

Natural Language Processing : Probabilistic Context Free Grammars



Updated 5/09

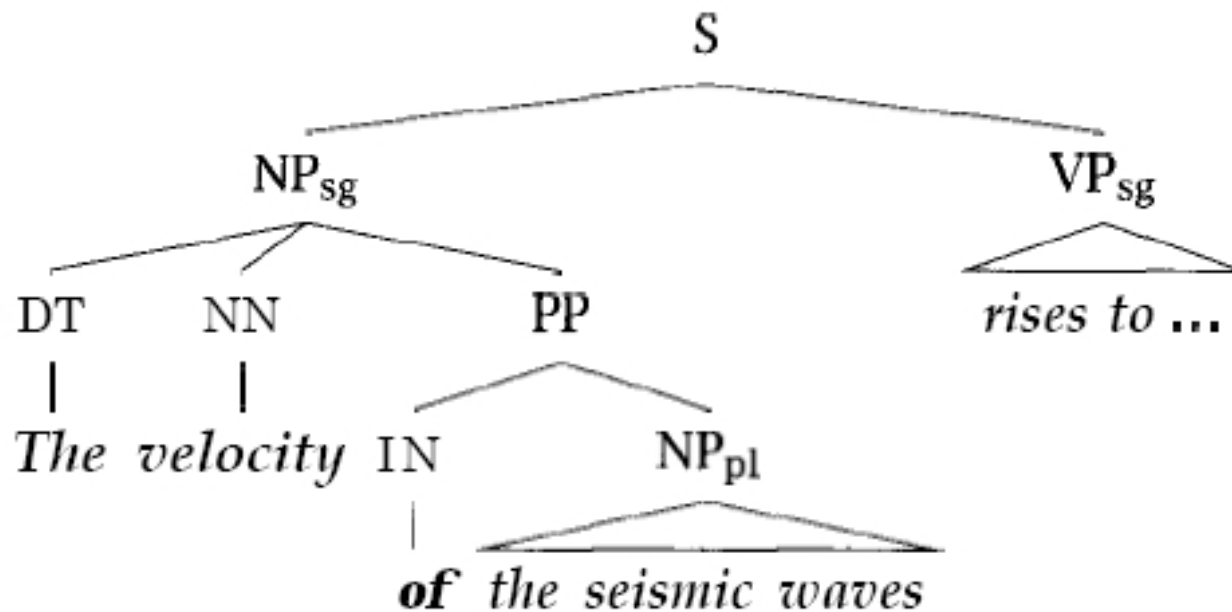


Motivation

- N-gram models and HMM Tagging only allowed us to process sentences linearly.
- However, even simple sentences require a nonlinear model that reflects the hierarchical structure of sentences rather than the linear order of words.
- For example, HMM taggers will have difficulties with long range dependencies like in *The velocity of the seismic waves **rises** to ...*

Phrase hierarchical structure

- Verb agrees in number with the noun velocity which is the head of the preceding noun phrase.





PCFGs

- Probabilistic Context Free Grammars are the simplest and most natural probabilistic model for tree structures and the algorithms for them are closely related to those for HMMs.
- Note, however, that there are other ways of building probabilistic models of syntactic structure (see Chapter 12).



Formal Definition of PCFGs

A PCFG consists of:

- A set of terminals, $\{w^k\}$, $k = 1, \dots, V$
- A set of nonterminals, N^i , $i = 1, \dots, n$
- A designated start symbol N^1
- A set of rules, $\{N^i \rightarrow \zeta^j\}$, (where ζ^j is a sequence of terminals and nonterminals)
- A corresponding set of probabilities on rules such that: $\forall i \sum_j P(N^i \rightarrow \zeta^j) = 1$
- The probability of a sentence (according to grammar G) is given by:
$$P(w_{1m}) = \sum_t P(w_{1m}, t)$$
 where t is a parse tree of the sentence.



Notation

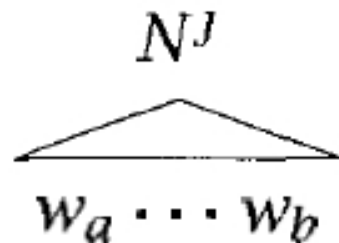
Notation	Meaning
G	Grammar (PCFG)
\mathcal{L}	Language (generated or accepted by a grammar)
t	Parse tree
$\{N^1, \dots, N^n\}$	Nonterminal vocabulary (N^1 is start symbol)
$\{w^1, \dots, w^V\}$	Terminal vocabulary
$w_1 \dots w_m$	Sentence to be parsed
N_{pq}^j	Nonterminal N^j spans positions p through q in string
$\alpha_j(p, q)$	Outside probabilities
$\beta_j(p, q)$	Inside probabilities



Domination

- A non-terminal N^j is said to *dominate* the words $w_a \dots w_b$ if it is possible to rewrite the non-terminal N^j as a sequence of words $w_a \dots w_b$.
- Alternative notations:

$$\text{yield}(N^j) = w_a \dots w_b \quad \text{or} \quad N^j \xRightarrow{*} w_a \dots w_b$$





Assumptions of the Model

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

$\forall k \quad P(N_{k(k+c)}^j \rightarrow \zeta)$ is the same

- **Context Free:** The probability of a subtree does not depend on words not dominated by the subtree.

$P(N_{kl}^j \rightarrow \zeta | \text{anything outside } k \text{ through } l) = P(N_{kl}^j \rightarrow \zeta)$

- **Ancestor Free:** The probability of a subtree does not depend on nodes in the derivation outside the subtree.

$P(N_{kl}^j \rightarrow \zeta | \text{any ancestor nodes outside } N_{kl}^j) = P(N_{kl}^j \rightarrow \zeta)$

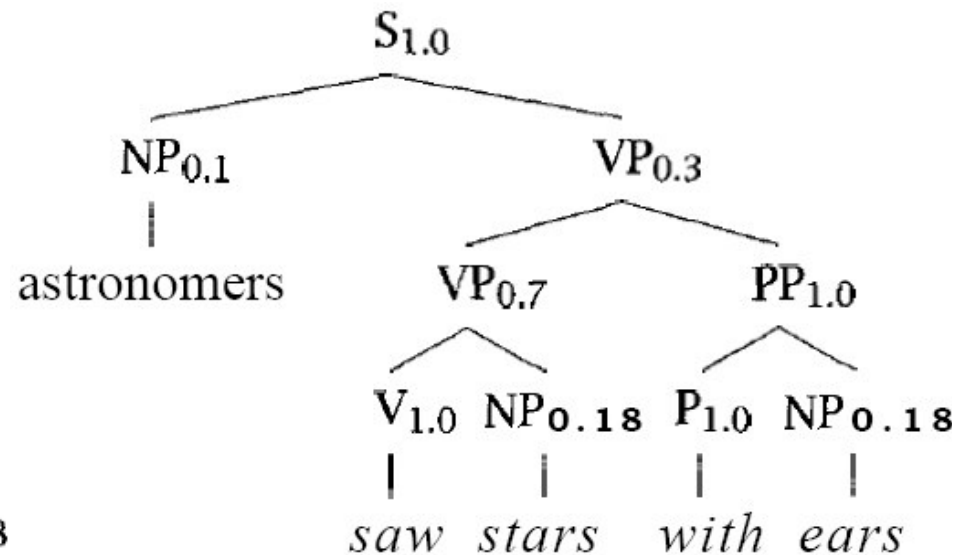
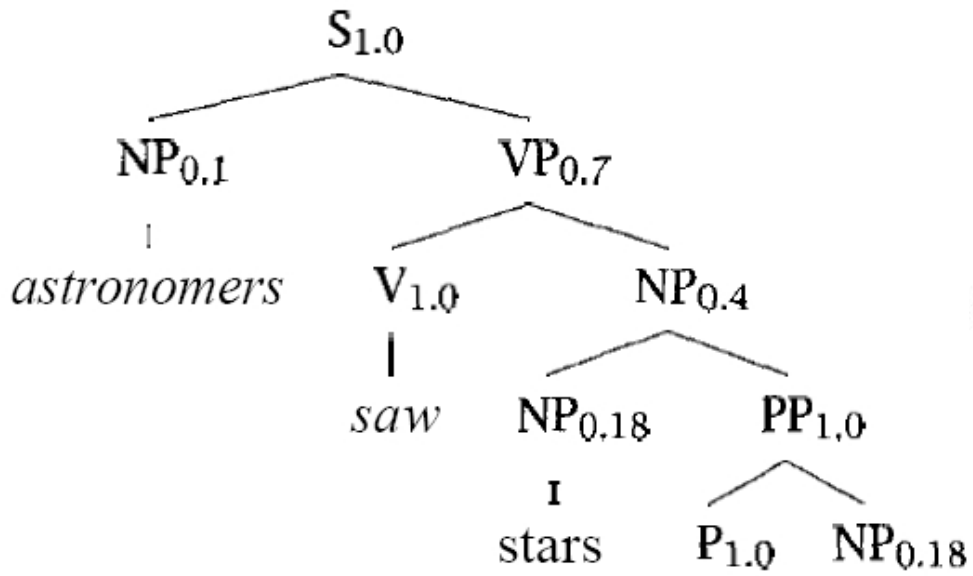


Example: simple PCFG

S --> NP VP	1.0	NP --> NP PP	0.4
PP --> P NP	1.0	NP --> <i>astronomers</i>	0.1
VP --> V NP	0.7	NP --> <i>ears</i>	0.18
VP --> VP PP	0.3	NP --> <i>saw</i>	0.04
P --> <i>with</i>	1.0	NP --> <i>stars</i>	0.18
V --> <i>saw</i>	1.0	NP --> <i>telescopes</i>	0.1

$P(\text{astronomers saw stars with ears}) = ?$

Example – parsing according to simple grammar



with ears

$$P(t_1) = 0.0009072$$

$$P(t_2) = 0.0006804$$

$$P(w_{1m}) = P(t_1) + P(t_2) = 0.0015876$$



Some Features of PCFGs

- A PCFG gives some idea of the plausibility of different parses. However, the probabilities are based on structural factors and not lexical ones.
- PCFG are good for grammar induction.
- PCFGs are robust.
- PCFGs give a probabilistic language model for English.
- The predictive power of a PCFG (measured by entropy) tends to be greater than for an HMM.
- PCFGs are not good models alone but they can be combined with a tri-gram model.
- PCFGs have certain biases which may not be appropriate.



Improper probability & PCFGs

- Is $\sum_w P(w) = 1$ always satisfied for PCFGs?

- Consider the grammar
generating the language

ab $P = 1/3$

abab $P = 2/27$

ababab $P = 8/243$

$S \rightarrow ab$	$P = 1/3$
$S \rightarrow S S$	$P = 2/3$

- $\sum_w P(w) = 1/3 + 2/27 + 8/243 + 40/2187 + \dots = 1/2$
- This is improper probability distribution.
- PCFGs learned from parsed training corpus always give a proper probability distribution (Chi, Geman 1998)



Questions for PCFGs

- Just as for HMMs, there are three basic questions we wish to answer:
- 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: $\mathit{argmax}_t P(t/w_{1m}G)$?
- How can we choose rule probabilities for the grammar G that maximizes the probability of a sentence, $\mathit{argmax}_G P(w_{1m}/G)$?



Restriction

- Here, we only consider the case of **Chomsky Normal Form Grammars**, which only have unary and binary rules of the form:
 - $N^i \rightarrow N^j N^k$
 - $N^i \rightarrow w^j$
- The parameters of a PCFG in Chomsky Normal Form are:
 - $P(N^j \rightarrow N^r N^s \mid G)$, an n^3 matrix of parameters
 - $P(N^j \rightarrow w^k \mid G)$, nV parameters
(where n is the number of nonterminals and V is the number of terminals)
- $\sum_{r,s} P(N^j \rightarrow N^r N^s) + \sum_k P(N^j \rightarrow w^k) = 1$



From HMMs to Probabilistic Regular Grammars (PRG)

- A **PRG** has start state N^1 and rules of the form:
 - $N^i \rightarrow w^j N^k$
 - $N^i \rightarrow w^j$
- This is similar to what we had for an HMM except that in an HMM, we have $\forall n \sum_{w \in L} P(w_{1n}) = 1$ whereas in a PCFG, we have $\sum_{w \in L} P(w) = 1$ where L is the language generated by the grammar and w is a sentence in the language.
- To see the difference consider the probability $P(\text{John decide to bake a})$ modeled by HMM and by PCFG.
- PRG are related to HMMs in that a PRG is a HMM to which we should add a start state and a finish (or sink) state.



From PRGs to PCFGs

- In the HMM, we were able to efficiently do calculations in terms of forward and backward probabilities.
- In a parse tree, the forward probability corresponds to everything above and including a certain node, while the backward probability corresponds to the probability of everything below a certain node.

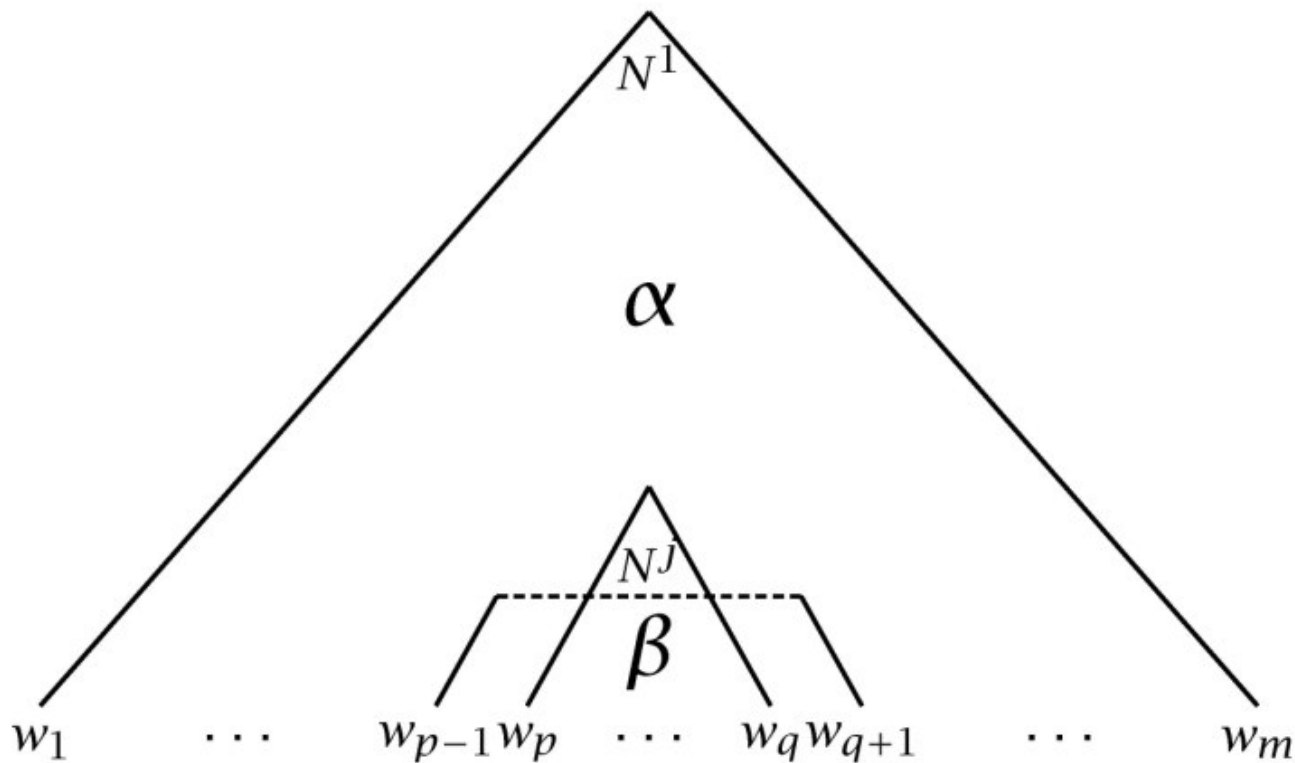
- We introduce outside (α_j) and inside (β_j) Probs.:

- $\alpha_j(p, q) = P(w_{1(p-1)}, N_{pq}^j, w_{(q+1)m} / \mathbf{G})$
- $\beta_j(p, q) = P(w_{pq} / N_{pq}^j, \mathbf{G})$



Inside and outside probabilities

- $\alpha_j(p, q)$ and $\beta_j(p, q)$



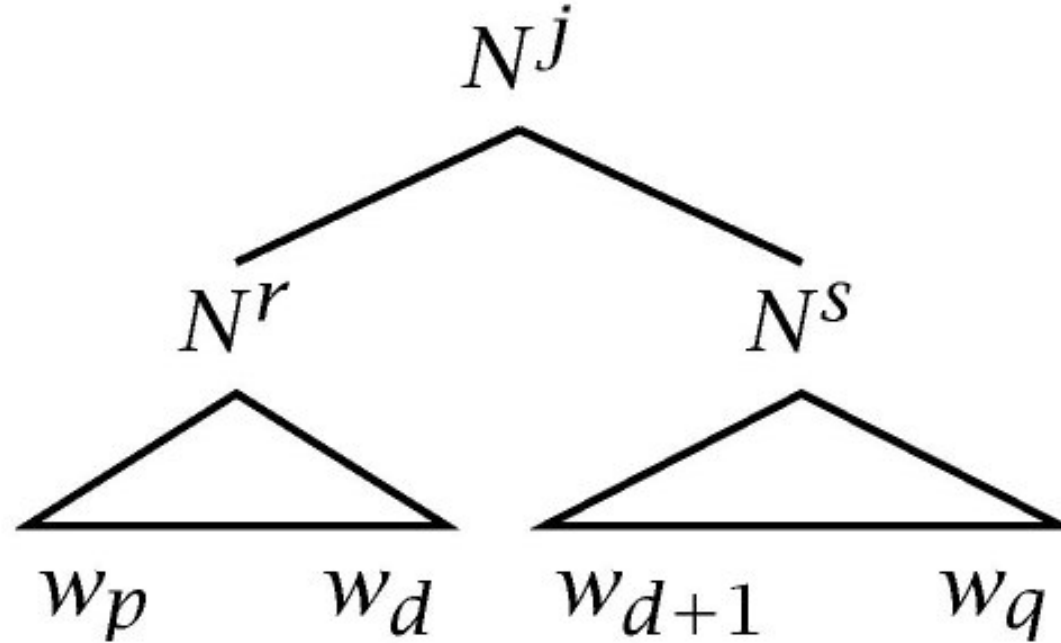


The Probability of a String I: Using Inside Probabilities

- We use the ***Inside Algorithm***, a dynamic programming algorithm based on the inside probabilities:
- $$\begin{aligned} P(w_{1m} / G) &= P(N^1 \Rightarrow^* w_{1m} / G) \\ &= P(w_{1m} / N_{1m}^1, G) \\ &= \beta_1(\mathbf{1}, m) \end{aligned}$$
- ***Base Case:***
$$\begin{aligned} \beta_j(k, k) &= P(w_k / N_{kk}^j, G) \\ &= P(N^j \rightarrow w_k / G) \end{aligned}$$
- ***Induction:***
$$\beta_j(p, q) = \sum_{r,s} \sum_{d=p}^{q-1} P(N^j \rightarrow N^r N^s) \beta_r(p, d) \beta_s(d+1, q)$$



Inside probs - inductive step





The simple PCFG

S --> NP VP	1.0	NP --> NP PP	0.4
PP --> P NP	1.0	NP --> <i>astronomers</i>	0.1
VP --> V NP	0.7	NP --> <i>ears</i>	0.18
VP --> VP PP	0.3	NP --> <i>saw</i>	0.04
P --> <i>with</i>	1.0	NP --> <i>stars</i>	0.18
V --> <i>saw</i>	1.0	NP --> <i>telescopes</i>	0.1

$P(\text{astronomers saw stars with ears}) = ?$

The Probability of a String II: example of inside probabilities

	1	2	3	4	5
1	$\beta_{NP} = 0.1$		$\beta_S = 0.0126$		$\beta_S = 0.001587$
2		$\beta_{NP} = 0.04$ $\beta_V = 1.0$	$\beta_{VP} = 0.126$		$\beta_{VP} = 0.01587$
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	<i>astronomers</i>	<i>saw</i>	<i>stars</i>	<i>with</i>	<i>ears</i>



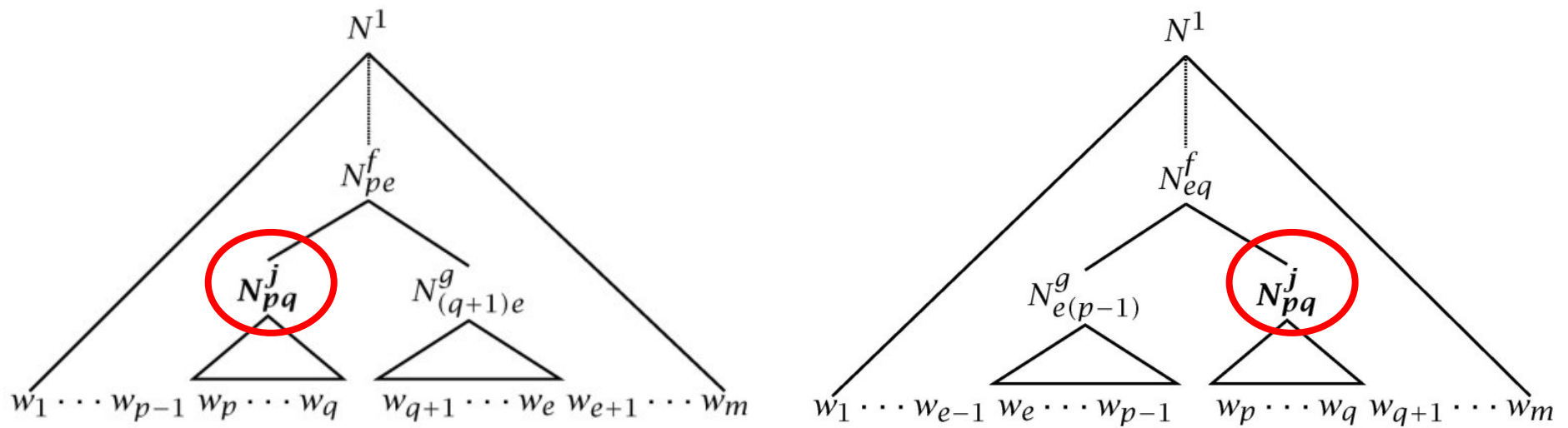
The Probability of a String III: Using Outside Probabilities

- We use the Outside Algorithm based on the outside probabilities:

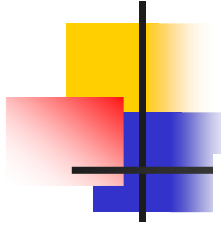
$$\begin{aligned} P(w_{1m} / G) &= \sum_j P(w_{1(k-1)} w_k w_{(k+1)m} N^j_{kk} / G) \\ &= \sum_j \alpha_j(k, k) P(N^j \rightarrow w_k) \quad \text{for any } k \end{aligned}$$

- **Base Case:** $\alpha_1(\mathbf{1}, m) = \mathbf{1}$; $\alpha_j(\mathbf{1}, m) = \mathbf{0}$ for $j \neq 1$
- **Inductive Case:** the node N^j_{pq} might be either on the left branch or the right branch of the parent node

outside probs - inductive step



- We sum over the two contributions



outside probs - inductive step II

$$\begin{aligned}
 \alpha_j(p, q) &= \left[\sum_{f, g} \sum_{e=q+1}^m P(w_{1(p-1)}, w_{(q+1)m}, N_{pe}^f, N_{pq}^j, N_{(q+1)e}^g) \right] \\
 &\quad + \left[\sum_{f, g} \sum_{e=1}^{p-1} P(w_{1(p-1)}, w_{(q+1)m}, N_{eq}^f, N_{e(p-1)}^g, N_{pq}^j) \right] \\
 &= \left[\sum_{f, g} \sum_{e=q+1}^m P(w_{1(p-1)}, w_{(e+1)m}, N_{pe}^f) P(N_{pq}^j, N_{(q+1)e}^g | N_{pe}^f) \right. \\
 &\quad \left. \times P(w_{(q+1)e} | N_{(q+1)e}^g) \right] + \left[\sum_{f, g} \sum_{e=1}^{p-1} P(w_{1(e-1)}, w_{(q+1)m}, N_{eq}^f) \right. \\
 &\quad \left. \times P(N_{e(p-1)}^g, N_{pq}^j | N_{eq}^f) P(w_{e(p-1)} | N_{e(p-1)}^g) \right] \\
 &= \left[\sum_{f, g} \sum_{e=q+1}^m \alpha_f(p, e) P(N^f \rightarrow N^j N^g) \beta_g(q+1, e) \right] \\
 &\quad + \left[\sum_{f, g} \sum_{e=1}^{p-1} \alpha_f(e, q) P(N^f \rightarrow N^g N^j) \beta_g(e, p-1) \right]
 \end{aligned}$$



Combining inside and outside

- Similarly to the HMM, we can combine the inside and the outside probabilities:
- $\alpha_j(\mathbf{p}, \mathbf{q}) \beta_j(\mathbf{p}, \mathbf{q}) = P(\mathbf{w}_{1m} N_{pq}^j / \mathbf{G})$
- **Notice** $\alpha_1(\mathbf{1}, \mathbf{m}) \beta_1(\mathbf{1}, \mathbf{m}) = P(\mathbf{w}_{1m} N_{1m}^1 / \mathbf{G})$
 $= P(\mathbf{w}_{1m} / \mathbf{G})$
- And the probability that there is any non-terminal yielding $w_{p,q}$ is
- $P(\mathbf{w}_{1m} N_{pq} / \mathbf{G}) = \sum_j \alpha_j(\mathbf{p}, \mathbf{q}) \beta_j(\mathbf{p}, \mathbf{q})$



Finding the Most Likely Parse for a Sentence

- The algorithm works by finding the highest probability partial parse tree spanning a certain substring that is rooted with a certain nonterminal.
- $\delta_i(p, q)$ = the highest inside probability parse of a subtree N_{pq}^i
- **Initialization:** $\delta_i(p, p) = P(N^i \rightarrow w_p)$
- **Induction:** $\delta_i(p, q) = \max_{j, k, p \leq r < q} P(N^i \rightarrow N^j N^k) \delta_j(p, r) \delta_k(r+1, q)$
- **Store backtrace:** $\psi_i(p, q) = \operatorname{argmax}_{(j, k, r)} P(N^i \rightarrow N^j N^k) \delta_j(p, r) \delta_k(r+1, q)$
- **Termination:** $P(\hat{t}) = \delta_1(1, m)$



Training a PCFG

- **Restrictions:** We assume that the set of rules is given in advance and we try to find the optimal probabilities to assign to different grammar rules.
- Like for the HMMs, we use an EM Training Algorithm called the **Inside-Outside Algorithm** which allows us to train the parameters of a PCFG on unannotated sentences of the language.
- **Basic Assumption:** a good grammar is one that makes the sentences in the training corpus likely to occur \implies we seek the grammar that maximizes the likelihood of the training data.



Problems with the inside- outside algorithm

- Extremely Slow: For each sentence, each iteration of training is $O(m^3n^3)$.
- Local Maxima are much more of a problem than in HMMs
- Satisfactory learning requires many more nonterminals than are theoretically needed to describe the language.
- There is no guarantee that the learned nonterminals will be linguistically motivated.