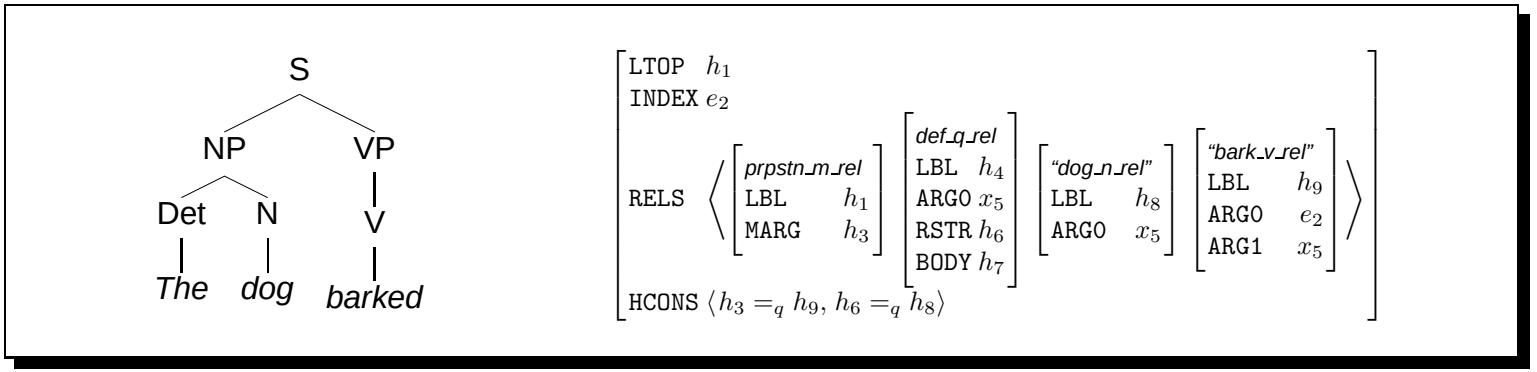


Computational Linguistics (INF2820 — PCFGs)

P(S $\longrightarrow$ NP VP) = 1.0; P(NP $\longrightarrow$ Det N) = 0.6

Stephan Oepen

Universitetet i Oslo & CSLI Stanford

oe@ifi.uio.no

Independence Assumptions: String Probabilities

Its water was so clean that the sands on its bed could be clearly seen.

Assigning Probabilities to Parse Trees

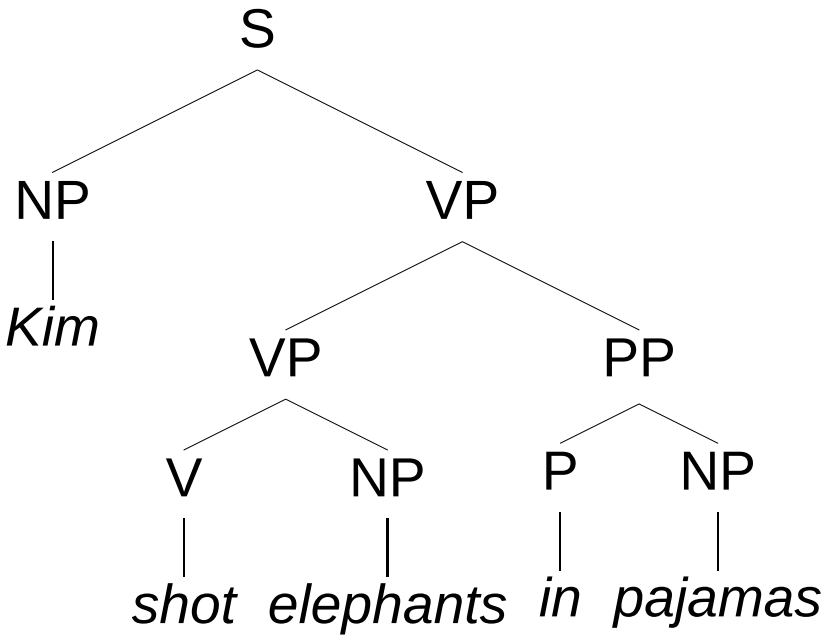
Treebanks

- For probability estimation, we need training data: 'correct' trees;
- a *treebank* pairs a corpus of sentences with *gold-standard* trees;
- *annotation* adds linguistic structure (e.g. trees) to raw corpus text;
- Penn Treebank: one million words of WSJ, manually annotated.

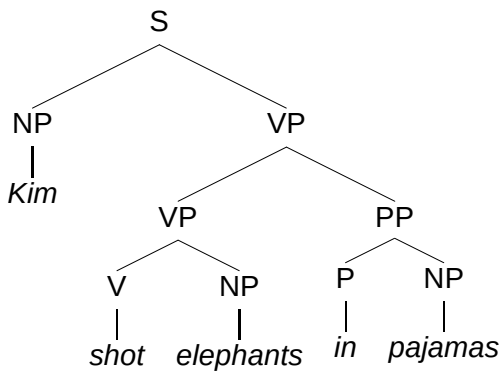
Probability Model

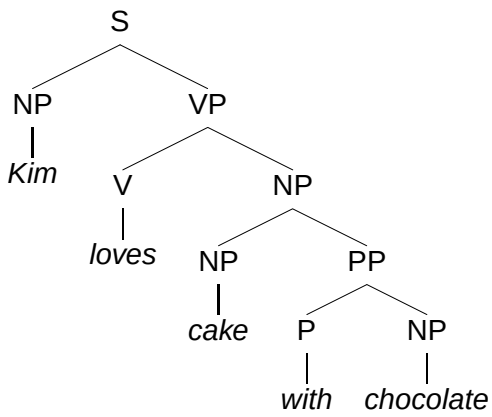
- A tree results from a sequence of rule applications (a derivation);
- joint probability: estimate *rule probabilities* and multiply (chain rule);
- assume that probability of each rule is independent from context.

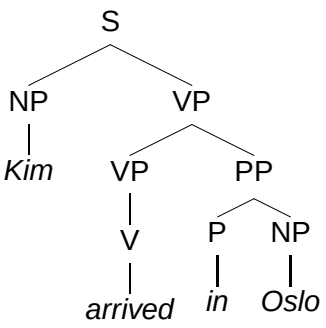
A Practical Example



A (Simplified) PCFG Estimation Example







P(RHS|LHS)	CFG Rule
	S → NP VP
	VP → VP PP
	VP → V NP
	PP → P NP
	NP → NP PP
	VP → V

- Estimate rule probability from observed distribution;

→ conditional probabilities:

$$P(\text{RHS}|\text{LHS}) = \frac{C(\text{LHS}, \text{RHS})}{C(\text{LHS})}$$

Formally: Probabilistic Context-Free Grammars

- Formally, a *context-free grammar* (CFG) is a quadruple: $\langle C, \Sigma, P, S \rangle$

...

- P is a set of category rewrite rules (aka *productions*), each with a conditional probability P(RHS|LHS), e.g.

...

NP $\rightarrow$ Kim [0.6]
NP $\rightarrow$ snow [0.4]

...

- for each rule '$\alpha \rightarrow \beta_1, \beta_2, ..., \beta_n$' $\in P$: $\alpha \in C$ and $\beta_i \in C \cup \Sigma$; $1 \leq i \leq n$;

...

- for each $\alpha \in C$, the probabilities of all rules R '$\alpha \rightarrow$ ...' must sum to 1.

Reminder: The CKY Algorithm

for ($0 \leq i < |\mathit{input}|$) do
  $\mathit{chart}_{[i,i+1]} \leftarrow \{\alpha \mid \alpha \rightarrow \mathit{input}_i \in P\}$;
for ($1 \leq l < |\mathit{input}|$) do
  for ($0 \leq i < |\mathit{input}| - l$) do
    for ($1 \leq j \leq l$) do
      if ($\alpha \rightarrow \beta_1\,\beta_2 \in P \wedge \beta_1 \in \mathit{chart}_{[i,i+j]} \wedge \beta_2 \in \mathit{chart}_{[i+j,i+l+1]}$) then
        $\mathit{chart}_{[i,i+l+1]} \leftarrow \mathit{chart}_{[i,i+l+1]} \cup \{\alpha\}$;

Kim adored snow in Oslo

$$
\begin{aligned}
[0,2] &\leftarrow [0,1] + [1,2] \\
&\ldots \\
[0,5] &\leftarrow [0,1] + [1,5] \\
[0,5] &\leftarrow [0,2] + [2,5] \\
[0,5] &\leftarrow [0,3] + [3,5] \\
[0,5] &\leftarrow [0,4] + [4,5]
\end{aligned}
$$

	1	2	3	4	5
0	NP		S		S
1		V	VP		VP
2			NP		NP
3				P	PP
4					NP

An Extension: Finding the Most Probable Tree

for ($1 \leq l < |\mathit{input}|$) do
  for ($0 \leq i < |\mathit{input}| - l$) do
    for ($1 \leq j \leq l$) do
      if ($\alpha \rightarrow \beta_1\,\beta_2 \in P \wedge \beta_1 \in \mathit{chart}_{[i,i+j]} \wedge \beta_2 \in \mathit{chart}_{[i+j,i+l+1]}$) then

- For each cell and each category, keep track of *most probably* tree;
- extra information: probabilities and *backpointers* $\rightarrow$ trivial read-out.

Backpointers: Recording the Derivation History

	0	1	2	3
0	2: S → • NP VP 1: NP → • NP PP 0: NP → • kim	10: S → 8 • VP 9: NP → 8 • PP 8: NP → kim •		17: S → 8 15 •
1		5: VP → • VP PP 4: VP → • V NP 3: V → • adored	12: VP → 11 • NP 11: V → adored •	16: VP → 15 • PP 15: VP → 11 13 •
2			7: NP → • NP PP 6: NP → • snow	14: NP → 13 • PP 13: NP → snow •
3				

- Use edges to record derivation trees: backpointers to daughters;
- a single edge can represent multiple derivations: backpointer sets.