right to produce X
      - X\Y: something combines with a Y on its left to produce X

Categorial Grammars

- Examples
  - Determiners receive the category NP/N
  - Transitive verbs might have the category VP/NP
  - Ditransitive verbs might have the category (VP/NP)/NP

Harry eats apples

NP V NP

VP/NP

S\NP

Evaluating Parsers

- Labeled recall

$$\frac{\text{\# of correct constituents in candidate parse of a sentence } s}{\text{\# of correct constituents in treebank parse of a sentence } s}$$

- Labeled precision

$$\frac{\text{\# of correct constituents in candidate parse of a sentence } s}{\text{\# of total constituents in candidate parse of a sentence } s}$$

- Cross-brackets
  - Number of total brackets
  - E.g., a cross-bracket

    ((A B) C) and (A (B C))

The correct constituent must have the same starting time, ending time, and non-terminal symbol as the "gold standard" of treebank.

Evaluating Parsers

- Examples
  - Using a portion of the Wall Street Journal as the test set, parsers such as Charniak (1997) and Collins (1999) achieve just
    - Under 90% recall and under 90% precision
    - About 1% cross-bracketed constituents per sentence