Probabilistic context-free grammars

Steve Renals
s.renals@ed.ac.uk

ICL — 17 November 2005

Context-free Grammars (CFGs)

Recall...

A CFG is a 4-tuple $< \mathbf{N}, \Sigma, \mathbf{P}, \mathbf{S} >$ where:

- **N** is the set of non-terminal symbols (eg S, NP, VP, ...)
- Σ is the set of terminal symbols (eg *the*, *cat*, *sat*, ...)
- **P** is the set of productions $A \rightarrow \alpha$, $A \in N$ and $\alpha \in (N \cup \Sigma)^*$ (eg S $\rightarrow$ VP NP)
- **S** is the start symbol S

Parsing

- Top-down Predict what you expect to see (eg Earley algorithm)
- Bottom-up Start with the words, then incrementally build up parse trees
  - CYK (Cocke-Younger-Kasami) algorithm
  - Well matched to probabilistic grammars—assign probabilities to constituents as they are completed and placed in the table
  - Dynamic programming approach—store and reuse intermediate results (ie probabilities of sub-trees)
  - Resolve ambiguities by taking the most probable sub-tree

Chomsky Normal Form (CNF)

- If a CFG is in Chomsky Normal Form, productions are constrained to be of two forms only:
  - **Expand to 2 non-terminals**, eg: A $\rightarrow$ B C
    with A, B, C all non-terminals
  - **Expand to 1 terminal**, eg: A $\rightarrow$ a
    where A is a non-terminal and a is a terminal
- Any CFG can be translated to a (weakly) equivalent CFG in CNF
- The CYK algorithm requires a grammar in CNF

CYK Example

Alice called Bob from Cardiff

Context-Free Grammar:

S $\rightarrow$ NP VP	NP $\rightarrow$ Alice
NP $\rightarrow$ NP PP	NP $\rightarrow$ Bob
VP $\rightarrow$ V NP	NP $\rightarrow$ Cardiff
VP $\rightarrow$ VP PP	V $\rightarrow$ called
PP $\rightarrow$ P NP	P $\rightarrow$ from

CYK Parsing

				Cardiff
			from	
		Bob		
	called			
Alice				

CYK Parse Chart

CYK Parsing

				Cardiff
			from	
		Bob		
NP	called			
Alice				

Base case

CYK Parsing

				Cardiff
			from	
	V	Bob		
NP	called			
Alice				

Base case

CYK Parsing

				Cardiff
		NP	from	
	V	**Bob**		
NP	called			
Alice				

Base case

CYK Parsing

			P	Cardiff
		NP	**from**	
	V	Bob		
NP	called			
Alice				

Base case

CYK Parsing

				NP
			P	**Cardiff**
		NP	from	
	V	Bob		
NP	called			
Alice				

Base case

CYK Parsing

				NP
			P	Cardiff
		NP	from	
X	V	Bob		
NP	called			
Alice				

Recursion

CYK Parsing

				NP
			P	Cardiff
	VP	**NP**	from	
X	**V**	**Bob**		
NP	**called**			
Alice				

Recursion

CYK Parsing

				NP
		X	P	Cardiff
	VP	NP	from	
X	V	Bob		
NP	called			
Alice				

Recursion

CYK Parsing

			PP	**NP**
		X	**P**	**Cardiff**
	VP	NP	**from**	
X	V	Bob		
NP	called			
Alice				

Recursion

CYK Parsing

			PP	NP
		X	P	Cardiff
S	**VP**	**NP**	from	
X	**V**	**Bob**		
NP	**called**			
Alice				

Recursion

CYK Parsing

			PP	NP
	X	X	P	Cardiff
S	VP	NP	from	
X	V	Bob		
NP	called			
Alice				

Recursion

CYK Parsing

		NP	**PP**	**NP**
	X	X	**P**	**Cardiff**
S	VP	**NP**	**from**	
X	V	**Bob**		
NP	called			
Alice				

Recursion

CYK Parsing

		NP	PP	NP
X	X	X	P	Cardiff
S	VP	NP	from	
X	V	Bob		
NP	called			
Alice				

Recursion

CYK Parsing

	VP_1	NP	**PP**	**NP**
X	X	X	**P**	**Cardiff**
S	**VP**	**NP**	**from**	
X	**V**	**Bob**		
NP	**called**			
Alice				

Recursion

CYK Parsing

	$\mathbf{VP}_2$	**NP**	**PP**	**NP**
X	X	X	**P**	**Cardiff**
S	VP	**NP**	**from**	
X	**V**	**Bob**		
NP	**called**			
Alice				

Recursion

CYK Parsing

	VP_1/VP_2	**NP**	**PP**	NP
X	X	X	P	**Cardiff**
S	**VP**	NP	**from**	
X	**V**	**Bob**		
NP	**called**			
Alice				

Recursion

CYK Parsing

S	**VP**	NP	**PP**	**NP**
X	X	X	**P**	**Cardiff**
S	**VP**	**NP**	**from**	
X	**V**	**Bob**		
NP	**called**			
Alice				

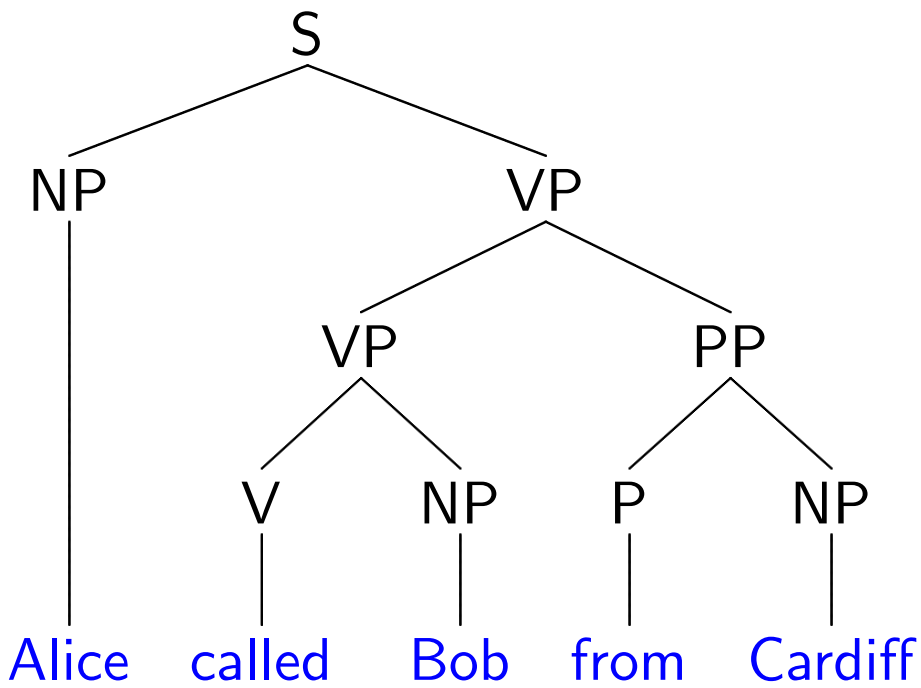
First Parse

CYK Parsing

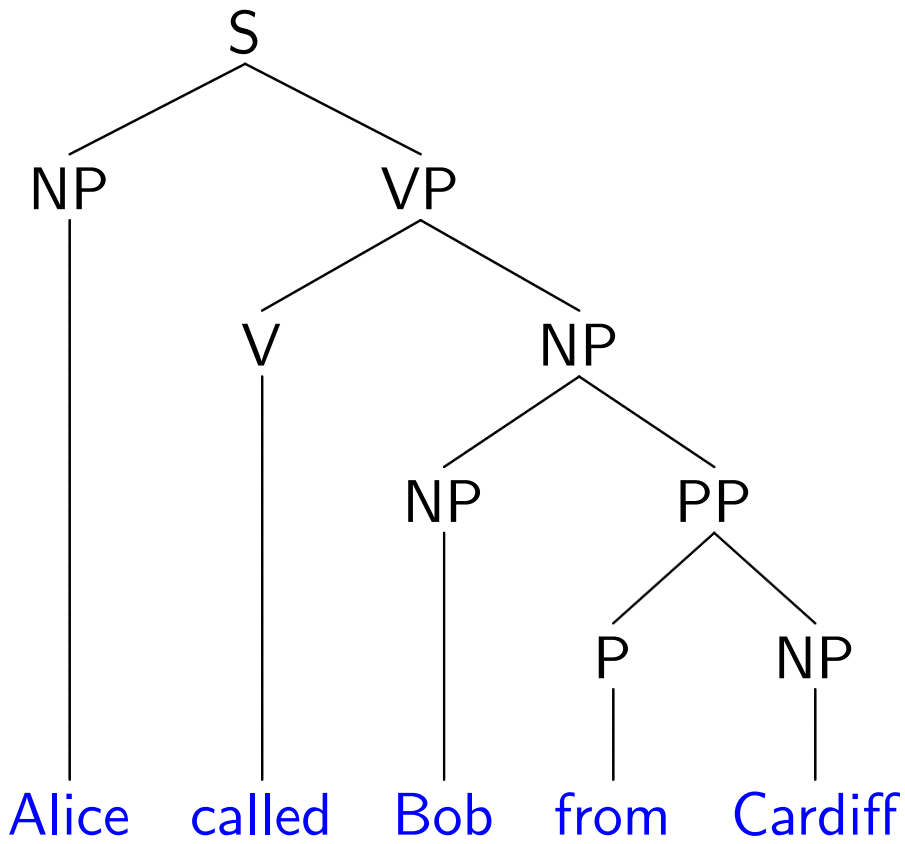
S	**VP**	**NP**	**PP**	**NP**
X	X	X	**P**	**Cardiff**
S	VP	**NP**	**from**	
X	**V**	**Bob**		
NP	**called**			
Alice				

Second Parse

First tree



Second tree



Probabilistic context-free grammars (PCFGs)

- A probabilistic context-free grammar augments each rule in a CFG with a conditional probability p
  $A \rightarrow \alpha \quad (p)$
- This probability is the probability that given non-terminal A it will be expanded to the sequence α; written as $P(A \rightarrow \alpha|A)$ or $P(A \rightarrow \alpha)$
- Probability of a parse tree is the product of the rule probabilities used to construct the parse

Probabilistic parsing

- Consider the rule:

$$\mathrm{S} \rightarrow \mathrm{NP\ VP}$$

  Then the probability of S is the product of the rule probability and the probability of each subtree:

$$P(S) = P(\mathrm{S} \rightarrow \mathrm{NP\ VP}) \cdot P(\mathrm{NP}) \cdot P(\mathrm{VP})$$

- We are doing bottom-up parsing... so we already know the subtree probabilities $P(\mathrm{NP})$ and $P(\mathrm{VP})$

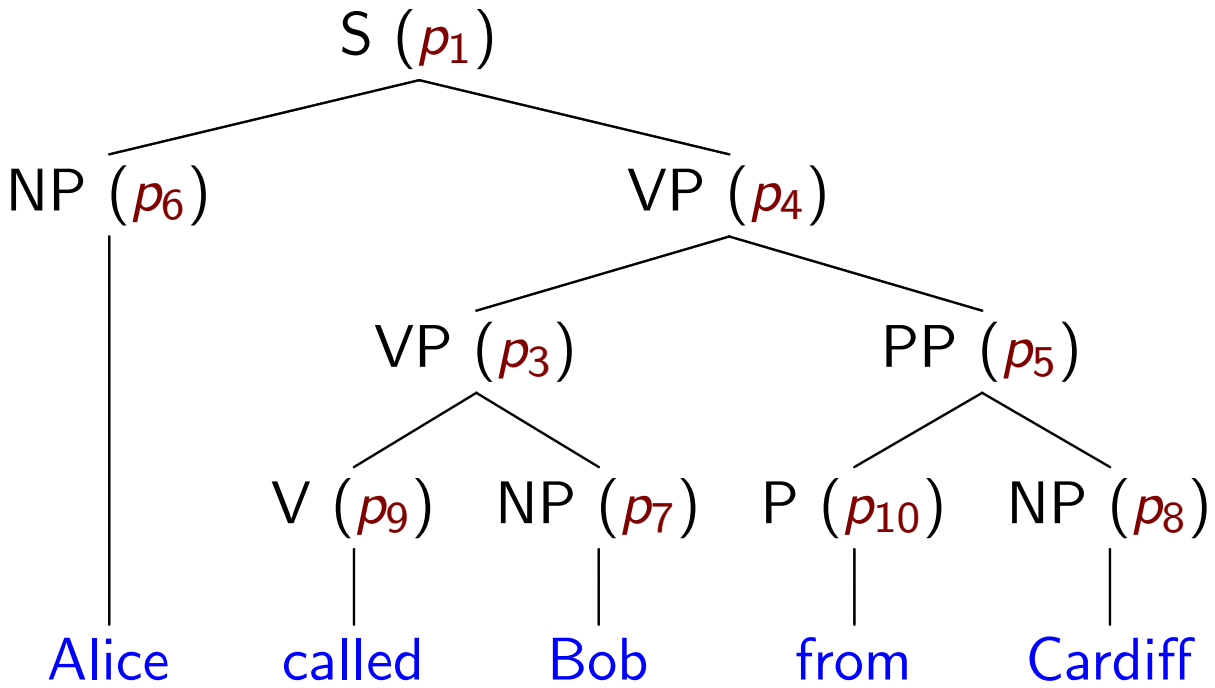
Probabilistic CYK Example

Alice called Bob from Cardiff

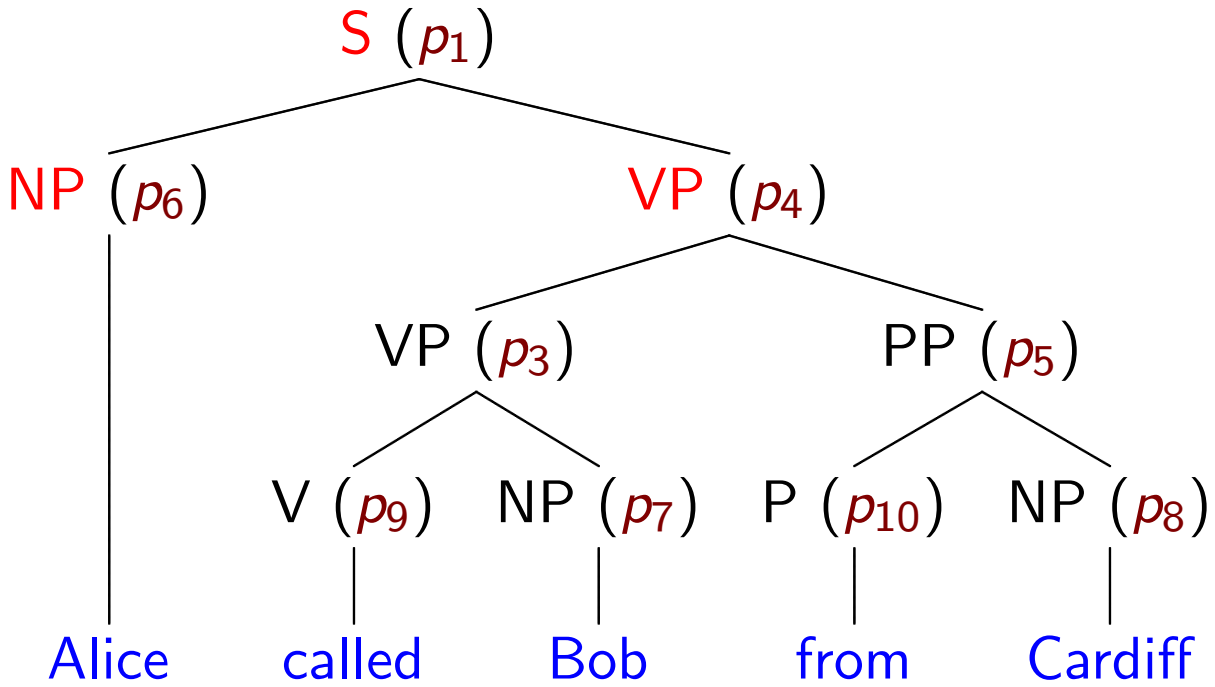
Context-Free Grammar:

$$
\begin{array}{ll}
\mathrm{S} \rightarrow \mathrm{NP\ VP} & (p_1) \\
\mathrm{NP} \rightarrow \mathrm{NP\ PP} & (p_2) \\
\mathrm{VP} \rightarrow \mathrm{V\ NP} & (p_3) \\
\mathrm{VP} \rightarrow \mathrm{VP\ PP} & (p_4) \\
\mathrm{PP} \rightarrow \mathrm{P\ NP} & (p_5)
\end{array}
\qquad
\begin{array}{ll}
\mathrm{NP} \rightarrow \mathrm{Alice} & (p_6) \\
\mathrm{NP} \rightarrow \mathrm{Bob} & (p_7) \\
\mathrm{NP} \rightarrow \mathrm{Cardiff} & (p_8) \\
\mathrm{V} \rightarrow \mathrm{called} & (p_9) \\
\mathrm{P} \rightarrow \mathrm{from} & (p_{10})
\end{array}
$$

First tree (T1)

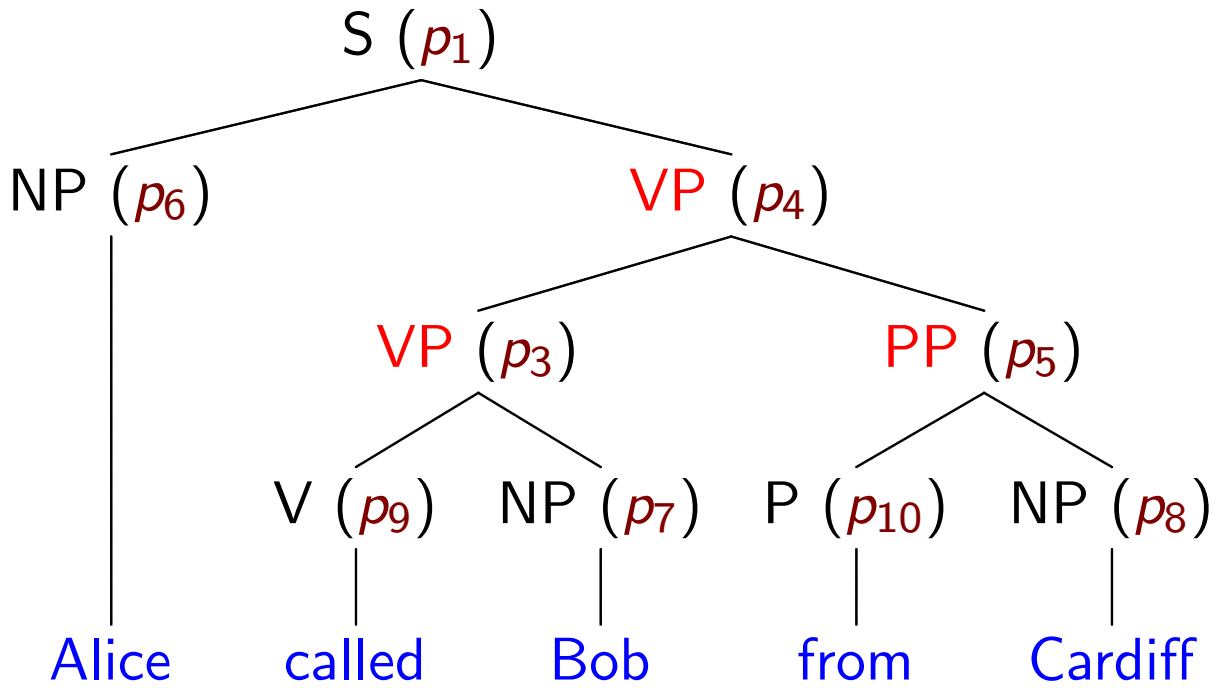


First tree (T1)



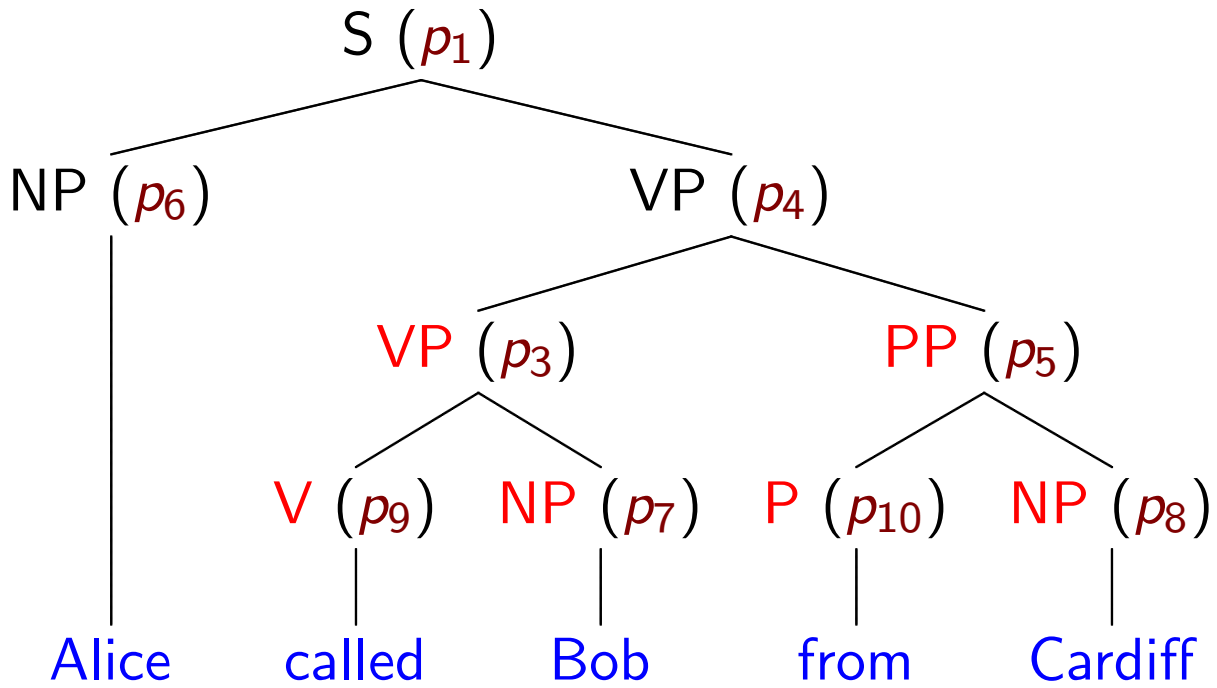
$$P(T1, S) = p_1(p_6)(p_4 \qquad)$$

First tree (T1)



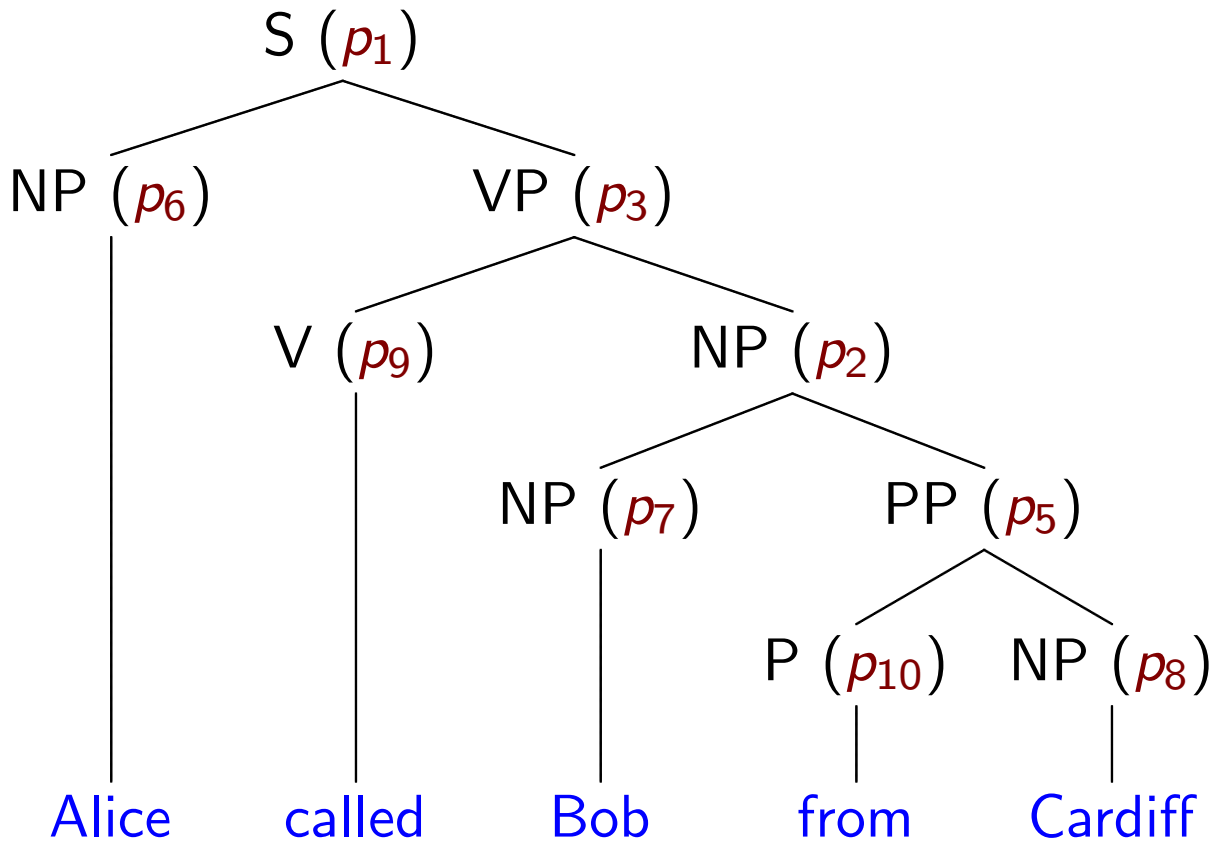
$$P(T1, S) = p_1(p_6)(p_4(p_3 \qquad)(p_5 \qquad))$$

First tree (T1)

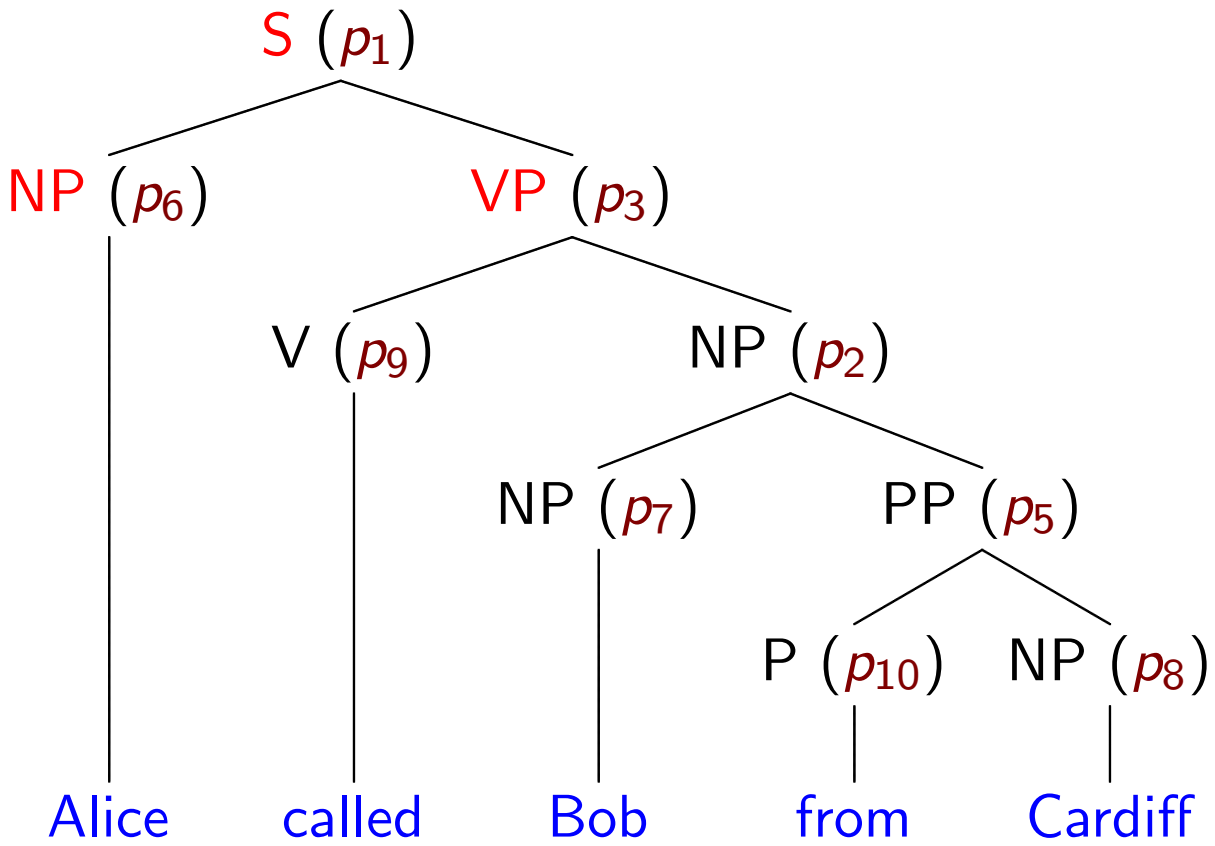


$$P(T1, S) = p_1(p_6)(p_4(p_3(p_9 p_7))(p_5(p_{10} p_8)))$$

Second tree (T2)

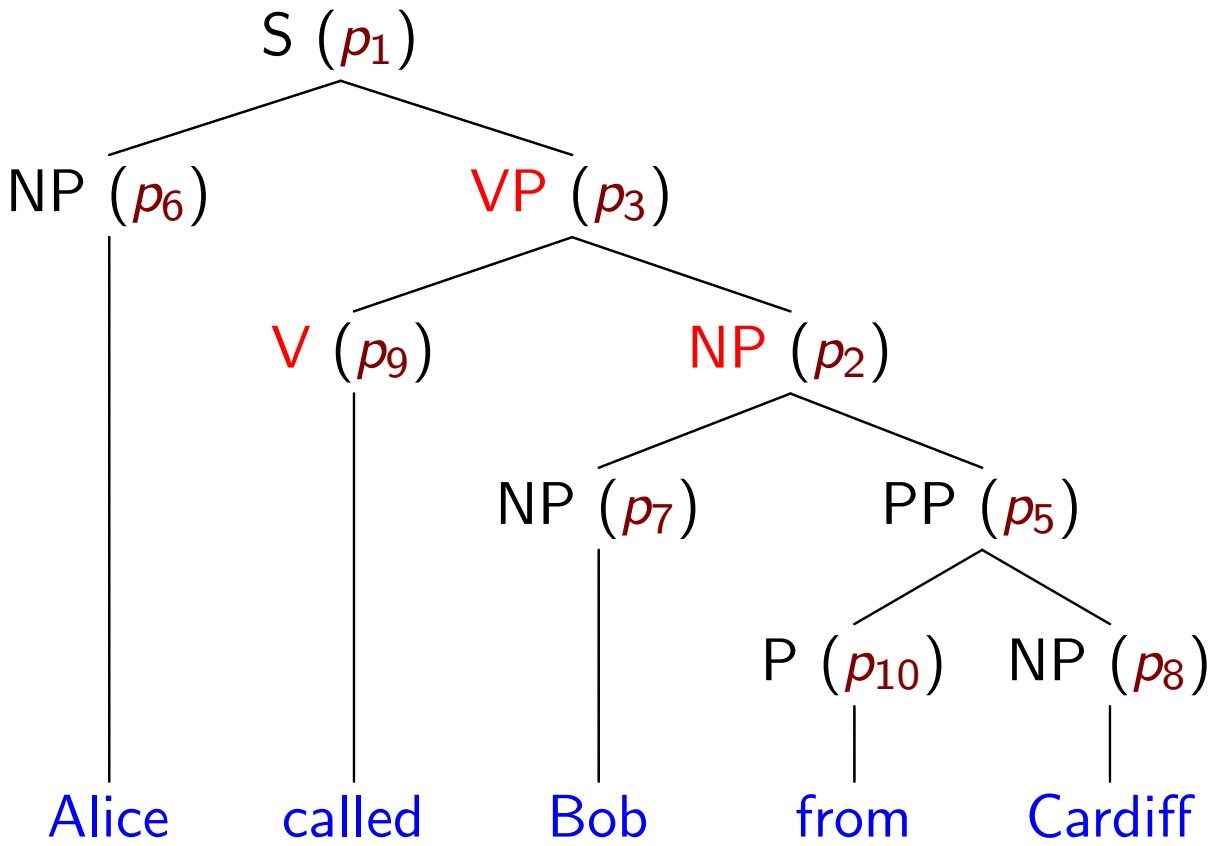


Second tree (T2)



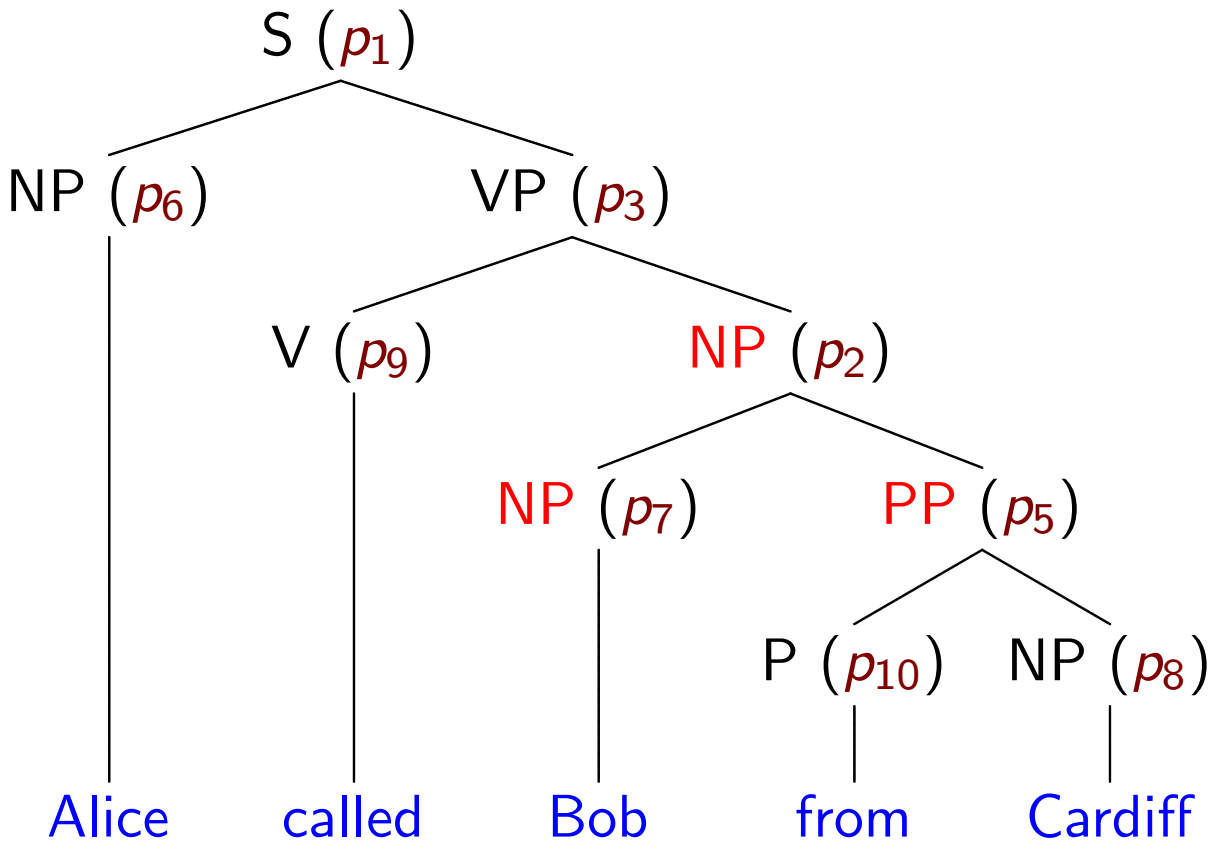
$$P(T2, s) = p_1(p_6)(p_3 \qquad)$$

Second tree (T2)



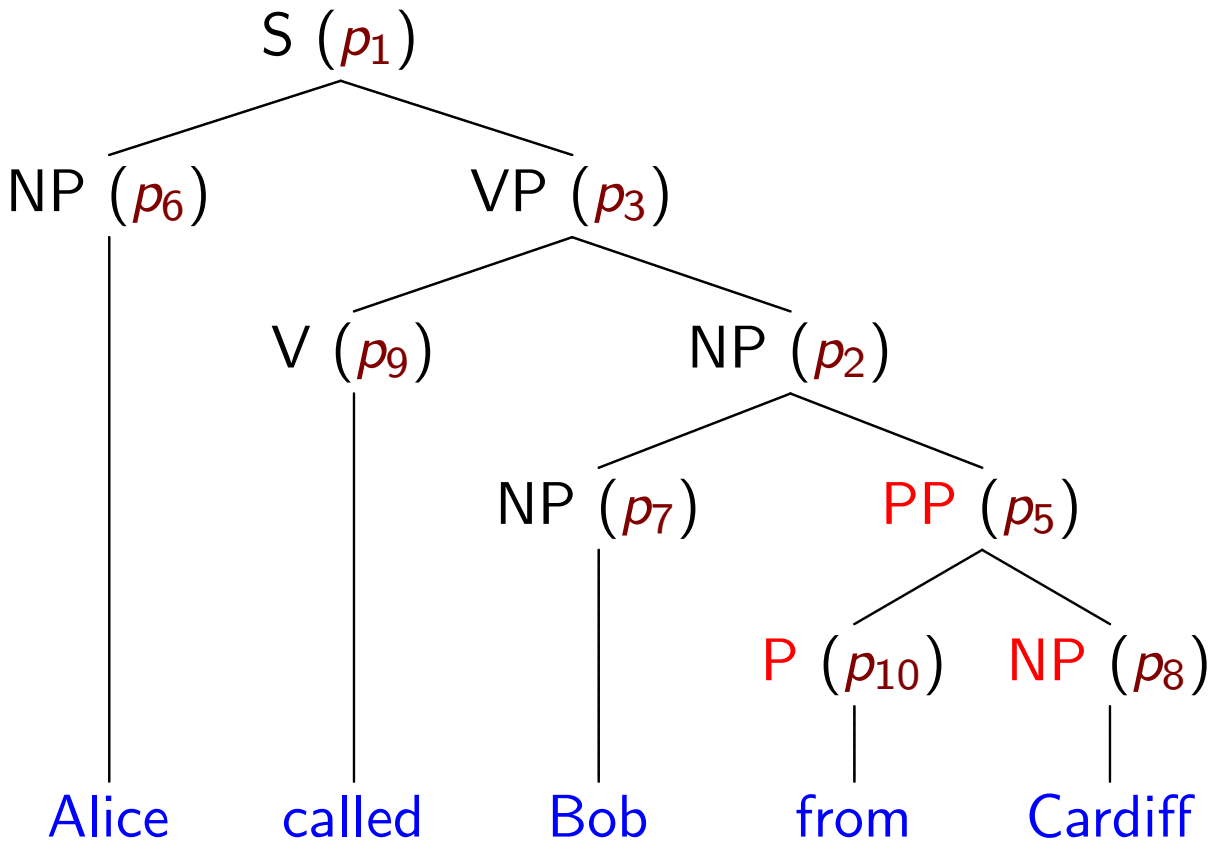
$$P(T2, s) = p_1(p_6)(p_3(p_9)(p_2 \qquad))$$

Second tree (T2)



$$P(T2, s) = p_1(p_6)(p_3(p_9)(p_2(p_7)(p_5 \quad)))$$

Second tree (T2)



$$P(T2, s) = p_1(p_6)(p_3(p_9)(p_2(p_7)(p_5(p_{10}p_8))))$$

Choosing the tree

Choose tree 1 if $P(T1, S)/P(T2, S) > 1$.

$$\frac{P(T1,S)}{P(T2,S)} = \frac{p_1(p_6)(p_4(p_3(p_9p_7))(p_5(p_{10}p_9)))}{p_1(p_6)(p_3(p_9)(p_2(p_7)(p_5(p_{10}p_8))))} = \frac{p_4}{p_2}$$

Probabilistic CYK

				NP p_8
			P p_{10}	Cardiff
		NP p_7	from	
	V p_9	Bob		
NP p_6	called			
Alice				

Probabilistic CYK

			PP $p_5\, p_8\, p_{10}$	NP p_8
		X	P p_{10}	Cardiff
	VP $p_3\, p_7\, p_9$	NP p_7	from	
X	V p_9	Bob		
NP p_6	called			
Alice				

Probabilistic CYK

		NP $p_2\, p_7\, p_5\, p_8\, p_{10}$	PP $p_5\, p_8\, p_{10}$	NP p_8
	X	X	P p_{10}	Cardiff
S $p_1\, p_6\, p_3\, p_7\, p_9$	VP $p_3\, p_7\, p_9$	NP p_7	from	
X	V p_9	Bob		
NP p_6	called			
Alice				

Probabilistic CYK

	VP_1	NP	PP	NP
	$p_4\,p_3\,p_7\,p_9\,p_5\,p_8\,p_{10}$	$p_2\,p_7\,p_5\,p_8\,p_{10}$	$p_5\,p_8\,p_{10}$	p_8
X	X	X	P	Cardiff
			p_{10}	
S	VP	NP	from	
$p_1\,p_6\,p_3\,p_7\,p_9$	$p_3\,p_7\,p_9$	p_7		
X	V	Bob		
	p_9			
NP	called			
p_6				
Alice				

Probabilistic CYK

	VP_2 $p_3\,p_9\,p_2\,p_7\,p_5\,p_8\,p_{10}$	NP $p_2\,p_7\,p_5\,p_8\,p_{10}$	PP $p_5\,p_8\,p_{10}$	NP p_8
X	X	X	P p_{10}	Cardiff
S $p_1\,p_6\,p_3\,p_7\,p_9$	VP $p_3\,p_7\,p_9$	NP p_7	from	
X	V p_9	Bob		
NP p_6	called			
Alice				

Probabilistic CYK

If $p_4 p_3 p_7 p_9 p_5 p_8 p_{10} > p_3 p_9 p_2 p_7 p_5 p_8 p_{10}$:

S $p_1 p_6 p_4 p_3 p_7 p_9 p_5 p_8 p_{10}$	VP_1 $p_4 p_3 p_7 p_9 p_5 p_8 p_{10}$	NP $p_2 p_7 p_5 p_8 p_{10}$	PP $p_5 p_8 p_{10}$	NP p_8
X	X	X	P p_{10}	Cardiff
S $p_1 p_6 p_3 p_7 p_9$	VP $p_3 p_7 p_9$	NP p_7	from	
X	V p_9	Bob		
NP p_6	called			
Alice				

Probabilistic CYK

If $p_4 p_3 p_7 p_9 p_5 p_8 p_{10} < p_3 p_9 p_2 p_7 p_5 p_8 p_{10}$:

S $p_1 p_6 p_3 p_9 p_2 p_7 p_5 p_8 p_{10}$	VP_2 $p_3 p_9 p_2 p_7 p_5 p_8 p_{10}$	NP $p_2 p_7 p_5 p_8 p_{10}$	PP $p_5 p_8 p_{10}$	NP p_8
X	X	X	P p_{10}	Cardiff
S $p_1 p_6 p_3 p_7 p_9$	VP $p_3 p_7 p_9$	NP p_7	from	
X	V p_9	Bob		
NP p_6	called			
Alice				

Estimating PCFG Probabilities

- Treebank—corpus of parsed sentences
- Given a treebank compute the probability of each non-terminal expansion $(A \rightarrow \alpha)$ based on the counts $c(A \rightarrow \alpha)$:

$$P(A \rightarrow \alpha | A) = \frac{c(A \rightarrow \alpha)}{\sum_Y c(A \rightarrow Y)} = \frac{c(A \rightarrow \alpha)}{c(A)}$$

- If a treebank is not available probabilities estimated by parsing the corpus, incrementing a counter for each rule in the parse, and normalizing to get probabilities
- Problem: multiple possible parses for most sentences
- Solution: keep separate counts for each parse, and weight by the probability of the parse. Can be computed using the Inside-Outside algorithm.

Problems with PCFGs

Structural Rule probabilities are independent of location in the parse tree. For example pronouns are much more likely to be subjects than objects

Lexical PCFGs only capture lexical information in the expansion of pre-terminals.
But, lexical dependenciescan often be used to choose the correct parse, eg:
Carol eats chips with ketchup
Carol eats chips with a fork

Lexical dependence

Data from Penn Treebank:

Rule	*come*	*take*	*think*	*want*
VP $\rightarrow$ V	0.095	0.026	0.046	0.057
VP $\rightarrow$ V NP	0.011	0.321	0.002	0.139
VP $\rightarrow$ V PP	0.345	0.031	0.071	0.003
VP $\rightarrow$ V S	0.022	0.013	0.048	0.708

The rule used to expand VP is strongly dependent on the verb

Lexicalised PCFGs

- Annotated each non-terminal with its *lexical head*
- Each rule has a head child on the left hand side; the headword for a node is then the headword of its head child
- Easy for simple examples (eg the noun is the head of an NP, the verb is the head of a VP); harder in practice
- Various heuristics for finding the head robustly (mainly developed on Penn Treebank)
- A "simple" lexicalised CFG is a basic CFG with a lot more rules (ie each rule is copied for each headword) — but this is impractical!
- Make some independence assumptions and condition the probability of each rule on the head (or nearby heads)

Summary

- **Reading:** J+M, chapter 12 (12.1 - 12.3)
- Bottom-up CYK parsing of CFGs
- Probabilistic CFGs and probabilistic parsing using CYK
- Lexical and structural problems with PCFGs; lexicalised PCFGs.