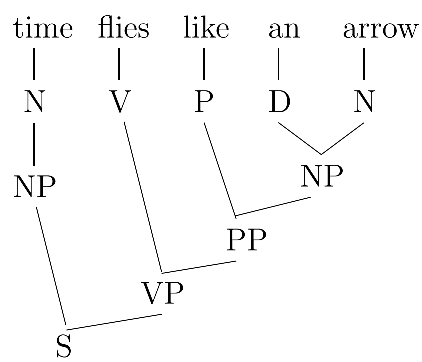
16 Syntactic disambiguation

Given word sequence,
extract hidden syntactic structure:

- objects
- relations
  (arguments of relation)
- modifiers
  (who modifies who)

16.1 Recover "phrase structure" of sentence

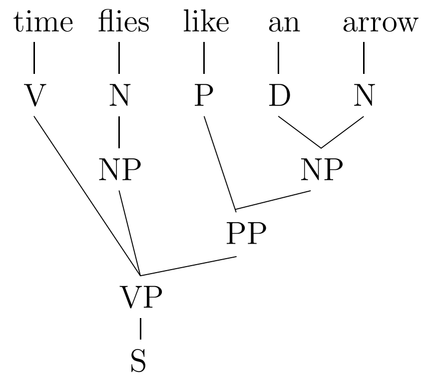
E.g.



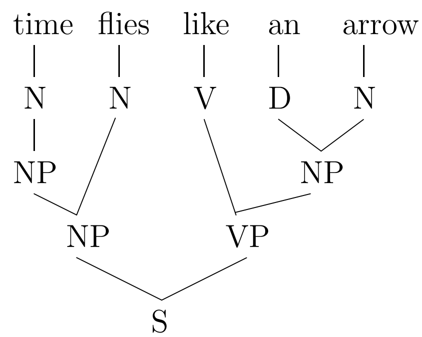
(standard "metaphorical" interpretation)

16.2 Syntactic ambiguity

"Command" interpretation



"Strange species of flies" interpretation



16.3 Capture phrase structure with a CFG

Context Free Grammar (CFG) consists of:

- Terminal symbols (words) $A = \{w_1, w_2, ...\}$
- Non-terminal symbols $N_1, N_2, ...$
- Special start symbol S
- Context free rules $N \rightarrow \alpha$
  - N a single non-terminal
  - α a finite string of terminals/non-terminals

E.g.

$$
\begin{array}{rcl@{\qquad}rcl}
S & \rightarrow & NP\ VP & V & \rightarrow & \text{flies} \\
S & \rightarrow & VP & V & \rightarrow & \text{like} \\
NP & \rightarrow & N & V & \rightarrow & \text{time} \\
NP & \rightarrow & D\ N & N & \rightarrow & \text{flies} \\
NP & \rightarrow & NP\ N & N & \rightarrow & \text{arrow} \\
VP & \rightarrow & V\ NP & N & \rightarrow & \text{time} \\
VP & \rightarrow & V\ PP & P & \rightarrow & \text{like} \\
VP & \rightarrow & V\ NP\ PP & D & \rightarrow & \text{an} \\
PP & \rightarrow & P\ NP & & &
\end{array}
$$

Sequences a grammar can produce are *legal*
Sequences a grammar cannot produce are *illegal*

In natural language

- Legal sequences can have many different parses (derivations)
- Selecting a parse is *important*
  → gives argument structure

Will select "right" parse with probability models

Build a joint distribution $P(sentence, parse)$

$$
\begin{aligned}
interp &= \arg\max_{parse} P(parse|sentence) \\
&= \arg\max_{parse} P(parse, sentence)
\end{aligned}
$$

16.4 Probabilistic Context Free Grammar

Add probabilities to a context free grammar

- For each non-terminal N_i

  Assign probability distribution over its rules

$$\begin{array}{ll} N_i \rightarrow \alpha_1 & p_1 \\ N_i \rightarrow \alpha_2 & p_2 \\ \vdots & \\ N_i \rightarrow \alpha_k & p_k \end{array}$$

  Where $\sum_j p_j = 1$

E.g.

S	→	NP VP	.6	V	→	flies	.5
S	→	VP	.4	V	→	like	.3
NP	→	N	.5	V	→	time	.2
NP	→	D N	.3	N	→	flies	.5
NP	→	NP N	.2	N	→	arrow	.3
VP	→	V NP	.5	N	→	time	.2
VP	→	V PP	.3	P	→	like	1
VP	→	V NP PP	.2	D	→	an	1
PP	→	P NP	1				

16.5 Generate

Sample random *tree*, *sentence*) configurations by

- Starting with S
- Expand non-terminals independently by selecting rules according to probabilities

This assumes subtrees are conditionally independent given their root
Generates random trees

$$\text{leaves} \quad = \quad \text{word sequence}$$

16.6 Evaluation

Calculate probability of a complete (*tree*, *sentence*) configuration by taking products of the individual probabilities

E.g. Let *sentence1* = "time flies like an arrow" and let *tree1*, *tree2* and *tree3* be the three different parse trees shown before (respectively). Then we have

$$\begin{aligned}
\mathrm{P}(\mathit{tree1}, \mathit{sentence1}) &= .6\ .5\ .2\ .3\ .5\ 1\ 1\ .3\ 1\ .3 \ = \ .00081 \\
\mathrm{P}(\mathit{tree2}, \mathit{sentence1}) &= .4\ .2\ .2\ .5\ .5\ 1\ 1\ .3\ 1\ .3 \ = \ .00036 \\
\mathrm{P}(\mathit{tree3}, \mathit{sentence1}) &= .6\ .2\ .5\ .2\ .5\ .5\ .3\ .3\ 1\ .3 \ = \ .000081
\end{aligned}$$

16.7 Inference

Marginalization

$$P(\mathit{sentence}) = \sum_{\mathit{trees}} \mathrm{P}(\mathit{sentence}, \mathit{tree})$$

Conditioning

$$P(\mathit{tree}|\mathit{sentence}) = \frac{\mathrm{P}(\mathit{sentence}, \mathit{tree})}{\mathrm{P}(\mathit{sentence})}$$

Completion

$$\begin{aligned}
\mathit{interpretation} &= \arg\max_{\mathit{tree}} \mathrm{P}(\mathit{tree}|\mathit{sentence}) \\
&= \arg\max_{\mathit{tree}} \mathrm{P}(\mathit{tree}, \mathit{sentence})
\end{aligned}$$

16.8 Polynomial time algorithms for PCFGs

First, we will assume CFG is in *Chomsky Normal Form (CNF)*.
That is, rules are restricted to be of form:

- $S \rightarrow N_i$
- $N_i \rightarrow N_j N_k$
- $N_i \rightarrow w$

Note For any PCFG there is an equivalent PCFG in Chomsky normal form
E.g.

- Eliminate unit chains $N_1 \to N_2, N_2 \to N_3, ..., N_k \to w$,
  by replacing each chain with a single rule $N_1 \to w$.

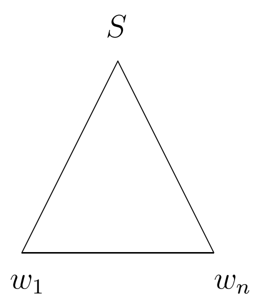
  Probability of new rule = product of probabilities in original chain

- Eliminate non-binary rules $N_1 \to N_2 N_3 ... N_k$,
  by replacing this with a set of binary rules on new non-terminals
  $N_1 \to N_2 A_2,\ A_2 \to N_3 A_3,\ ...\ A_k \to N_k$.

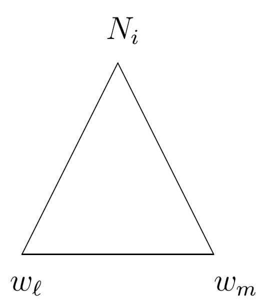
  Where probability of $N_1 \to N_2 A_2$ = probability of original rule,
  remaining probabilities = 1

Efficient marginalization

Compute $\mathrm{P}(w_1 ... w_n | S)$

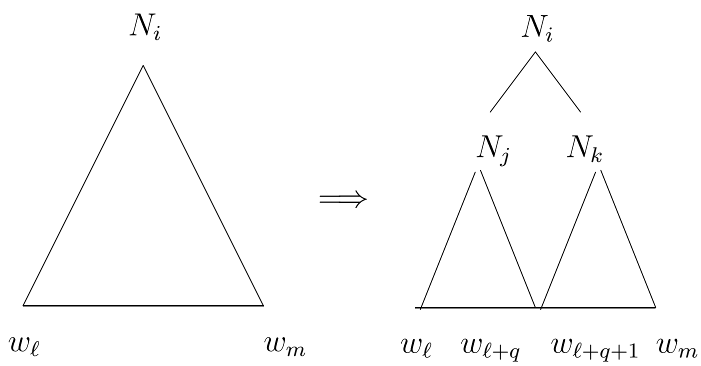


Consider recursive divide and conquer approach:



$$\mathrm{P}(w_\ell ... w_m | N_i)$$

$$= \begin{cases} \mathrm{P}(N_i \to w_\ell) & \text{if } m = \ell \\ \displaystyle\sum_{N_j} \sum_{N_k} \mathrm{P}(N_i \to N_j N_k) \sum_{q=0}^{m-\ell-1} \mathrm{P}(w_\ell ... w_{\ell+q} | N_j)\, \mathrm{P}(w_{\ell+q+1} ... w_m | N_k) & \\ & \text{otherwise} \end{cases}$$



- Note that the rightmost product encodes the assumption that the subtrees generated below N_j and N_k are independent once N_j and N_k are chosen.
- Unfortunately the computation time of this recursive procedure is exponential (because subtree computations can be repeated)

Efficient bottom-up dynamic programming

Compute all	Time
$\mathrm{P}(w_\ell \| N_i)$	$n \times N$
$\mathrm{P}(w_\ell w_{\ell+1} \| N_i)$	$(n-1) \times 1 \times N^3$
$\mathrm{P}(w_\ell w_{\ell+1} w_{\ell+2} \| N_i)$	$(n-2) \times 2 \times N^3$
$\vdots$	
$\mathrm{P}(w_\ell ... w_{\ell+j} \| N_i)$	$(n-j) \times j \times N^3$

Total time $= N^3 \sum_{j=1}^{n} (n-j)j \ = \ O(N^3 n^3)$

Note $N^3 \geq G$ where G is the number of non-terminal rules in the grammar, so the running time is actually more like $O(Gn^3)$

16.9 Completion

$$tree^* = \arg\max_{tree} \mathrm{P}(w_1 ... w_n, tree)$$

Same algorithm as above!

Just replace $\displaystyle\sum_{N_j} \sum_{N_k} \sum_{q=0}^{m-\ell-1}$

with $\displaystyle\max_{N_j} \max_{N_k} \max_{q=0}^{m-\ell-1}$

Readings

Russell and Norvig: Chapter 23
Dean, Allen, Aloimonos: Sections 10.2-10.5