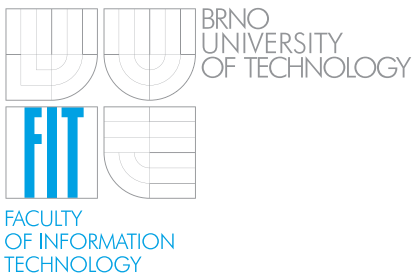
Probabilistic Context-Free Grammar

Petr Horáček, Eva Zámečníková and Ivana Burgetová

Department of Information Systems
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 00 Brno, CZ



FRVŠ MŠMT FR97/2011/G1

Outline

- **Probabilistic Context-Free Grammar**
  - Definition and examples
  - Properties and usage

Outline

- **Probabilistic Context-Free Grammar**
  - Definition and examples
  - Properties and usage
- **Inside and Outside Probabilities**
  - Definitions
  - Algorithms

Outline

- **Probabilistic Context-Free Grammar**
  - Definition and examples
  - Properties and usage
- **Inside and Outside Probabilities**
  - Definitions
  - Algorithms
- **Inside-Outside Algorithm**
  - Idea, formal description and properties

- **Probabilistic Context-Free Grammar**
  - Definition and examples
  - Properties and usage
- **Inside and Outside Probabilities**
  - Definitions
  - Algorithms
- **Inside-Outside Algorithm**
  - Idea, formal description and properties

- A probabilistic context-free grammar (PCFG; also called stochastic CFG, SCFG) is a context-free grammar, where a certain probability is assigned to each rule.
  - Thus, some derivations become more likely than other.

Definition

A PCFG G is a quintuple $G = (M, T, R, S, P)$, where

- $M = \{N^i : i = 1, \ldots, n\}$ is a set of *nonterminals*
- $T = \{w^k : k = 1, \ldots, V\}$ is a set of *terminals*
- $R = \{N^i \to \zeta^j \ : \zeta^j \in (M \cup T)^*\}$ is a set of *rules*
- $S = N^1$ is the *start symbol*
- P is a corresponding set of *probabilities* on rules such that

$$\forall i \sum_j P(N^i \to \zeta^j) = 1$$

Notations

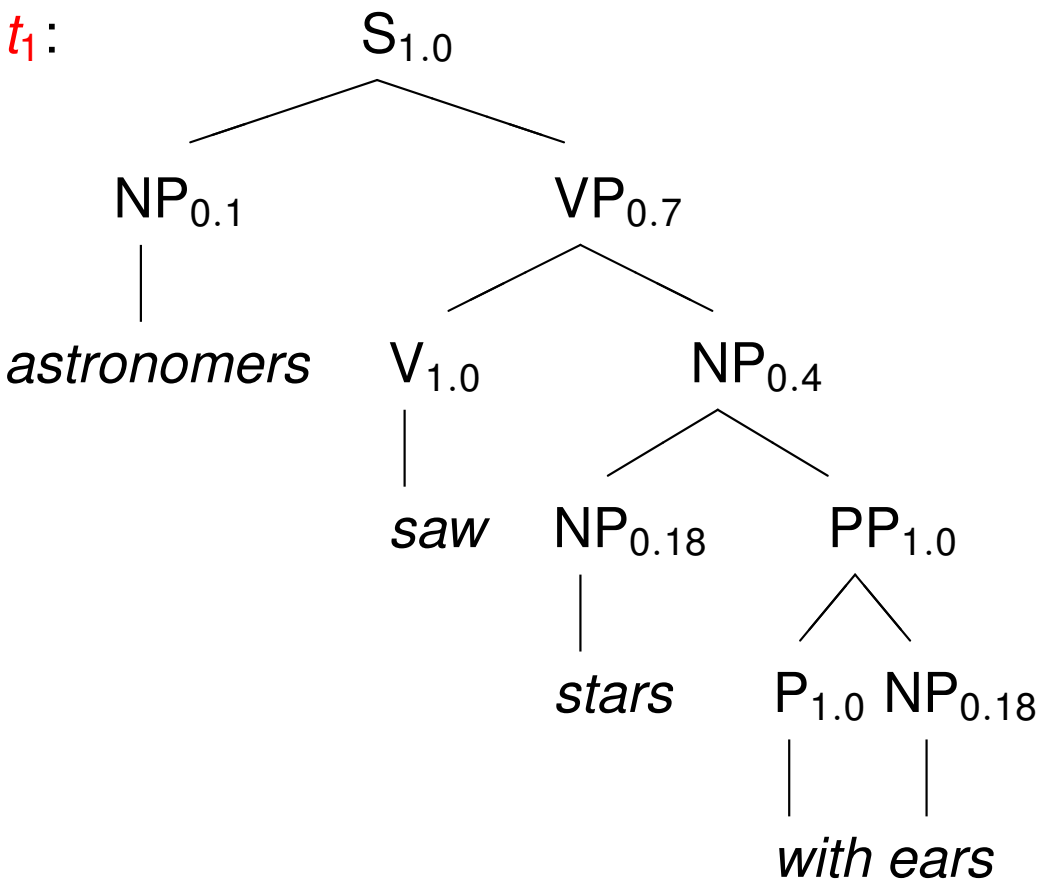
G	Grammar (PCFG)
$L(G)$	Language generated by grammar G
t	Parse tree
$\{N^1, \ldots, N^n\}$	Nonterminal vocabulary
$\{w^1, \ldots, w^V\}$	Terminal vocabulary
N^1	Start symbol
$w_1 \ldots w_m$	Sentence to be parsed
N^j_{pq}	Nonterminal N^j spans positions p through q in string
$\alpha_j(p, q)$	Outside probabilities
$\beta_j(p, q)$	Inside probabilities

PCFG – Example

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

PCFG – Example

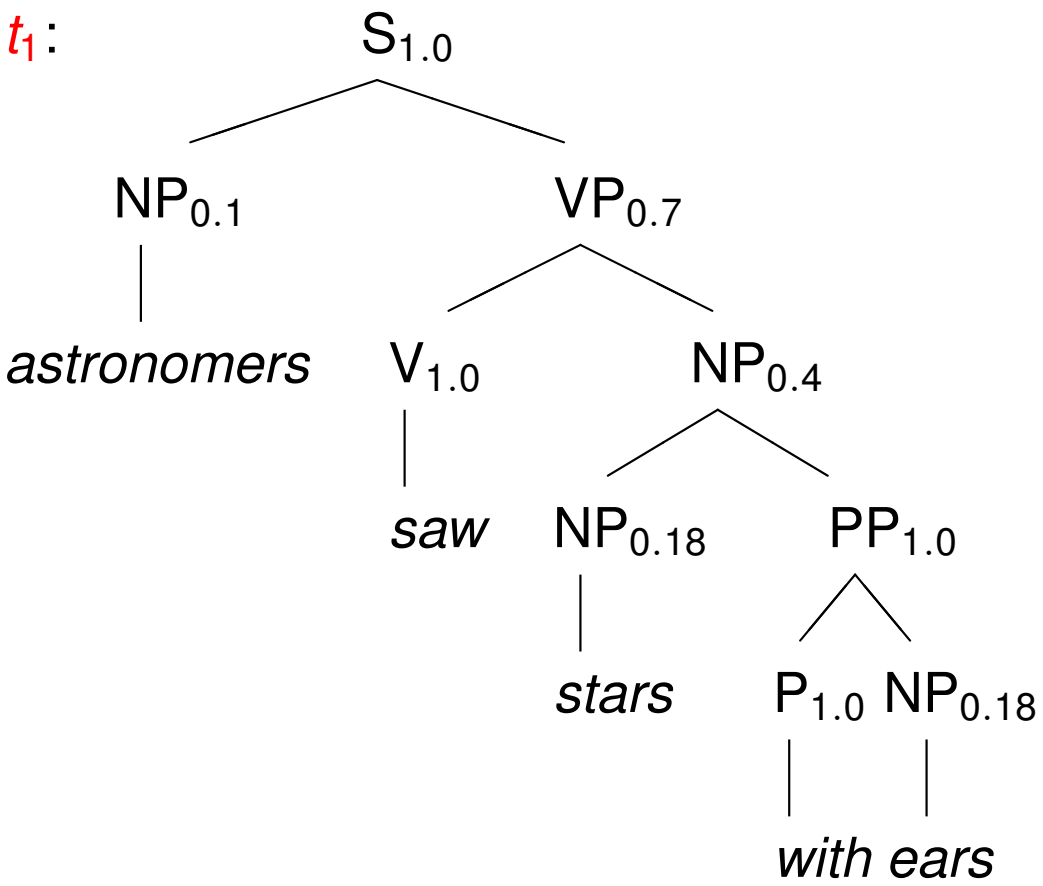
S → NP VP	1.0
PP → P NP	1.0
VP → V NP	0.7
VP → VP PP	0.3
P → *with*	1.0
V → *saw*	1.0
NP → NP PP	0.4
NP → *astronomers*	0.1
NP → *ears*	0.18
NP → *saw*	0.04
NP → *stars*	0.18
NP → *telescopes*	0.1



PCFG – Example

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

t_1:



$$P(t_1) = 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \times 0.18 \times 1.0 \times 1.0 \times 0.18$$
$$= 0.0009072$$

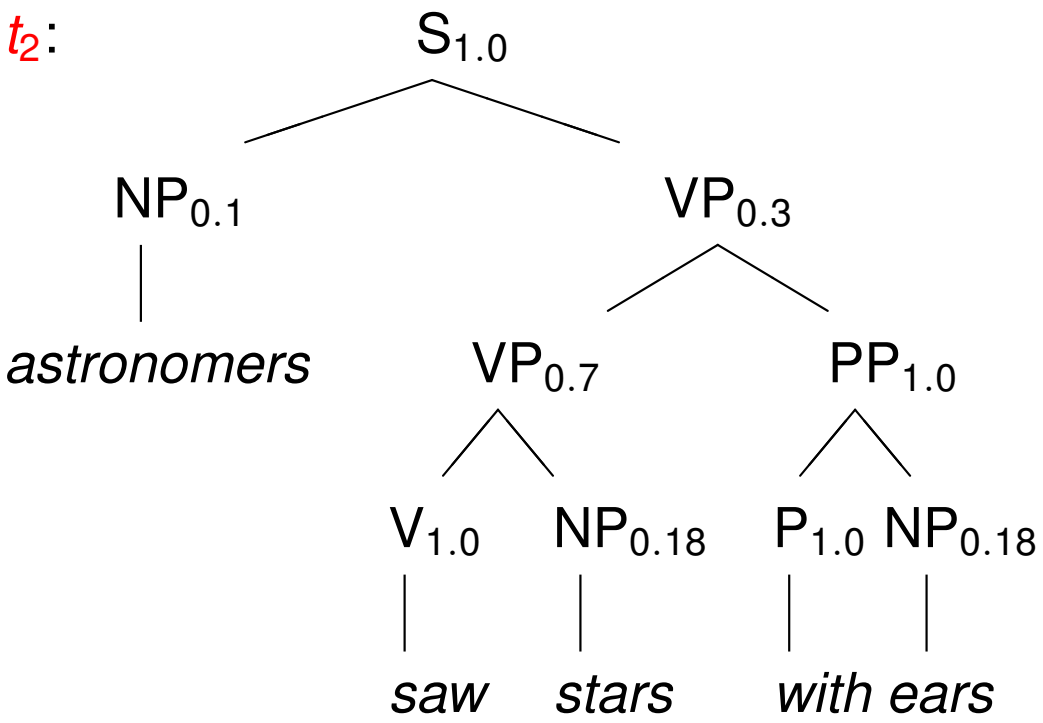
PCFG – Example

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

PCFG – Example

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

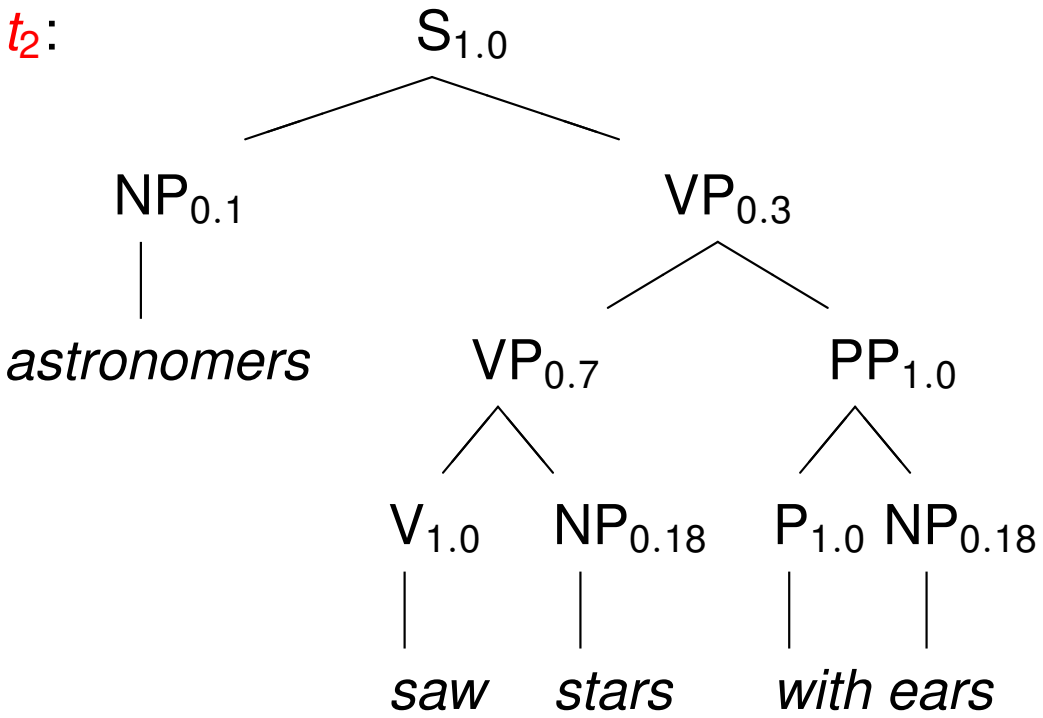
t_2:



PCFG – Example

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

t_2:



$$P(t_2) = 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \times 0.18 \times 1.0 \times 1.0 \times 0.18$$
$$= 0.0006804$$

PCFG – Example

- For the sentence

  astronomers saw stars with ears,

  we can construct 2 parse trees.

$$\begin{aligned} P(t_1) &= 0.0009072 \\ P(t_2) &= 0.0006804 \end{aligned}$$

- Sentence probability:

$$\begin{aligned} P(w_{15}) &= P(t_1) + P(t_2) \\ P(w_{15}) &= 0.0009072 + 0.0006804 \\ P(w_{15}) &= 0.0015876 \end{aligned}$$

1. Place invariance

$$\forall k, l \;\; P(N^j_{k(k+c)} \rightarrow \zeta) = P(N^j_{l(l+c)} \rightarrow \zeta)$$

1. Place invariance

$$\forall k, l \;\; P(N^j_{k(k+c)} \rightarrow \zeta) = P(N^j_{l(l+c)} \rightarrow \zeta)$$

2. Context-free

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

1. Place invariance

$$\forall k, l \;\; P(N^j_{k(k+c)} \rightarrow \zeta) = P(N^j_{l(l+c)} \rightarrow \zeta)$$

2. Context-free

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

3. Ancestor-free

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

PCFG – Features

- Gives a probabilistic language model.
- Can give some idea of the plausibility of different parses of ambiguous sentences.
  - However, only structure is taken into account, no lexical co-occurence.
- Good for grammar induction.
  - Can be learned from positive data alone.
- Robust, able to deal with grammatical mistakes.
- In practice, PCFG shows to be a worse language model for English than *n*-gram models (no lexical context).
- However, we could combine the strengths of PCFGs (sentence structure) and *n*-gram models (lexical co-ocurence).

PCFG – Questions

1. Probability of a sentence w_{1m} according to grammar G:

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

- We will consider grammars in Chomsky Normal Form (without loss of generality).

PCFG – Questions

1. Probability of a sentence w_{1m} according to grammar G:

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

2. The most likely parse for a sentence:

$$\arg\max_t P(t|w_{1m}, G) = \ ?$$

- We will consider grammars in Chomsky Normal Form (without loss of generality).

PCFG – Questions

1. Probability of a sentence w_{1m} according to grammar G:

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

2. The most likely parse for a sentence:

$$\arg\max_t P(t|w_{1m}, G) = \ ?$$

3. Setting rule probabilities to maximize the probability of a sentence:

$$\arg\max_G P(w_{1m}|G) = \ ?$$

- We will consider grammars in Chomsky Normal Form (without loss of generality).

- **Probabilistic Context-Free Grammar**
  - Definition and examples
  - Properties and usage
- **Inside and Outside Probabilities**
  - Definitions
  - Algorithms
- **Inside-Outside Algorithm**
  - Idea, formal description and properties

Sentence Probability

Definition

Sentence probability of a sentence w_{1m} according to grammar G:

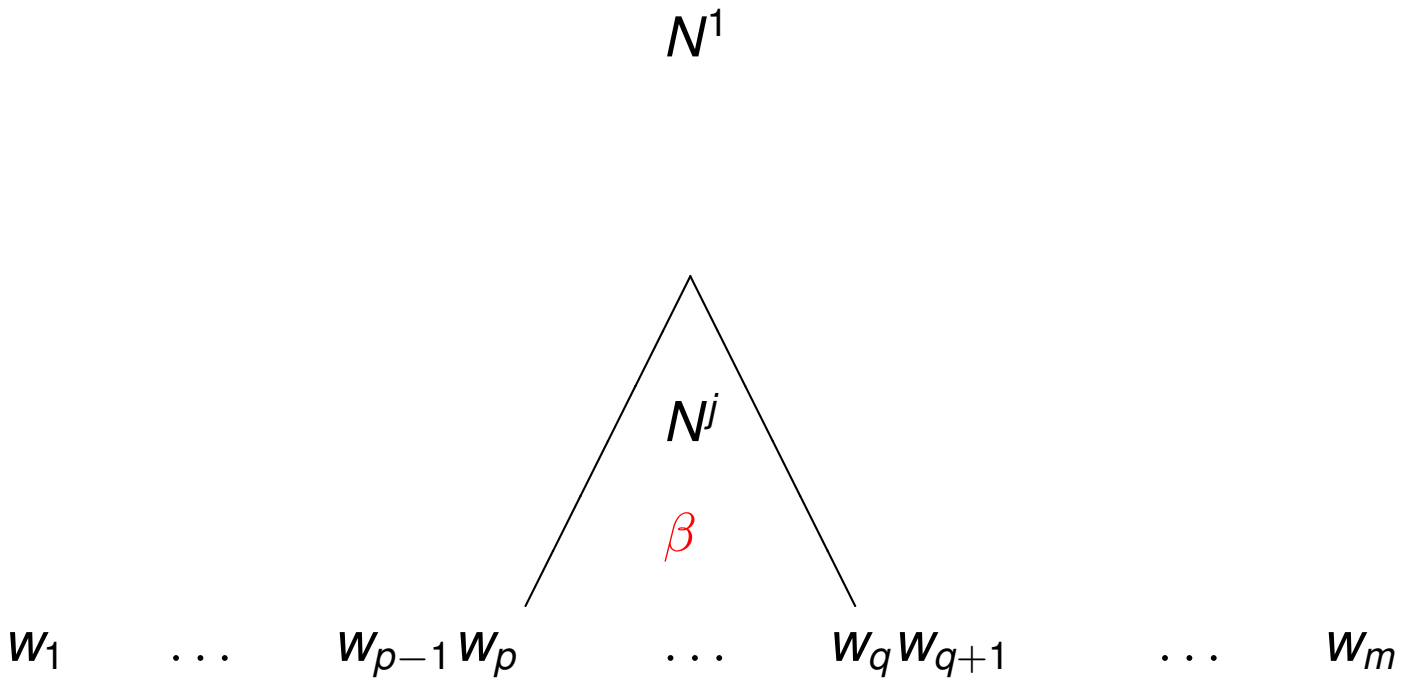
$$P(w_{1m}|G) = \sum_t P(w_{1m}, t)$$

where t is a parse tree of the sentence.

- Trivial solution:
  Find all parse trees, calculate and sum up their probabilities.
- Problem:
  Exponential time complexity in general - unsuitable in practice.
- Efficient solution:
  Using inside and outside probabilities.

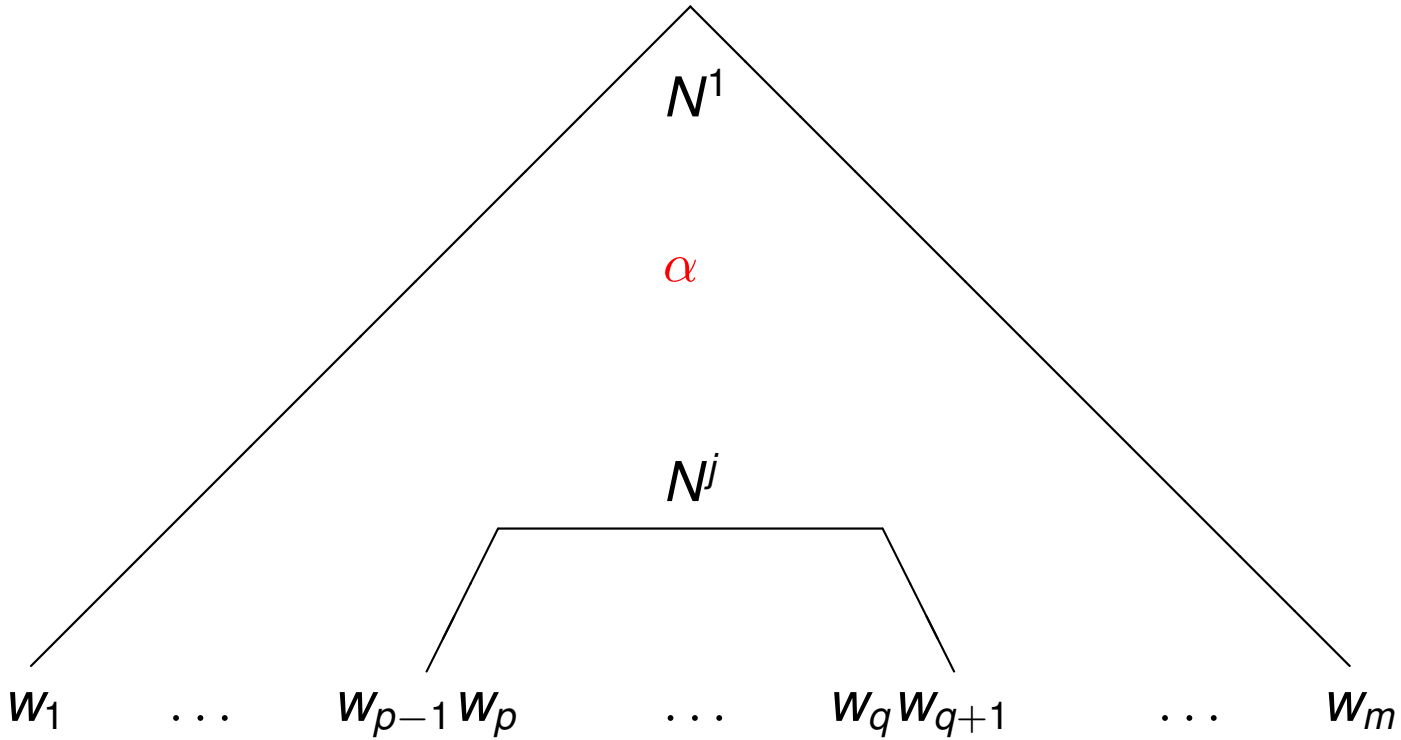
Definition

- Inside probability: $\beta_j(p,q) = P(w_{pq}|N^j_{pq}, G)$



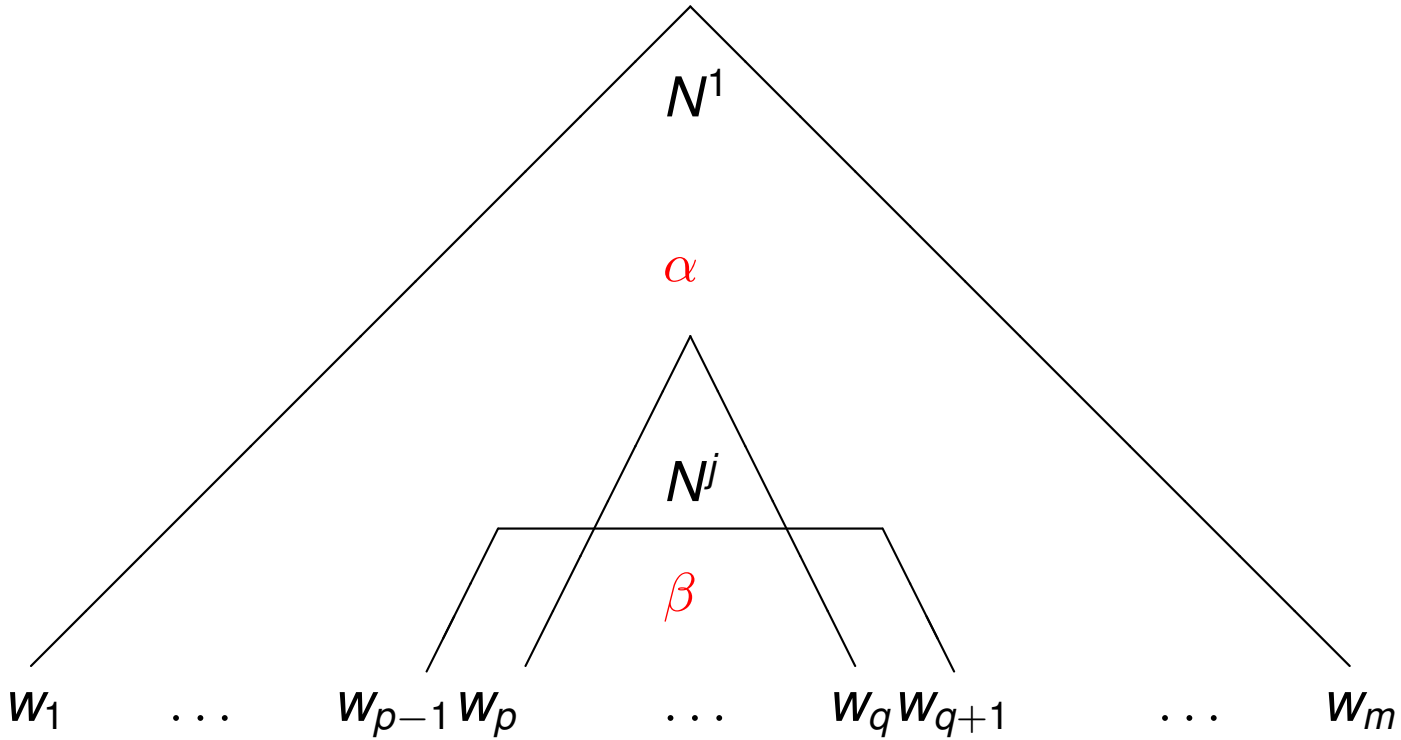
Definition

- Outside probability: $\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)$



Definition

- Inside probability: $\beta_j(p,q) = P(w_{pq}|N^j_{pq}, G)$
- Outside probability: $\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)$



Inside Algorithm

- Dynamic programming algorithm based on inside probabilities:

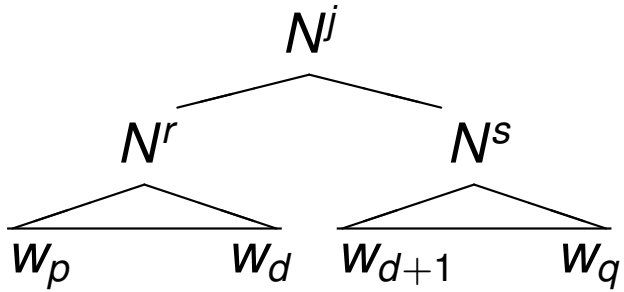
$$P(w_{1m}|G) = P(w_{1m}|N^1_{1m}, G) = \beta_1(1, m)$$

- Calculates the inside probabilities recursively, bottom up.

1. Base case:

$$\beta_j(k, k) = P(w_k|N^j_{kk}, G) = P(N^j \to w_k|G)$$

2. Induction:



$$\begin{aligned}\beta_j(p, q) &= P(w_{pq}|N^j_{pq}, G) \\ &= \sum_{r,s}\sum_{d=p}^{q-1} P(N^j \to N^r N^s)\beta_r(p, d)\beta_s(d + 1, q)\end{aligned}$$

Inside Algorithm – 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

	1	2	3	4	5
1					
2					
3					
4					
5					
	astronomers	*saw*	*stars*	*with*	*ears*

Inside Algorithm – Example

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

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2					
3					
4					
5					
	astronomers	*saw*	*stars*	*with*	*ears*

Inside Algorithm – Example

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

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_{NP} = 0.04$ $\beta_{V} = 1.0$			
3					
4					
5					
	astronomers	*saw*	*stars*	*with*	*ears*

Inside Algorithm – Example

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

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_{NP} = 0.04$ $\beta_{V} = 1.0$			
3			$\beta_{NP} = 0.18$		
4					
5					
	astronomers	*saw*	*stars*	*with*	*ears*

Inside Algorithm – Example

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

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_{NP} = 0.04$ $\beta_{V} = 1.0$			
3			$\beta_{NP} = 0.18$		
4				$\beta_{P} = 1.0$	
5					
	astronomers	*saw*	*stars*	*with*	*ears*

Inside Algorithm – Example

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

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_{NP} = 0.04$ $\beta_{V} = 1.0$			
3			$\beta_{NP} = 0.18$		
4				$\beta_{P} = 1.0$	
5					$\beta_{NP} = 0.18$
	astronomers	*saw*	*stars*	*with*	*ears*

Inside Algorithm – Example

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

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_{NP} = 0.04$ $\beta_{V} = 1.0$	$\beta_{VP} = 0.126$		
3			$\beta_{NP} = 0.18$		
4				$\beta_{P} = 1.0$	
5					$\beta_{NP} = 0.18$
	astronomers	*saw*	*stars*	*with*	*ears*

Inside Algorithm – Example

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

	1	2	3	4	5
1	$\beta_{NP} = 0.1$				
2		$\beta_{NP} = 0.04$ $\beta_{V} = 1.0$	$\beta_{VP} = 0.126$		
3			$\beta_{NP} = 0.18$		
4				$\beta_{P} = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	astronomers	*saw*	*stars*	*with*	*ears*

Inside Algorithm – 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

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

Inside Algorithm – Example

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

	1	2	3	4	5
1	$\beta_{NP} = 0.1$		$\beta_S = 0.0126$		
2		$\beta_{NP} = 0.04$ $\beta_V = 1.0$	$\beta_{VP} = 0.126$		
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*

Inside Algorithm – 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

	1	2	3	4	5
1	$\beta_{NP} = 0.1$		$\beta_S = 0.0126$		
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*

Inside Algorithm – Example

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

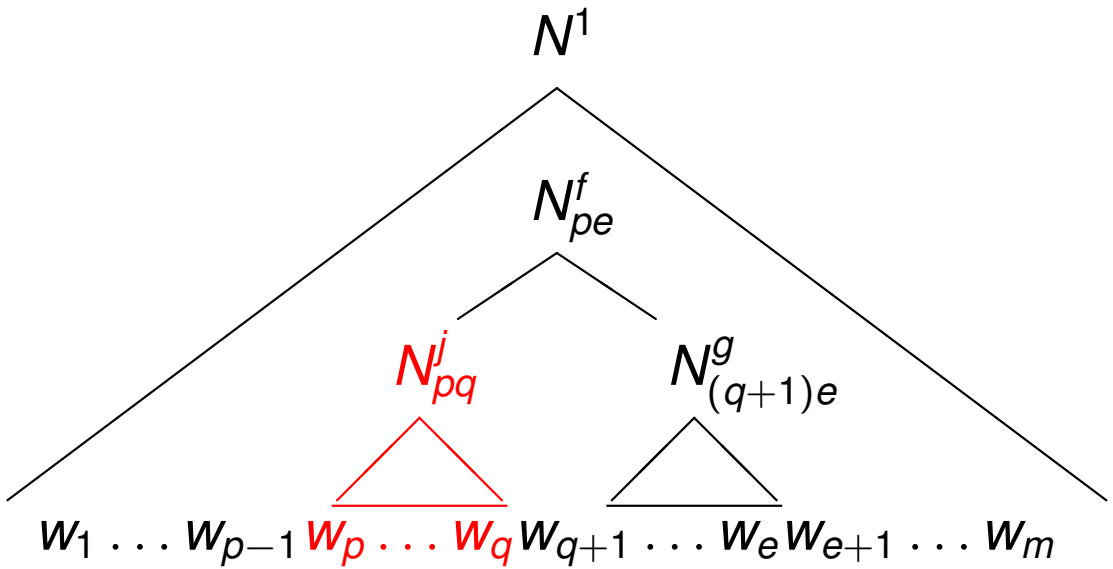
	1	2	3	4	5
1	$\beta_{NP} = 0.1$		$\beta_S = 0.0126$		$\beta_S = 0.0015876$
2		$\beta_{NP} = 0.04$ $\beta_V = 1.0$	$\beta_{VP} = 0.126$		$\beta_{VP} = 0.015876$
3			$\beta_{NP} = 0.18$		$\beta_{NP} = 0.01296$
4				$\beta_P = 1.0$	$\beta_{PP} = 0.18$
5					$\beta_{NP} = 0.18$
	astronomers	*saw*	*stars*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

- Dynamic programming algorithm based on outside probabilities:

$$
\begin{aligned}
P(w_{1m}|G) &= \sum_j P(w_{1(k-1)}, w_k, w_{(k+1)m}, N^j_{kk}|G) \\
&= \sum_j P(w_{1(k-1)}, N^j_{kk}, w_{(k+1)m}|G) \\
&\qquad \times P(w_k|w_{1(k-1)}, N^j_{kk}, w_{(k+1)m}, G) \\
&= \sum_j \alpha_j(k,k) P(N^j \rightarrow w_k)
\end{aligned}
$$

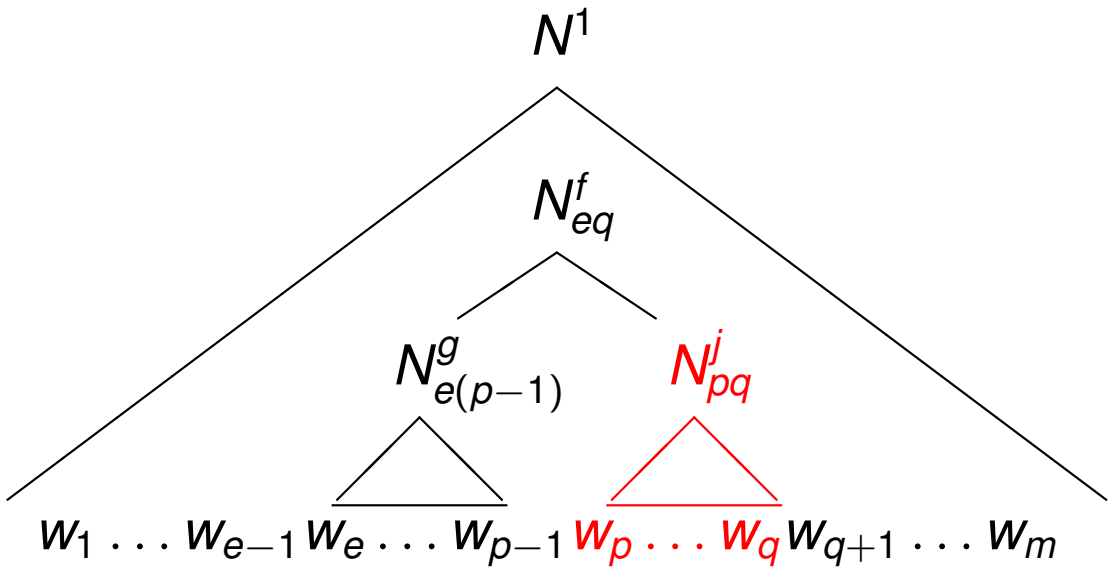
  for any k such that $1 \leq k \leq m$.
- Calculates the outside probabilities recursively, top down.
- Requires reference to inside probabilities.

Outside Algorithm – Case 1



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

Outside Algorithm – Case 2



$$\alpha_j(p,q) = \sum_{f,g}\sum_{e=1}^{p-1} \alpha_f(e,q) P(N^f \rightarrow N^g N^j) \beta_g(e,p-1)$$

Outside Algorithm

1. Base case:

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

2. Induction:

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

Sentence Probability – Summary

- Using inside probabilities:

$$P(w_{1m}|G) = \beta_1(1, m)$$

- Using outside probabilities:

$$P(w_{1m}|G) = \sum_j \alpha_j(k, k) P(N^j \rightarrow w_k)$$

  for any k such that $1 \leq k \leq m$.

- Probability of a sentence w_{1m} and that there is some constituent spanning from word p to q:

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

Finding the Most Likely Parse

- Modication of the inside algorithm:
  - Find the maximum element of the sum in each step.
  - Record which rule gave this maximum.

- We can define accumulators (similar to Viterbi algorithm for HMM):

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

Finding the Most Likely Parse

1. Base case:

$$\delta_i(p,p) = P(N^i \to w_p)$$

2. Induction:

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

Backtrace:

$$\psi_i(p,q) = \arg\max_{(j,k,r)} P(N^i \to N^j N^k)\delta_j(p,r)\delta_k(r+1,q)$$

3. Termination:

$$P(\hat{t}) = \delta_1(1,m)$$

We need to reconstruct the parse tree $\hat{t}$.

- **Probabilistic Context-Free Grammar**
  - Definition and examples
  - Properties and usage
- **Inside and Outside Probabilities**
  - Definitions
  - Algorithms
- **Inside-Outside Algorithm**
  - Idea, formal description and properties

Training a PCFG

- Assume a certain topology of the grammar G given in advance.
  - Number of terminals and nonterminals.
  - Name of the start symbol.
  - Set of rules (we can have a given structure of the grammar, but we can also assume all possible rewriting rules exist).
- We want to set the probabilities of rules to maximize the likelihood of the training data.

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

  where $C(x)$ is the number of times the rule x is used.
- Trivial if we have a parsed corpus for training.

Inside-Outside Algorithm

- Usually, a parsed training corpus is not available.
- Hidden data problem - we can only directly see the probabilities of sentences, not rules.
- We can use an iterative algorithm to determine improving estimates - the inside-outside algorithm.

Idea

1. Begin with a given grammar topology and some initial probability estimates for rules.
2. The probability of each parse of a training sentence according to G will act as our confidence in it.
3. Sum the probabilities of each rule being used in each place to give an expectation of how often each rule was used.
4. Use the expectations to refine the probability estimates - increase the likelihood of the traning corpus according to G.

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

- To estimate the count of times the nonterminal N^j is used in the derivation:

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

- If N^j is not a preterminal, we can substitute the inductive definition of β. Then, $\forall r, s, p, q$:

$$P(N^j_{pq}|w_{1m}, G) = \frac{\sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)}{P(w_{1m}|G)}$$

- To estimate the number of times this rule is used in the derivation:

$$E(N^j \rightarrow N^r N^s, N^j \text{ used})$$

$$= \frac{\sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)}{P(w_{1m}|G)} \quad (2)$$

- For the maximization step, we want:

$$\hat{P}(N^j \rightarrow N^r N^s) = \frac{E(N^j \rightarrow N^r N^s, N^j \text{ used})}{E(N^j \text{ used})}$$

- Reestimation formula:

$$\hat{P}(N^j \rightarrow N^r N^s) = (1)/(2)$$

$$= \frac{\sum_{p=1}^{m-1} \sum_{q=p+1}^{m} \sum_{d=p}^{q-1} \alpha_j(p,q) P(N^j \rightarrow N^r N^s) \beta_r(p,d) \beta_s(d+1,q)}{\sum_{p=1}^{m} \sum_{q=p}^{m} \alpha_j(p,q) \beta_j(p,q)} \quad (3)$$

- Analogically for preterminals, we get:

$$\hat{P}(N^j \rightarrow w^k) = \frac{\sum_{h=1}^{m} \alpha_j(h,h) P(w_h = w^k) \beta_j(h,h)}{\sum_{p=1}^{m} \sum_{q=p}^{m} \alpha_j(p,q) \beta_j(p,q)} \quad (4)$$

Inside-Outside Algorithm

Method

1. Initialize probabilities of rules in G.
2. Calculate inside probabilities for the training sentence.
3. Calculate outside probabilities for the training sentence.
4. Update the rule probabilities using reestimation formulas (3) and (4).
5. Repeat from step 2 until the change in estimated rule probabilities is sufficiently small.

- The probability of the training corpus according to G will improve (or at least not get worse):

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

where i is the current iteration of training.

Inside-Outside Algorithm – Properties

- Time complexity:
  For each sentence, each iteration of training is $O(m^3 n^3)$, where m is the length of the sentence and n is the number of nonterminals in the grammar.
  - Relatively slow compared to linear models (such as HMM).

- Problems with local maxima, higly sensitive to the initialization of parameters.

- Generally, we cannot guarantee any resemblance between the trained grammar and the kinds of structures commonly used in NLP (NP, VP, etc.). The only hard constraint is that N^1 remains the start symbol.
  - We could impose further constraints.

References

Christopher D. Manning, Hinrich Schütze:
Foundations of Statistical Natural Language Processing,
MIT Press, 1999

Fei Xia:
Inside-ouside algorithm (presentation),
University of Washington, 2006
`http://faculty.washington.edu/fxia/courses/LING572/inside-outside.ppt`

Thank you for your attention!

End