Stochastic Context Free Languages

Bryan Jurish

jurish@ling.uni-potsdam.de

Universität Potsdam, Institut für Linguistik,

Potsdam, Germany

2005-06-30

Outline

The Basics

- Definitions
- Assumptions
- Language Model

Efficient String Probability Computation

- Inside Algorithm
- Outside Algorithm

Parameter Reestimation

- Inside-Outside Algorithm
- Commentary and Caveats

Basics

SCFG: Definition

A Stochstic Context-Free Grammar (SCFG or PCFG) is a 5-tuple $G = \langle V, N, R, S, P \rangle$, where:

- $\langle V, N, R, S \rangle$ is a context-free grammar over the terminal alphabet $\mathcal{A} = V - N$ in Chomsky normal form.
- $P : R \rightarrow [0, 1]$ is a probability distribution on the rules of R conditioned on their left-hand-sides:

$$\forall A \in N : \sum_{\alpha \in V*} P(A \rightarrow \alpha | A) = 1$$

- We write $P(\alpha|A)$ or $P(A \rightarrow \alpha)$ for $P(A \rightarrow \alpha|A)$ where convenient

SCFG: Assumptions

Given an SCFG G, the following are assumed to hold for all strings $w = w_{1..n} \in \mathcal{A}^*$, substrings $\zeta \in \mathcal{A}^*$, nonterminals $A, B \in N$, and string position indices $0 \leq k, \ell \leq n$:

- Place Invariance

$$P(A \lhd^*_{k,\ell} \zeta | A) = P(A \lhd^*_{1,\ell-k+1} \zeta | A)$$

- Context Freedom

$$P(A \lhd^*_{k,\ell} \zeta | w_{1..k-1}, A, w_{\ell+1..n}) = P(A \lhd^*_{k,\ell} \zeta | A)$$

- Ancestor Freedom

$$P(A \lhd^*_{k,\ell} \zeta | B \lhd^+_{k-i,\ell+j} \nu\zeta\eta, A) = P(A \lhd^*_{k,\ell} \zeta | A)$$

SCFG: Probability

Tree Probability

The probability $P_G(t) = P(t|G)$ of generating of a tree $t \in T(V \times \mathbb{N})$ with an SCFG G is defined for $A \in N, a \in \mathcal{A}, t_1, t_2 \in T(V \times \mathbb{N})$:

$$\begin{aligned} P(\langle A, 1\rangle(a)|G) &= P(a|A) \\ P(\langle A, 2\rangle(t_1, t_2)|G) &= P(\mathrm{root}(t_1)\mathrm{root}(t_2)|A)P(t_1)P(t_2) \end{aligned}$$

String Probability

The probability $P_G(w) = P(w|G)$ of generating a string $w \in \mathcal{A}^*$ with an SCFG G is:

$$P(w|G) \quad \sum_{t \in T(V \times \mathbb{N}) : \mathrm{root}(t) = S \,\&\, \mathrm{yield}(t) = w} \quad P(t|G)$$

SCFG: Language Model

- The stochastic string language generated by an SCFG G is the probability distribution $P(\cdot|G) : \mathcal{A}^* \to [0, 1]$ over terminal strings:

$$\sum_{w \in \mathcal{A}^*} P(w|G) \leq 1$$

  NOTE: G may not define a string language model in the technical sense, as some probability mass may be lost on “useless” trees.

- The symbolic or categorical language $\mathcal{L}(G)$ generated by G is the set of strings generated by G with nonzero probability:

$$\mathcal{L}(G) = \{w \mid P(w|G) \neq 0\}$$

String Probability Computation

String Probability

The Problem

- Efficiently compute $P(w|G)$ for given w and G
- Definition: $P(w|G) = \sum_{t \in \mathrm{yield}^{-1}(w)} P(t|G)$
- Bummer: $|\mathrm{yield}^{-1}(w)| = O(c^{|w|})$

The Solution

- Dynamic programming approach
- Akin to forward-, backward-algorithms for HMMs
- Inside probabilities: $\alpha_{i,j}(A)$
- Outside probabilities: $\beta_{i,j}(A)$

Inside & Outside Probabilities

Inside Probability

$$\beta_{i,j}(A) = P(w_{i..j} | A \lhd^*_{i,j} w, G)$$

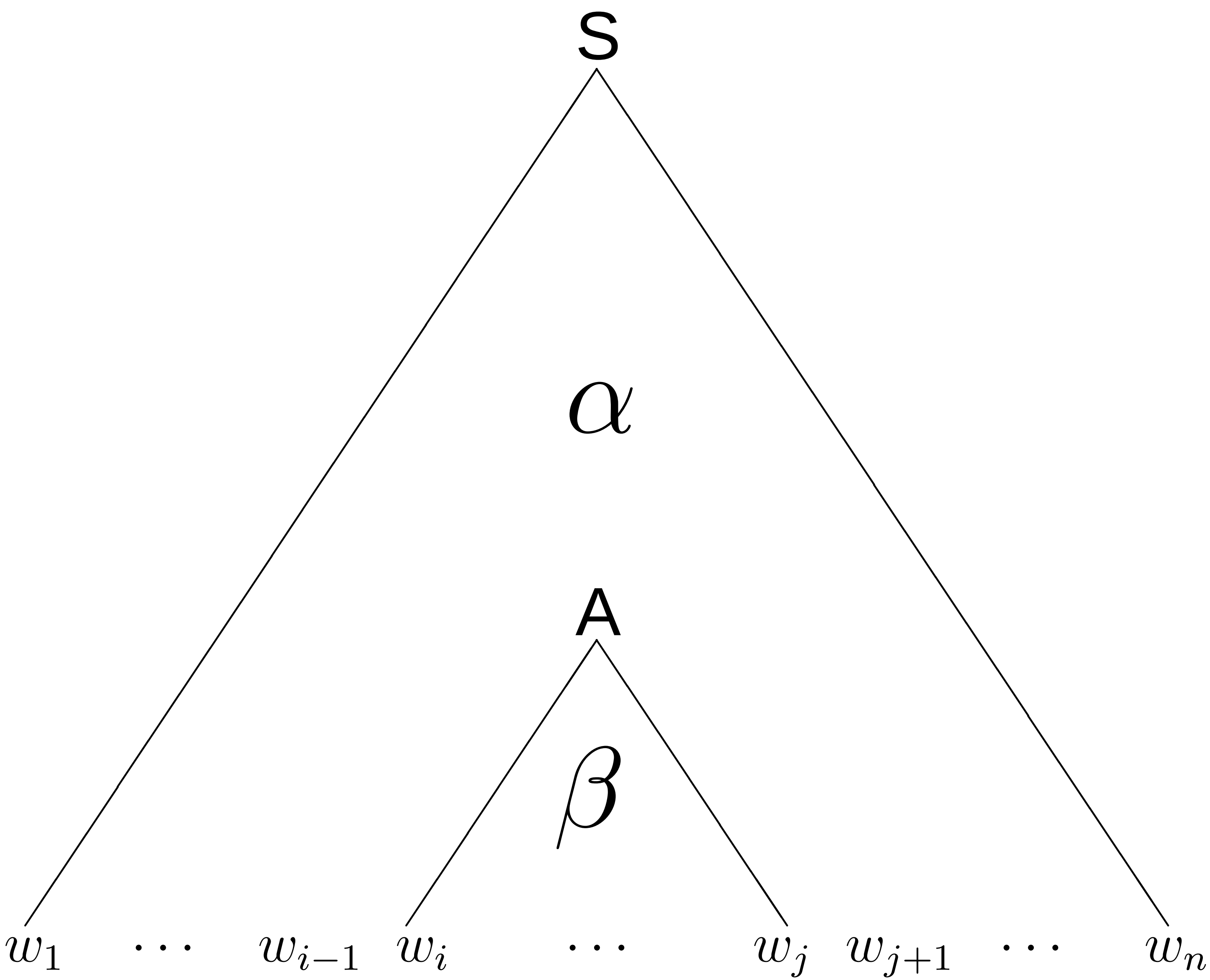
- Total probability of generating substring $w_{1..j}$ as the yield of a subtree rooted at A.

Outside Probability

$$\alpha_{i,j}(A) = P(w_{1..i-1}, A \lhd^*_{i,j} w, w_{j+1..n} | G)$$

- Total probability of generating a partial derivation lacking only the substructure $A \lhd^*_{i,j} w$ for completion.

Inside & Outside



Inside Algorithm: Basis

Notation

Let $A \lhd^*_{i,j} w$ be abbreviated ${}_iA_j$

The Approach

$$\begin{aligned} P(w_{1..n}|G) &= P(S \Rightarrow^* w_{1..n}|G) \\ &= P(w_{1..n}|{}_1S_n, G) \\ &= \beta_{1,n}(S) \end{aligned}$$

Basis: Lexical Insertion $(i = j)$

$$\begin{aligned} \beta_{i,i}(A) &= P(w_i|{}_iA_i, G) \\ &= P(A \rightarrow w_i|G) \end{aligned}$$

Inside Algorithm: Induction

$$\begin{aligned}
\beta_{i,j}(A) &= P(w_{i..j} | {}_iA_j, G) \\
[Marg] &= \textstyle\sum_{B,C} \sum_{k=i}^{j-1} P(w_{i..k}, {}_iB_k, w_{k+1..j}, {}_{k+1}C_j | {}_iA_j, G) \\
[Chain] &= \textstyle\sum_{B,C} \sum_{k=i}^{j-1} P({}_iB_k, {}_{k+1}C_j | {}_iA_j, G) \\
&\qquad \times P(w_{i..k} | {}_iA_j, {}_iB_k, {}_{k+1}C_j, G) \\
&\qquad \times P(w_{k+1..j} | w_{i..k}, {}_iA_j, {}_iB_k, {}_{k+1}C_j, G) \\
[Indp] &= \textstyle\sum_{B,C} \sum_{k=i}^{j-1} P({}_iB_k, {}_{k+1}C_j | {}_iA_j, G) \\
&\qquad \times P(w_{i..k} | {}_iB_k, G) \\
&\qquad \times P(w_{k+1..j} | {}_{k+1}C_j, G) \\
[Defn] &= \textstyle\sum_{B,C} \sum_{k=i}^{j-1} P(A \to B\ C) \beta_{i,k}(B) \beta_{k+1,j}(C)
\end{aligned}$$

Outside Algorithm: Basis

The Approach: $\forall k, 1 \leq k \leq n$,

$$\begin{aligned}
P(w_{1..n}|G) &= \sum_A P(w_{1..k-1}, w_k, w_{k+1..n}, {}_kA_k|G) \\
&= \textstyle\sum_A P(w_{1..k-1}, {}_kA_k, w_{k+1..n}, |G) \\
&\qquad \times P(w_k|w_{1..k-1}, {}_kA_k, w_{k+1..n}, G) \\
&= \sum_A \alpha_{k,k}(A) P(A \rightarrow w_k)
\end{aligned}$$

Basis: Root Node

$$\alpha_{1,n}(A) = \begin{cases} 1 & \text{if } A = S \\ 0 & \text{otherwise} \end{cases}$$

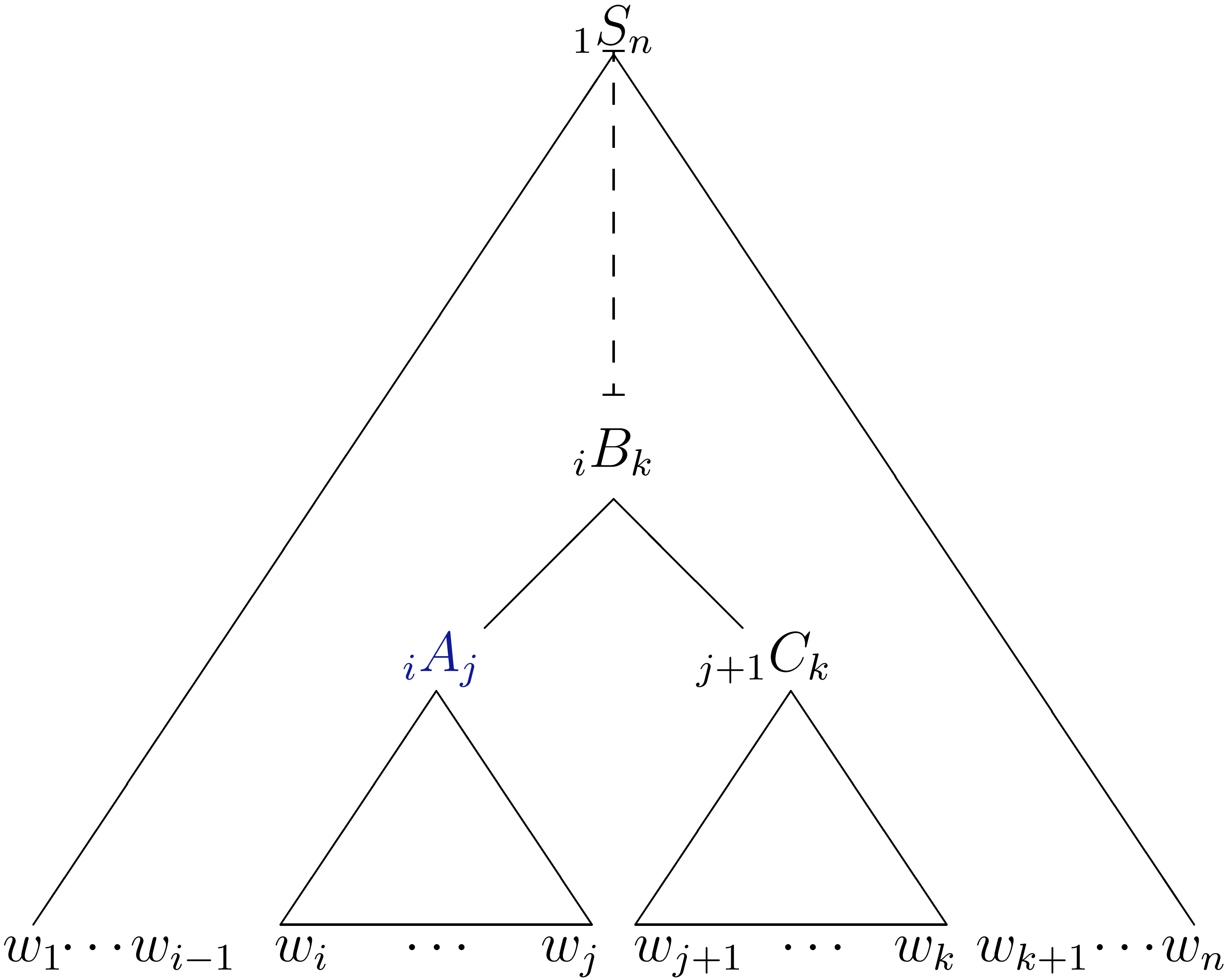
Outside Algorithm: Induction

$$\alpha_{i,j}(A) = \left[\sum_{B,C\neq A}\sum_{k=j+1}^{n} P(w_{1..i-1}, w_{j+1..n}, {}_iB_k, {}_iA_j, {}_{j+1}C_k)\right] + \left[\sum_{B,C}\sum_{k=1}^{i-1} P(w_{1..i-1}, w_{j+1..n}, {}_kB_j, {}_kC_{i-1}, {}_iA_j)\right]$$

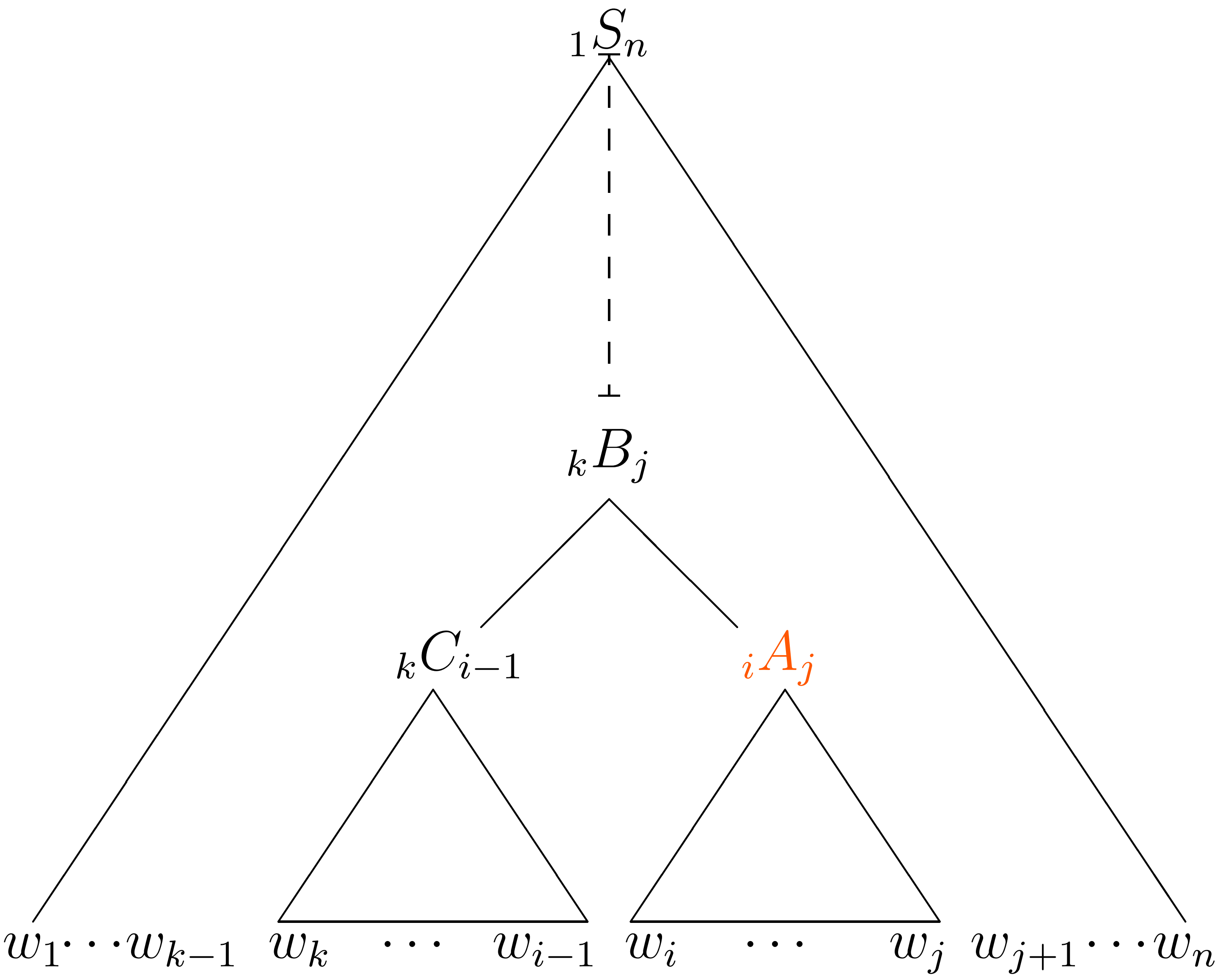
$$= \left[\sum_{B,C\neq A}\sum_{k=j+1}^{n} P(w_{1..i-1}, w_{k+1..n}, {}_iB_k) \times P({}_iA_j, {}_{j+1}C_k | {}_iB_k P(w_{j+1}k|_{j+1}C_k)\right] + \left[\sum_{B,C}\sum_{k=1}^{i-1} P(w_{1..k-1}, w_{j+1..n}, {}_kB_j) \times P({}_kC_{i-1}, {}_iA_j | {}_kB_j) P(w_{k..i-1} | {}_kC_{i-1})\right]$$

$$= \left[\sum_{B,C\neq A}\sum_{k=j+1}^{n} \alpha_{i,k}(B) P(B \to A\ C) \beta_{j+1,k}(C)\right] + \left[\sum_{B,C}\sum_{k=1}^{i-1} \alpha_{k,j}(B) P(B \to C\ A) \beta_{k,i-1}(C)\right]$$

Outside: Left Daughter



Outside: Right Daughter



IO Probabilities Combined

Inside-Outside Product

- As for HMMs, $\alpha\beta$ gives us the joint probability of the string and a nonterminal node:

$$\begin{aligned}\alpha_{i,j}(A)\beta_{i,j}(A) &= P(w_{1..i-1}, {}_iA_j, w_{j+1..n}|G)P(w_{i..j}|{}_iA_j, G) \\ &= P(w_{1..n}, {}_iA_j|G)\end{aligned}$$

Sum of Products

- Summing over $\alpha\beta$ products is tricky, since we must consider non-constituent substrings:

$$P(w_{1..n}, \exists A : {}_iA_j|G) = \sum_A \alpha_{i,j}(A)\beta_{i,j}(A)$$

Paramter Reestimation

Parameter Reestimation

Idea: Expectation Maximization

- Unsupervised parameter reestimation for SCFGs
- Analagous to Baum-Welch algorithm for HMMs
- Expect: compute sentence probability
- Maximize: adjust SCFG parameters

Specification

- Input:
  - SCFG $G = \langle V, N, R, S, P \rangle$
  - Training sentence w (extension: corpus)
- Output:
  - Reestimated SCFG $\hat{G} = \langle V, N, R, S, \hat{P} \rangle$ such that $P(w|\hat{G}) \geq P(w|G)$

EM: The Plan

Excursus: the supervised case

- Given a treebank, an ML estimator for P is:

$$P_{ML}(A \to \zeta) = \frac{f(A \lhd \zeta)}{f(A)} = \frac{f(A \lhd \zeta)}{\sum_\gamma f(A \lhd \gamma)}$$

- All we need to do is estimate $f(A \lhd \zeta)$ for all $A \in N, \zeta \in \mathcal{A} \cup N^2$
- Given an initial grammar, we can use α, β to compute expected frequencies $\ldots$

EM: Nonterminal Frequency

Recall:

$$\begin{aligned}\alpha_{i,j}(A)\beta_{i,j}(A) &= P(S \Rightarrow^* w_{1..n}, A \Rightarrow^* w_{i..j} | G) \\ &= P(S \Rightarrow^* w_{1..n} | G) P(A \Rightarrow^* w_{i..j} | S \Rightarrow^* w_{1..n}, G)\end{aligned}$$

therefore,

$$P(A \Rightarrow^* w_{i..j} | S \Rightarrow^* w_{1..n}, G) = \frac{\alpha_{i,j}(A)\beta_{i,j}(A)}{P(S \Rightarrow^* w_{1..n} | G)}$$

so we can estimate:

$$E(A) = \sum_{i=1}^{n} \sum_{j=i}^{n} \frac{\alpha_{i,j}(A)\beta_{i,j}(A)}{P(S \Rightarrow^* w_{1..n} | G)}$$

EM: Internal Branches

Note that $\forall B, C, i, j$:

$$P(A \Rightarrow B\ C \Rightarrow^* w_{i..j} | S \Rightarrow^* w_{1..n}, G)$$

$$= \frac{\sum_{k=i}^{j-1} \alpha_{i,j}(A) P(A \to B\ C) \beta_{i,k}(B) \beta_{k+1,j}(C)}{P(S \Rightarrow^* w_{1..n} | G)}$$

summing, we get:

$$E(A \to B\ C, A)$$

$$= \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=i}^{j-1} \alpha_{i,j}(A) P(A \to B\ C) \beta_{i,k}(B) \beta_{k+1,j}(C)}{P(S \Rightarrow^* w_{1..n} | G)}$$

so that we wind up maximizing with:

$$\hat{P}(A \to B\ C | A) = \frac{E(A \to B\ C, A)}{E(A)}$$

$$= \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sum_{k=i}^{j-1} \alpha_{i,j}(A) P(A \to B\ C) \beta_{i,k}(B) \beta_{k+1,j}(C)}{\sum_{i=1}^{n} \sum_{j=i}^{n} \alpha_{i,j}(A) \beta_{i,j}(A)}$$

EM: Preterminals

Note also:

$$
\begin{aligned}
P(A \Rightarrow w_k | S \Rightarrow^* w_{1..n}, G) &= \frac{\sum_{i=1}^{n} \alpha_{i,i}(A) P(A \rightarrow w_i, w_i = w_k)}{P(S \Rightarrow^* w_{1..n} | G)} \\
&= \frac{\sum_{i=1}^{n} \alpha_{i,i}(A) P(w_i = w_k) \beta_{i,i}(A)}{P(S \Rightarrow^* w_{1..n} | G)} \\
&= E(A \rightarrow w_k, A)
\end{aligned}
$$

so that we can maximize with:

$$
\begin{aligned}
\hat{P}(A \rightarrow w_k | A) &= \frac{E(A \rightarrow w_k, A)}{E(A)} \\
&= \frac{\sum_{i=1}^{n} \alpha_{i,i}(A) P(w_i = w_k) \beta_{i,i}(A)}{\sum_{i=1}^{n} \sum_{j=i}^{n} \alpha_{i,j}(A) \beta_{i,j}(A)}
\end{aligned}
$$

Caveat Programmor

- EM is dog slow: $O = O(n^3|N|^3)$
- Many nonterminals required for accurate learning (did I mention it was dog slow?)
- Local maxima problem: huge parameter space
  - *a priori* restrictions can help here
- Doubtful linguistic utility:
  - "What the heck is a `N29`?"
  - SCFGs **really** like small trees
  - No account of non-local precedence phenomena
    - Using an HMM for the lexical insertion stage helps here, and reduces alphabet size to boot

The End