inside-outside

February 28, 2023

1 Inside-Outside Algorithm

1.1 Background

The seminal paper on the inside-outside algorithm is (Baker). However, (Lari and Young) claim that Baker neither specified the actual algorithm nor gave an application (I haven't yet found my copy of (Baker) to verify).

My re-derivation here of the inside-outside algorithm (from first principles) is mostly based on (Lari and Young). However, they normalise their rule probabilities differently (as discussed in a later section).

1.1.1 The problem

Consider a stochastic process that outputs an arbitrary-length sequence $\mathbf{Y}$ of discrete symbols, where $\mathbf{Y}$ may be considered as a vector random variable. In the language domain, each symbol might be a character (e.g. alphanumeric, space or punctuation), or a word with the sequence of words forming a sentence (with the inclusion of spacing and punctuation in natural language). In general, we let each symbol be a token from a finite set $\mathcal{Y} = \{\nu_1, \nu_2, \ldots\}$ of discrete tokens.

Thus, we suppose that we have observed an entire sequence $\mathbf{y} = (y_1, y_2, \ldots)$ of such tokens. Note that once we know that $|\mathbf{y}| = T$, say, then $\mathbf{y} \in \mathcal{Y}^T$, and we may consider either the complete sequence $\mathbf{y} = \mathbf{y}_{1:T} \doteq (y_1, y_2, \ldots, y_T)$, or any contiguous subsequence $\mathbf{y}_{s:t} \doteq (y_s, y_{s+1}, \ldots, y_t)$ of tokens. Given $\mathbf{y}$, there are two main problems of interest.

One problem is to construct the (or a) most probable parse structure $\mathbf{s}$ that best explains this sequence, given a grammar $\mathcal{G}$ of plausible substructures or rules. Thus, if we let $\mathbf{S}$ be a vector random variable representing each possible parse structure $\mathbf{s}$, then parsing seeks $\mathbf{s}$ to maximise $P(\mathbf{S} = \mathbf{s} \mid \mathbf{Y} = \mathbf{y})$.

The other problem of interest is to induce the grammar $\mathcal{G}$ directly from a corpus $\mathbb{Y} = (\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots)$ of observed sequences, using a generative model of the form $P(\mathbf{S} = \mathbf{s}, \mathbf{Y} = \mathbf{y})$. This is the purpose of the *inside-outside* algorithm - to automatically extract the various rules and their probabilities for a stochastic, context-free grammar in Chomsky normal form. Consequently, only unary rules may produce tokens, and unary rules must produce only tokens. This means that unary *type-raising* rules are forbidden (which also simplifies parsing).

1.1.2 Rule conditioning

We assume that all rules are conditioned on the antecedent, and that summing over all possible consequents results in a value of unity. Thus, a stochastic unary rule of the form `A` $\rightarrow$ `B` has

probability $P(\mathtt{B} \mid \mathtt{A})$, such that $\sum_b P(b \mid \mathtt{A}) = 1$. Likewise, a top-down binary rule of the form $\mathtt{A} \to \mathtt{B} \oplus \mathtt{C}$ has probability $P(\mathtt{B}, \mathtt{C} \mid \mathtt{A})$, such that $\sum_{b,c} P(b, c \mid \mathtt{A}) = 1$. Conversely, the bottom-up (reversed) binary rule $\mathtt{B} \oplus \mathtt{C} \to \mathtt{A}$ has probability $P(\mathtt{A} \mid \mathtt{B}, \mathtt{C})$, such that $\sum_a P(a \mid \mathtt{B} \oplus \mathtt{C}) = 1$.

In contrast, (Lari and Young) normalise their generative rules to satisfy $\sum_d P(A \to d) + \sum_{b,c} P(A \to b \oplus c) = 1$. This makes sense in a generative process **prior** to observation, since once the process has entered an antecedent state $\mathtt{A}$, any of its consequents (unary token or binary states) could potentially be generated.

However, the situation changes **after** we have observed sequence $\mathbf{y}$, since we now have more information. For example, if the grammar $\mathcal{G}$ always starts with a single, unique root state $\mathtt{S}$ and we know that $|\mathbf{y}| > 1$, then it is inconsistent to consider any unary rules $\mathtt{S} \to \nu_m$. In other words, for observed sequence $\mathbf{y}$ and potential parse structure $\mathbf{s}$, we must have both $P(\mathbf{S} = \mathbf{s}, \mathbf{Y} = \mathbf{y}) = 0$ and $P(\mathbf{S} = \mathbf{s} \mid \mathbf{Y} = \mathbf{y}) = 0$ whenever $\mathbf{s}$ is inconsistent with $\mathbf{y}$.

Put another way, the assumption of Chomsky normal form means that each token must be spanned by a unary rule, and every (contiguous) subsequence of two or more tokens must be spanned by a binary rule. Since the decision of a unary or binary rule is deterministic given the span $s : t$, it no longer makes sense to ignore this conditioning information. Hence, we make unary and binary rules separately sum to unity over the consequents given the antecedent, instead of making them collectively sum to unity like (Lari and Young).

In practice, either normalisation scheme will work the same once we condition on $\mathbf{y}$, e.g. $P(\mathbf{S} = \mathbf{s} \mid \mathbf{Y} = \mathbf{y})$ and its various marginalisations over components of $\mathbf{S}$. However, the choice affects the viability of parsing.

1.1.3 Comparison with a HMM

Before proceeding to the details of the inside-outside algorithm, let us first briefly summarise some basic details about modelling a sequence with a hidden Markov model (HMM). Consider a simple Markov chain transitioning from stage t to stage $t + 1$, for $t = 1, 2, \ldots, \infty$, such as that shown in the figure below for the sentence "*The cat sat on the mat.*".

Let the hidden state at stage t be represented by the variable S_t, taking one of a finite set of possible values $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots\}$. For the simple HMM shown above, the allowable transitions between stages t and $t + 1$ have probabilities $P(S_{t+1} = \sigma_j \mid S_t = \sigma_i)$, for $\sigma_i, \sigma_j \in \mathcal{S}$. These transition probabilities are typically assumed to be invariant with respect to the stages, and are usually represented by the matrix $\mathbf{A} = [a_{ij}]$, such that $a_{ij} \doteq P(S_{t+1} = \sigma_j \mid S_t = \sigma_i)$ and $\sum_{j=1}^{|\mathcal{S}|} a_{ij} = 1$. Hence, the stage-invariant transitions correspond to context-free rules of the form $\sigma_i \to \sigma_j$.

Similarly, let the output at stage t be represented by the variable $Y_t \in \mathcal{Y}$. This output is generated by the HMM process via the state S_t with probability $P(Y_t = \nu_m \mid S_t = \sigma_i)$ for $\sigma_i \in \mathcal{S}$ and $\nu_m \in \mathcal{Y}$. Once again, these *emission* probabilities are assumed to be invariant with respect to the stage, and are usually represented by the matrix $\mathbf{B} = [b_{im}]$, such that $b_{im} \doteq P(Y_t = \nu_m \mid S_t = \sigma_i)$ and $\sum_{m=1}^{|\mathcal{Y}|} b_{im} = 1$. Hence, the stage-invariant emissions correspond to context-free rules of the form $\sigma_i \to \nu_m$.

Given these transition and emission probabilities, the HMM is a generative model that specifies the joint probabilities $P(\mathbf{S} = \mathbf{s}, \mathbf{Y} = \mathbf{y})$. Since the hidden states $\mathbf{s}$ are unknown in general, the key

quantity is the likelihood of the observed sequence, namely

$$P(\mathbf{Y} = \mathbf{y}) \quad = \quad \sum_{\mathbf{s} \in \mathcal{S}^{|\mathbf{y}|}} P(\mathbf{S} = \mathbf{s}, \mathbf{Y} = \mathbf{y}) \,. \tag{1}$$

An efficient method for calculating this probability is offered by the *forward-backward* algorithm. The forward pass along the sequence $\mathbf{y}_{1:T}$ recursively computes joint probabilities of the form

$$\alpha_t(i) \quad \doteq \quad P(\mathbf{Y}_{1:t} = \mathbf{y}_{1:t}, S_t = \sigma_i) \,, \tag{2}$$

for $t = 1, 2, \ldots, T$. Conversely, the backward pass recursively computes conditional probabilities of the form

$$\beta_t(i) \quad \doteq \quad P(\mathbf{Y}_{t+1:T} = \mathbf{y}_{t+1:T} \mid S_t = \sigma_i) \,, \tag{3}$$

for $t = T, T-1, \ldots, 1$, where $\mathbf{y}_{t:s} = ()$ for $t > s$ by definition.

The required Markov property is that the past (i.e. $\mathbf{y}_{1:t}$) is conditionally independent of the future (i.e. $\mathbf{y}_{t+1:T}$) given the present (i.e. $S_t = \sigma_i$). It follows that the likelihood of the observed sequence $\mathbf{y}_{1:T}$ is

$$\begin{aligned}
P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}) \quad &= \quad \sum_{i=1}^{|\mathcal{S}|} P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_t = \sigma_i) && (4) \\
&= \quad \sum_{i=1}^{|\mathcal{S}|} P(\mathbf{Y}_{1:t} = \mathbf{y}_{1:t}, S_t = \sigma_i)\, P(\mathbf{Y}_{t+1:T} = \mathbf{y}_{t+1:T} \mid S_t = \sigma_i) && (5) \\
&= \quad \sum_{i=1}^{|\mathcal{S}|} \alpha_t(i)\, \beta_t(i) \,, && (6)
\end{aligned}$$

for any and every $t = 1, 2, \ldots, T$.

The standard forward formulation computes $\alpha_t(i)$ via the recursion

$$\begin{aligned}
\alpha_t(i) \quad &= \quad P(\mathbf{Y}_{1:t} = \mathbf{y}_{1:t}, S_t = \sigma_i) && (7) \\
&= \quad \sum_{j=1}^{|\mathcal{S}|} P(\mathbf{Y}_{1:t-1} = \mathbf{y}_{1:t-1}, S_{t-1} = \sigma_j)\, P(S_t = \sigma_i \mid S_{t-1} = \sigma_j)\, P(Y_t = y_t \mid S_t = \sigma_i) && (8) \\
&= \quad \sum_{j=1}^{|\mathcal{S}|} \alpha_{t-1}(j)\, a_{ji}\, b_{i,y_t} \,, && (9)
\end{aligned}$$

for $t = 2, 3, \ldots, T$. The initial step is given by

$$\alpha_1(i) \quad = \quad P(Y_1 = y_1, S_1 = \sigma_i) \;=\; \iota_i\, b_{i,y_1} \,, \tag{10}$$

where the *initial* state probabilities $\iota_i \doteq P(S_1 = \sigma_i)$ satisfy $\sum_{i=1}^{|\mathcal{S}|} \iota_i = 1$.

Similarly, the standard backward formulation computes $\beta_t(i)$ via the recursion

$$\begin{aligned}
\beta_t(i) \quad &= \quad P(\mathbf{Y}_{t+1:T} = \mathbf{y}_{t+1:T} \mid S_t = \sigma_i) && (11) \\
&= \quad \sum_{j=1}^{|\mathcal{S}|} P(S_{t+1} = \sigma_j \mid S_t = \sigma_i)\, P(Y_{t+1} = y_{t+1} \mid S_{t+1} = \sigma_j)\, P(\mathbf{Y}_{t+2:T} = \mathbf{y}_{t+2:T} \mid S_{t+1} = \sigma_j) && (12) \\
&= \quad \sum_{j=1}^{|\mathcal{S}|} a_{ij}\, b_{j,y_{t+1}}\, \beta_{t+1}(j) \,, && (13)
\end{aligned}$$

for $t = T-1, T-2, \ldots, 1$. Note that this standard formulation allows each sequence to potentially continue forever, permitted by the weak constraint that $\beta_T(i) = 1$ for every $\sigma_i \in \mathcal{S}$.

However, when modelling complete sequences, such as those obtained from sentences of a grammar, it is necessary to include the fact that such sequences have both an implicit start, here already represented via ι_i, and also an implict end. For this purpose, we introduce start-of-sequence and end-of-sequence markers, $\triangleleft$ and $\triangleright$ respectively, such that the initial probabilities are now $\iota_i \doteq P(S_1 = \sigma_1 \mid S_0 = \triangleleft)$. Additionally, we introduce the *final* or *termination* probabilities $\tau_i \doteq P(S_{T+1} = \triangleright \mid S_T = \sigma_i)$, with the new constraint that $\beta_T(i) = \tau_i$.

Note that in some formulations, the pseudo-states $\triangleleft$ and $\triangleright$ are implicitly added into both the state space $\mathcal{S}$ and the transition matrix $\mathbf{A}$. However, I prefer to explicitly keep them separate. Hence, let us generalise the final probabilities to be stage-invariant, such that $\tau_i \doteq P(S_{t+1} = \triangleright \mid S_t = \sigma_i)$ is now the probability that the subsequence $\mathbf{y}_{1:t}$ terminates immediately after stage t from state $S_t = \sigma_i$. Conversely, let $\bar{\tau}_i \doteq 1 - \tau_i$ be the probability that the subsequence $\mathbf{y}_{1:t}$ does not terminate immediately after stage t. Note that if a subsequence does not terminate with stage t, then there **must** be a further transition to the next stage $t+1$. Hence, for the purposes of sequence generation, the introduction of termination probabilities allows a stochastic choice after every stage of whether or not to terminate the sequence.

Consequently, the forward pass now takes the form

$$\alpha_t(i) \quad = \quad \sum_{j=1}^{|\mathcal{S}|} \alpha_{t-1}(j)\, \bar{\tau}_j\, a_{ji}\, b_{i,y_t}\,, \tag{14}$$

for $t = 2, \ldots, T$, and the backward pass takes the form

$$\beta_t(i) \quad = \quad \sum_{j=1}^{|\mathcal{S}|} \bar{\tau}_i\, a_{ij}\, b_{j,y_{t+1}}\, \beta_{t+1}(j)\,, \tag{15}$$

for $t = T-1, \ldots, 1$.

1.2 Binary Hierarchical Structure

1.2.1 Top-down generative model

We now generalise the notion of a one-dimensional HMM to a two-dimensional structure representing a hierarchy of hidden states. As shown in the figure below, we consider a *generative* process that outputs a sequence of tokens (e.g. a sentence) from the grammar $\mathcal{G}$.

Here we restrict $\mathcal{G}$ to a context-free grammar using only unary and binary rules. The binary rules take the top-down form $\sigma_i \to \sigma_j \oplus \sigma_k$ for $\sigma_i, \sigma_j\sigma_k \in \mathcal{S}$, where σ_i is the *parent* state, and σ_j and σ_k are the *left child* and *right child* states, respectively. Similarly, the unary rules take the form $\sigma_i \to \nu_m$ for $\sigma_i \in \mathcal{S}$ and $\nu_m \in \mathcal{Y}$, where σ_i is again the parent state, and now ν_m is the child token. Thus, the final *derivation* is a binary tree from a root state $S_{1:T}$, through intermediate states $S_{s:t}$, down to leaf states $S_{t:t} \doteq S_t$, spanning the tokens $\mathbf{y}_{1:T}$.

If we let the *collection* $\mathbf{S}_{1:T}$ of variables loosely represent the entire hierarchy of hidden states, then

the derivation shown above has a joint probability that factors in top-down fashion as

$$\begin{aligned} P(\mathbf{S}_{1:T} = \mathtt{s}, \mathbf{Y}_{1:T} = \mathbf{y}) \quad = \quad & P(S_{1:6} = \mathtt{S})\, P(S_{1:2} = \mathtt{NP}, S_{3:6} = \mathtt{VP} \mid S_{1:6} = \mathtt{S})\, P(S_1 = \mathtt{D}, S_2 = \mathtt{N} \mid S_{1:2} = \mathtt{NP}) && (16) \\ & \times P(Y_1 = \mathtt{The} \mid S_1 = \mathtt{D})\, P(Y_2 = \mathtt{cat} \mid S_2 = \mathtt{N})\, P(S_3 = \mathtt{V}, S_{4:6} = \mathtt{PP} \mid S_{3:6} = \mathtt{VP}) && (17) \\ & \times P(Y_3 = \mathtt{sat} \mid S_3 = \mathtt{V})\, P(S_4 = \mathtt{P}, S_{5:6} = \mathtt{NP} \mid S_{4:6} = \mathtt{PP}) P(Y_4 = \mathtt{on} \mid S_4 = \mathtt{P}) && (18) \\ & \times P(S_5 = \mathtt{D}, S_5 = \mathtt{N} \mid S_{5:6} = \mathtt{NP})\, P(Y_5 = \mathtt{the} \mid S_5 = \mathtt{D})\, P(Y_6 = \mathtt{mat} \mid S_6 = \mathtt{N}) && (19) \end{aligned}$$

Note that in comparison to the corresponding HMM model, the hierarchical model lacks strong left-to-right dependencies between consecutive leaf states. However, the binary rules do implicitly constrain the left and right child states to span adjacent subsequences of tokens, such that right-to-left combinations are forbidden. This issue is re-examined in a later section.

1.2.2 Bottom-up parsing model

Conversely to the top-down, generative process that produces tokens, there is also notionally a corresponding bottom-up, *parsing* process from tokens to root state. Loosely, we suppose that the top-down rules of grammar $\mathcal{G}$ may be reversed to become unary rules of the form $\nu_m \to \sigma_i$ for $\nu_m \in \mathcal{Y}$ and $\sigma_i \in \mathcal{S}$, and binary rules of the form $\sigma_j \oplus \sigma_k \to \sigma_i$ for $\sigma_i, \sigma_j, \sigma_k \in \mathcal{S}$. Note that in this reversed form, σ_i has become a child state, and σ_j and σ_k have become parent states. In order to retain consistency with the generative process, we shall call σ_i the *head* state regardless of the direction, e.g. $\sigma_i \to \sigma_j \oplus \sigma_k$ or $\sigma_j \oplus \sigma_k \to \sigma_i$.

We now suppose that the contiguous sequence $\mathbf{y}_{1:T}$ of tokens can be *parsed* via a series of bottom-up unary and binary rules into a single structure, specifically a binary tree called a *parse* tree. Since the only difference between the top-down derivation and the bottom-up parse tree is the direction of the edges (i.e. the rules), then we may loosely refer to both structures (ignoring direction) as a *parse*. Hence, if there is a state $S_{s:t}$ spanning the subsequence $\mathbf{y}_{s:t}$ of tokens, we may say that $\mathbf{y}_{s:t}$ has a *subparse* with head state $S_{s:t}$.

Note, however, that directionality of the rules is important when computing probabilities, since $P_{\mathtt{gen}}(\mathtt{A} \to \mathtt{B}) \neq P_{\mathtt{parse}}(\mathtt{B} \to \mathtt{A})$ in general, due to the former being $P(\mathtt{B} \mid \mathtt{A})$ and the latter being $P(\mathtt{A} \mid \mathtt{B})$.

The parse tree for our example sentence is shown in the figure below.

This parse has conditional probability

$$\begin{aligned} P(\mathbf{S}_{1:T} = \mathtt{s} \mid \mathbf{Y}_{1:T} = \mathbf{y}) \quad = \quad & P(S_1 = \mathtt{D} \mid Y_1 = \mathtt{The})\, P(S_2 = \mathtt{N} \mid Y_2 = \mathtt{cat})\, P(S_{1:2} = \mathtt{NP} \mid S_1 = \mathtt{D}, S_2 = \mathtt{N}) && (20) \\ & \times P(S_5 = \mathtt{D} \mid Y_5 = \mathtt{the})\, P(S_6 = \mathtt{N} \mid Y_6 = \mathtt{mat})\, P(S_{5:6} = \mathtt{NP} \mid S_5 = \mathtt{D}, S_6 = \mathtt{N}) && (21) \\ & \times P(S_4 = \mathtt{P} \mid Y_4 = \mathtt{on})\, P(S_{4:6} = \mathtt{PP} \mid S_4 = \mathtt{P}, S_{5:6} = \mathtt{NP}) && (22) \\ & \times P(S_3 = \mathtt{V} \mid Y_3 = \mathtt{cat})\, P(S_{3:6} = \mathtt{VP} \mid S_3 = \mathtt{V}, S_{4:6} = \mathtt{PP}) && (23) \\ & \times P(S_{1:6} = \mathtt{S} \mid S_{1:2} = \mathtt{NP}, S_{3:6} = \mathtt{VP})\,. && (24) \end{aligned}$$

Observe that the order of combining tokens and head states in bottom-up token parsing differs, in general, from the order of generating head states and tokens in a top-down fashion.

1.2.3 Inside structure

The *inside* pass is analogous to the *backward* pass of a HMM. We assume that a subsequence $\mathbf{y}_{s:t}$ can be parsed into a substructure with given head state $S_{s:t}$. Unlike proper parsing, however, we

are not concerned with the actual substructure so obtained. Instead, we only wish to measure the overall likelihood that one or more such subparses exist. This probability is given by

$$\beta_{s:t}(i) \;\doteq\; P(\mathbf{Y}_{s:t} = \mathbf{y}_{s:t} \mid S_{s:t} = \sigma_i)\,. \tag{25}$$

Clearly, in the special case where $s = t$, this reduces to

$$\beta_{t:t}(i) \;=\; P(Y_t = y_t \mid S_t = \sigma_i)\,, \tag{26}$$

where we define $S_t \doteq S_{t:t}$ for convenience. Under the assumption that the latter distribution is invariant to the position t within the sequence, we define

$$P(Y_t = \nu_m \mid S_t = \sigma_i) \;\doteq\; P(\sigma_i \to \nu_m) \;=\; b_{im}\,, \tag{27}$$

where $\sum_{m=1}^{|\mathcal{Y}|} b_{im} = 1$ for all $i = 1, 2, \ldots, |\mathcal{S}|$. Hence, $\mathbf{B} = [b_{im}]$ is just the emission matrix from the corresponding HMM.

The more general case, however, is that $s < t$, for which there exists at least one $s \le r < t$ such that the substructure for subsequence $\mathbf{y}_{s:t}$ may be further partitioned into two adjacent substructures, one for subsequence $\mathbf{y}_{s:r}$ and the other for subsequence $\mathbf{y}_{r+1:t}$. In order for this to be plausible, the former subsequence must have some head state $S_{s:r} = \sigma_j$, and the latter must have some head state $S_{r+1:t} = \sigma_k$. Consequently, there must exist at least one (top-down) rule of the form $\sigma_i \to \sigma_j \oplus \sigma_k$ that satisfies the partitioning $S_{s:t} \to S_{s:r} \oplus S_{r+1:t}$ for $\mathbf{y}_{s:t} = \mathbf{y}_{s:r} \odot \mathbf{y}_{r+1:t}$.

Now, we noted above that we are not interested in the specific detail of a particular parse. Hence, we obscure all such substructural detail by summing over the probability of each subparse, for all permissible values of the partition variable r, and weight each substructure by the plausibility of the values of j and k. Thus, we obtain the recursive relation

$$\begin{aligned}
\beta_{s:t}(i) \;&=\; P(\mathbf{Y}_{s:t} = \mathbf{y}_{s:t} \mid S_{s:t} = \sigma_i)\,. && (28)\\
&=\; \sum_{r=s}^{t-1} P(\mathbf{Y}_{s:r} = \mathbf{y}_{s:r}, \mathbf{Y}_{r+1:t} = \mathbf{y}_{r+1:t} \mid S_{s:t} = \sigma_i)\,. && (29)\\
&=\; \sum_{r=s}^{t-1} \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} P(\mathbf{Y}_{s:r} = \mathbf{y}_{s:r} \mid S_{s:r} = \sigma_j)\, P(\mathbf{Y}_{r+1:t} = \mathbf{y}_{r+1:t} \mid S_{r+1:t} = \sigma_k) && (30)\\
&\qquad \times P(S_{s:r} = \sigma_j, S_{r+1:t} = \sigma_k \mid S_{s:t} = \sigma_i) && (31)\\
&=\; \sum_{r=s}^{t-1} \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} P(S_{s:r} = \sigma_j, S_{r+1:t} = \sigma_k \mid S_{s:t} = \sigma_i)\, \beta_{s:r}(j)\, \beta_{r+1:t}(k)\,, && (32)
\end{aligned}$$

for $s < t$. Under the assumption that the probabilities of the various binary rules are invariant to the positions within the sequence, we define

$$P(S_{s:r} = \sigma_j, S_{r+1:t} = \sigma_k \mid S_{s:t} = \sigma_i) \;\doteq\; P(\sigma_i \to \sigma_j \oplus \sigma_k) \;=\; a_{ijk}\,, \tag{33}$$

where $\sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} a_{ijk} = 1$ for $i = 1, 2, \ldots, |\mathcal{S}|$. Observe that $\mathbf{A} = [a_{ijk}]$ is now a three-dimensional tensor, in constrast to the two-dimensional transition matrix of the corresponding HMM.

Consequently, we obtain

$$\beta_{s:t}(i) \;=\; \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} a_{ijk} \sum_{r=s}^{t-1} \beta_{s:r}(j)\, \beta_{r+1:t}(k)\,, \tag{34}$$

for $s < t$. Observe that the subscripts of β go from a narrower range to a wider range, and hence the inside pass corresponds to a *bottom-up* pass. This is the direction that token parsing takes.

1.2.4 Outside structure

Given an *inside* substructure for some subsequence $\mathbf{y}_{s:t}$ with head state $S_{s:t} = \sigma_i$, the rest of the parse, spanning the remaining subsequences $\mathbf{y}_{1:s-1}$ and $\mathbf{y}_{t+1:T}$, forms the *outside* structure. Analagous to the *forward* pass of a HMM, the outside structure has joint probability

$$\alpha_{s:t}(i) \doteq P(\mathbf{Y}_{1:s-1} = \mathbf{y}_{1:s-1}, \mathbf{Y}_{t+1:T} = \mathbf{y}_{t+1:T}, S_{s:t} = \sigma_i)\,. \tag{35}$$

We observe that

$$\begin{aligned} \alpha_{s:t}(i)\,\beta_{s:t}(i) &= P(\mathbf{Y}_{1:s-1} = \mathbf{y}_{1:s-1}, \mathbf{Y}_{t+1:T} = \mathbf{y}_{t+1:T}, S_{s:t} = \sigma_i)\, P(\mathbf{Y}_{s:t} = \mathbf{y}_{s:t} \mid S_{s:t} = \sigma_i) && (36)\\ &= P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{s:t} = \sigma_i)\,, && (37) \end{aligned}$$

for every $s \leq t$ with $s, t \in \{1, 2, \ldots, T\}$.

At the extreme, for $s = 1$ and $t = T$, the outside probability reduces to

$$\alpha_{1:T}(i) = P(S_{1:T} = \sigma_i)\,, \tag{38}$$

for $i = 1, 2, \ldots, |\mathcal{S}|$. It is usual, but not necessary, for there to be a single, distinguished state at the root of every complete sequence $\mathbf{y}_{1:T}$ (see the section on root states for more detail).

We now consider the other cases where $s > 1$ and/or $t < T$. For $s > 1$, there exists at least one $1 \leq r \leq s-1$ such that we may subdivide the left-hand part of the outer structure to include an *inner* substructure spanning subsequence $\mathbf{y}_{r:s-1}$, adjacent to the existing inner substructure for $\mathbf{y}_{s:t}$. We give this new substructure some head state σ_j, and combine the two adjacent, inner substructures with some rule $\sigma_k \to \sigma_j \oplus \sigma_i$. The result of the combination is a single inner structure spanning $\mathbf{y}_{r:t}$ with head state $S_{r:t} = \sigma_k$, which forms part of the inner probability $\beta_{r:t}(k)$. What remains is therefore a new outer structure forming part of the outer probability $\alpha_{r:t}(k)$.

This 'left-hand-side' probability is thus

$$\begin{aligned} \alpha^{\texttt{LHS}}_{s:t}(i) &= \sum_{r=1}^{s-1} P(\mathbf{Y}_{1:r-1} = \mathbf{y}_{1:r-1}, \mathbf{Y}_{r:s-1} = \mathbf{y}_{r:s-1}, \mathbf{Y}_{t+1:T} = \mathbf{y}_{t+1:T}, S_{s:t} = \sigma_i) && (39)\\ &= \sum_{r=1}^{s-1} \sum_{j=1}^{|\mathcal{S}|} P(\mathbf{Y}_{r:s-1} = \mathbf{y}_{r:s-1} \mid S_{r:s-1} = \sigma_j)\, P(\mathbf{Y}_{1:r-1} = \mathbf{y}_{1:r-1}, \mathbf{Y}_{t+1:T} = \mathbf{y}_{t+1:T}, S_{s:t} = \sigma_i, S_{r:s-1} = \sigma_j) && (40)\\ &= \sum_{r=1}^{s-1} \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} P(\mathbf{Y}_{r:s-1} = \mathbf{y}_{r:s-1} \mid S_{r:s-1} = \sigma_j)\, P(\mathbf{Y}_{1:r-1} = \mathbf{y}_{1:r-1}, \mathbf{Y}_{t+1:T} = \mathbf{y}_{t+1:T}, S_{r:t} = \sigma_k) && (41)\\ &\qquad \times P(S_{r:s-1} = \sigma_j, S_{s:t} = \sigma_i \mid S_{r:t} = \sigma_k) && (42)\\ &= \sum_{r=1}^{s-1} \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} P(\sigma_k \to \sigma_j \oplus \sigma_i)\, \beta_{r:s-1}(j)\, \alpha_{r:t}(k) && (43)\\ &= \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} a_{kji} \sum_{r=1}^{s-1} \alpha_{r:t}(k)\, \beta_{r:s-1}(j)\,. && (44) \end{aligned}$$

Similarly, for $t < T$ we may choose at least one $t+1 \leq r \leq T$ by which to subdivide the right-hand side of the outer structure to include a new inner substructure spanning $\mathbf{y}_{t+1:r}$ with head state

$S_{t+1:r} = \sigma_j$. This new substructure is adjacent to the substructure spanning $\mathbf{y}_{s:t}$, and so we can again follow the logic described above. The resulting 'right-hand-side' probability is thus

$$\begin{aligned}
\alpha_{s:t}^{\texttt{RHS}}(i) &= \sum_{r=t+1}^{T} P(\mathbf{Y}_{1:s-1} = \mathbf{y}_{1:s-1}, \mathbf{Y}_{t+1:r} = \mathbf{y}_{t+1:r}, \mathbf{Y}_{r+1:T} = \mathbf{y}_{r+1:T}, S_{s:t} = \sigma_i) && (45)\\
&= \sum_{r=t+1}^{T} \sum_{j=1}^{|\mathcal{S}|} P(\mathbf{Y}_{t+1:r} = \mathbf{y}_{t+1:r} \mid S_{t+1:r} = \sigma_j)\, P(\mathbf{Y}_{1:s-1} = \mathbf{y}_{1:s-1}, \mathbf{Y}_{r+1:T} = \mathbf{y}_{r+1:T}, S_{s:t} = \sigma_i, S_{t+1:r} = \sigma_j) && (46)\\
&= \sum_{r=t+1}^{r} \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} P(\mathbf{Y}_{t+1:r} = \mathbf{y}_{t+1:r} \mid S_{t+1:r} = \sigma_j)\, P(\mathbf{Y}_{1:s-1} = \mathbf{y}_{1:s-1}, \mathbf{Y}_{r+1:T} = \mathbf{y}_{r+1:T}, S_{s:r} = \sigma_k) && (47)\\
&\qquad \times P(S_{s:t} = \sigma_i, S_{t+1:r} = \sigma_j \mid S_{s:r} = \sigma_k) && (48)\\
&= \sum_{r=t+1}^{T} \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} P(\sigma_k \to \sigma_i \oplus \sigma_j)\, \beta_{t+1:r}(j)\, \alpha_{s:r}(k) && (49)\\
&= \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} a_{kij} \sum_{r=t+1}^{T} \alpha_{s:r}(k)\, \beta_{t+1:r}(j)\,. && (50)
\end{aligned}$$

Consequently, the total outside probability is now just

$$\begin{aligned}
\alpha_{s:t}(i) &= \alpha_{s:t}^{\texttt{LHS}}(i) + \alpha_{s:t}^{\texttt{RHS}}(i) && (51)\\
&= \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} \left\{ a_{kji} \sum_{r=1}^{s-1} \alpha_{r:t}(k)\, \beta_{r:s-1}(j) + a_{kij} \sum_{r=t+1}^{T} \alpha_{s:r}(k)\, \beta_{t+1:r}(j) \right\}, && (52)
\end{aligned}$$

for $s > 1$ or $t < T$. Observe that the subscripts on α go from a wider range to a narrower range, and hence the outside pass corresponds to a *top-down* pass. This is the direction that token generation takes.

1.2.5 Arbitrary structure

The above formulae for the inside and outside passes deliberately obscure individual parsing structures. However, sometimes we desire more detailed knoweledge, especially about which states have been used.

In particular, we assume that every token sequence $\mathbf{y}_{1:T}$ can in fact be parsed up to some root state $S_{1:T}$. The likelihood of an observed sequence $\mathbf{y}_{1:T}$ is therefore given by

$$P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}) = \sum_{i=1}^{|\mathcal{S}|} P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{1:T} = \sigma_i) = \sum_{i=1}^{|\mathcal{S}|} \alpha_{1:T}(i)\, \beta_{1:T}(i)\,. \tag{53}$$

However, note that our notation still admits some ambiguity, since

$$\sum_{i=1}^{|\mathcal{S}|} P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{s:t} = \sigma_i) \neq P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T})\,, \tag{54}$$

in general. The left-hand side of this inequality gives the joint probability of the observed sequence **and** the fact that the subsequence $\mathbf{y}_{s:t}$ is spaned by some substructure with arbitrary head state

$S_{s:t}$. There is no reason why different parses with different substructures should be equiprobable. Thus, with slightly less ambiguity, we define

$$P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{s:t} = *) \doteq \sum_{i=1}^{|\mathcal{S}|} P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{s:t} = \sigma_i) = \sum_{i=1}^{|\mathcal{S}|} \alpha_{s:t}(i)\, \beta_{s:t}(i)\,. \tag{55}$$

By similar reasoning, we note that the values of s and t are not particularly special, and so we may sum over these values instead. However, we must be careful to distinguish between the two modes of head state $S_{s:t} = \sigma_i$. For $s = t$, we **must** have a unary rule of the form $\sigma_i \to \nu_m$, whereas for $s < t$ we **must** have a binary rule of the form $\sigma_i \to \sigma_j \oplus \sigma_k$. This is the primary reason for our choice of the way we condition probabilities, as briefly discussed in the introductory section.

Hence, using the shorthand that $S_t \doteq S_{t:t}$, we define the unary case as

$$P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_* = \sigma_i) \doteq \sum_{t=1}^{T} P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_t = \sigma_i) = \sum_{t=1}^{T} \alpha_{t:t}(i)\, \beta_{t:t}(i)\,. \tag{56}$$

This is the joint probability of observing the sequence $\mathbf{y}_{1:T}$ coupled with the fact that state σ_i appears as the head state of a single token, at least once in the derivation.

Similarly, the binary case is defined as

$$P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{*:*} = \sigma_i) \doteq \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{s:t} = \sigma_i) = \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \alpha_{s:t}(i)\, \beta_{s:t}(i) \tag{57}$$

This is the joint probability of observing the sequence $\mathbf{y}_{1:T}$ coupled with the fact that state σ_i appears as a head state spanning two or more tokens, at least once in the derivation.

By similar reasoning, we now have

$$\begin{aligned} P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_* = \sigma_i, \sigma_i \to \nu_m) &= \sum_{t=1}^{T} P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{t:t} = \sigma_i, y_t = \nu_m) && (58)\\ &= \sum_{t=1}^{T} \delta(y_t = \nu_m)\, P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{t:t} = \sigma_i) && (59)\\ &= \sum_{t=1}^{T} \delta(y_t = \nu_m)\, \alpha_{t:t}(i)\, \beta_{t:t}(i)\,, && (60) \end{aligned}$$

for the unary case, and hence

$$\begin{aligned} P(\sigma_i \to \nu_m \mid \mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_* = \sigma_i) &= \frac{P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_* = \sigma_i, \sigma_i \to \nu_m)}{P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_* = \sigma_i)} && (61)\\ &= \frac{\sum_{t=1}^{T} \delta(y_t = \nu_m)\, \alpha_{t:t}(i)\, \beta_{t:t}(i)}{\sum_{t=1}^{T} \alpha_{t:t}(i)\, \beta_{t:t}(i)} && (62)\\ &= b_{im}\, \frac{\sum_{t=1}^{T} \delta(y_t = \nu_m)\, \alpha_{t:t}(i)}{\sum_{t=1}^{T} \alpha_{t:t}(i)\, \beta_{t:t}(i)}\,. && (63) \end{aligned}$$

Clearly this is normalised to unity summing over all $\nu_m \in \mathcal{Y}$.

For the binary rule, we similarly have that

$$
\begin{aligned}
P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{*:*} = \sigma_i, \sigma_i \to \sigma_j \oplus \sigma_k) &= \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \sum_{r=s}^{t-1} P(\mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{s:t} = \sigma_i, S_{s:r} = \sigma_j, S_{r+1:t} = \sigma_k) && (64)\\
&= \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \sum_{r=s}^{t-1} P(\mathbf{Y}_{s:r} = \mathbf{y}_{s:r} \mid S_{s:r} = \sigma_j) && (65)\\
&\qquad \times P(\mathbf{Y}_{r+1:t} = \mathbf{y}_{r+1:t} \mid S_{r+1:t} = \sigma_k) && (66)\\
&\qquad \times P(S_{s:r} = \sigma_j, S_{r+1:t} = \sigma_k \mid S_{s:t} = \sigma_i) && (67)\\
&\qquad \times P(\mathbf{Y}_{1:s-1} = \mathbf{y}_{1:s-1}, \mathbf{Y}_{t+1:T} = \mathbf{y}_{t+1:T}, S_{s:t} = \sigma_i) && (68)\\
&= \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \sum_{r=s}^{t-1} \beta_{s:r}(j)\, \beta_{r+1:t}(k)\, P(\sigma_i \to \sigma_j \oplus \sigma_k)\, \alpha_{s:t}(i) && (69)\\
&= a_{ijk} \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \alpha_{s:t}(i) \sum_{r=s}^{t-1} \beta_{s:r}(j)\, \beta_{r+1:t}(k)\,, && (70)
\end{aligned}
$$

and hence

$$
P(\sigma_i \to \sigma_j \oplus \sigma_k \mid \mathbf{Y}_{1:T} = \mathbf{y}_{1:T}, S_{*:*} = \sigma_i) = a_{ijk} \frac{\sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \alpha_{s:t}(i) \sum_{r=s}^{t-1} \beta_{s:r}(j)\, \beta_{r+1:t}(k)}{\sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \alpha_{s:t}(i)\, \beta_{s:t}(i)} \quad (71)
$$

We observe that this is normalised to unity when summed over j and k, since we have previously established that

$$
\beta_{s:t}(i) = \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{|\mathcal{S}|} a_{ijk} \sum_{r=s}^{t-1} \beta_{s:r}(j)\, \beta_{r+1:t}(k)\,, \quad (72)
$$

for $s < t$.

1.3 Conditional Estimates

A major role of the inside-outside algorithm is to estimate the conditional probabilities $\mathbf{A} = [a_{ijk}]$ of the binary rules $\sigma_i \to \sigma_j \oplus \sigma_k$, and the conditional probabilities $\mathbf{B} = [b_{im}]$ of the unary rules $\sigma_i \to \nu_m$. These values are typically estimated from observed data via the *Expectation-Maximisation* algorithm.

1.3.1 Expectation-maximisation

Suppose that we have observed a collection $\mathbb{Y} = \left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(N)}\right)$ of indepdendent token sequences of arbitrary lengths $T^{(1)} = \left|\mathbf{y}^{(1)}\right|, \ldots, T^{(N)} = \left|\mathbf{y}^{(N)}\right|$. Next, we notionally *complete* the data by assuming that each sequence $\mathbf{y}^{(d)}$ has a true but unknown parse structure loosely represented by $\mathbf{s}^{(d)}$. Clearly, the allowable parses depend upon both the grammar $\mathcal{G}$ and the length $T^{(d)}$ of the sequence. Hence, we take $\mathbf{s}^{(d)} \in \boldsymbol{\mathcal{S}}_{T^{(d)}}$. The complete data therefore takes the form $\mathbb{C} = ((\mathbf{s}^{(d)}, \mathbf{y}^{(d)}))_{d=1}^{N}$.

Assuming the complete cases are independent, the overall likelihood is given by

$$
P(\mathbb{C} \mid \Theta) = \prod_{d=1}^{N} P(\mathbf{s}^{(d)}, \mathbf{y}^{(d)} \mid \Theta)\,, \quad (73)
$$

where $\Theta = (\mathbf{A}, \mathbf{B})$ represents the current parameters of the grammar. However, since the true parse structure $\mathbf{s}^{(d)}$ of each sequence $\mathbf{y}^{(d)}$ is unknown, we invent a notional zero/one indicator $z_{d,\mathbf{s}} \doteq \delta(\mathbf{s} = \mathbf{s}^{(d)})$, such that

$$P(\mathbb{C} \mid \Theta) \;=\; \prod_{d=1}^{N} \prod_{\mathbf{s} \in \boldsymbol{\mathcal{S}}_{T^{(d)}}} P(\mathbf{s}, \mathbf{y}^{(d)} \mid \Theta)^{z_{d,\mathbf{s}}} \tag{74}$$

$$\Rightarrow L(\Theta) \;\doteq\; \ln P(\mathbb{C} \mid \Theta) \;=\; \sum_{d=1}^{N} \sum_{\mathbf{s} \in \boldsymbol{\mathcal{S}}_{T^{(d)}}} z_{d,\mathbf{s}} \ln P(\mathbf{s}, \mathbf{y}^{(d)} \mid \Theta)\,. \tag{75}$$

The expected value of the complete log-likelihood is therefore

$$Q(\Theta, \Theta') \;\doteq\; \mathbb{E}\left[L(\Theta) \mid \mathbb{Y}, \Theta'\right] \tag{76}$$

$$= \sum_{d=1}^{N} \sum_{\mathbf{s} \in \boldsymbol{\mathcal{S}}_{T^{(d)}}} \mathbb{E}\left[z_{d,\mathbf{s}} \mid \mathbb{Y}, \Theta'\right] \ln P(\mathbf{s}, \mathbf{y}^{(d)} \mid \Theta) \tag{77}$$

$$= \sum_{d=1}^{N} \sum_{\mathbf{s} \in \boldsymbol{\mathcal{S}}_{T^{(d)}}} P(\mathbf{s} \mid \mathbf{y}^{(d)}, \Theta') \ln P(\mathbf{s}, \mathbf{y}^{(d)} \mid \Theta)\,, \tag{78}$$

where Θ' is a prior estimate of Θ. The expectation-maximisation (EM) algorithm then proceeds by choosing a new estimate $\hat{\Theta}$ such that

$$\hat{\Theta} \;=\; \arg\max_{\Theta} Q(\Theta, \Theta')\,. \tag{79}$$

Alternatively, the generalised expectation-maximisation (GEM) algorithm proceeds more simply by choosing a new estimate $\hat{\Theta}$ such that

$$Q(\hat{\Theta}, \Theta') \;>\; Q(\Theta', \Theta')\,. \tag{80}$$

In either case, the estimate is iteratively updated via $\Theta' \leftarrow \hat{\Theta}$. Under mild conditions, the iterations will converge to a local optimum Θ^*.

1.3.2 Expected binary rule counts

As an example, consider now the simple sequence $\mathbf{y}^{(d)} = (\texttt{John}, \texttt{ran})$ with ground-truth parse $\mathbf{s}^{(d)} = (S_{1:2} = \texttt{S}, S_{1:1} = \texttt{N}, S_{2:2} = \texttt{V})$, where here `S` represents the state of a complete sentence, `N` represents a noun, and `V` represents a verb. We observe that the appropriate indicator is

$$z_{d,\mathbf{s}} \;=\; \delta(S_{1:2} = \texttt{S} \wedge S_{1:1} = \texttt{N} \wedge S_{2:2} = \texttt{V})\,, \tag{81}$$

such that

$$\mathbb{E}[z_{d,\mathbf{s}} \mid \mathbf{y}^{(d)}, \Theta] \;=\; P(S_{1:2} = \texttt{S}, S_{1:1} = \texttt{N}, S_{2:2} = \texttt{V} \mid \mathbf{y}^{(d)}, \Theta) \tag{82}$$

$$= P(S_{1:2} = \texttt{S} \mid \mathbf{y}^{(d)}, \Theta)\, P(S_{1:1} = \texttt{N}, S_{2:2} = \texttt{V} \mid S_{1:2} = \texttt{S}, \mathbf{y}^{(d)}, \Theta)\,. \tag{83}$$

The latter term gives the empirical probability of the rule $\texttt{S} \rightarrow \texttt{N} \oplus \texttt{V}$.

More generally, there might be a number of opportunities for a given rule $\sigma_i \to \sigma_j \oplus \sigma_k$ to appear in parse $\mathbf{s}^{(d)}$, e.g. $P(S_{s:r} = \sigma_j, S_{r+1:t} = \sigma_k \mid S_{s:t} = \sigma_i, \mathbf{y}^{(d)}, \Theta)$. In fact, from the previous section we have

$$P(S_{s:r} = \sigma_j, S_{r+1:t} = \sigma_k, S_{s:t} = \sigma_i, \mathbf{Y} = \mathbf{y}^{(d)} \mid \Theta) \quad = \quad a_{j,k|i}\, \beta^{(d)}_{s:r}(j)\, \beta^{(d)}_{r+1:t}(k)\, \alpha^{(d)}_{s:t}(i) \tag{84}$$

$$\Rightarrow P(S_{s:r} = \sigma_j, S_{r+1:t} = \sigma_k, S_{s:t} = \sigma_i \mid \mathbf{y}^{(d)}, \Theta) \quad = \quad \frac{a_{j,k|i}\, \beta^{(d)}_{s:r}(j)\, \beta^{(d)}_{r+1:t}(k)\, \alpha^{(d)}_{s:t}(i)}{\sum_{i'=1}^{|\mathcal{S}|} \alpha^{(d)}_{1:T^{(d)}}(i')\, \beta^{(d)}_{1:T^{(d)}}(i')}\,. \tag{85}$$

Note for interest that by summing this over r, j and k we obtain

$$P(S_{s:t} = \sigma_i \mid \mathbf{y}^{(d)}, \Theta) \quad = \quad \frac{\alpha^{(d)}_{s:t}(i)\, \beta^{(d)}_{s:t}(i)}{\sum_{i'=1}^{|\mathcal{S}|} \alpha^{(d)}_{1:T^{(d)}}(i')\, \beta^{(d)}_{1:T^{(d)}}(i')}\,, \tag{86}$$

which is the joint probability that subsequence $\mathbf{y}^{(d)}_{s:t}$ is spanned by a subparse **and** the head of the subparse is σ_i. Hence, the probability that subsequence $\mathbf{y}^{(d)}_{s:t}$ is spanned by some subparse is

$$P(S_{s:t} = * \mid \mathbf{y}^{(d)}, \Theta) \quad = \quad \frac{\sum_{i=1}^{|\mathcal{S}|} \alpha^{(d)}_{s:t}(i)\, \beta^{(d)}_{s:t}(i)}{\sum_{i'=1}^{|\mathcal{S}|} \alpha^{(d)}_{1:T^{(d)}}(i')\, \beta^{(d)}_{1:T^{(d)}}(i')}\,. \tag{87}$$

Observe that $P(S_{1:T^{(d)}} = * \mid \mathbf{y}^{(d)}, \Theta) = 1$, as required, since we have assumed from the outset that every observed sequence has been generated by the grammar $\mathcal{G}$ and thus must be parsable.

The conditional probability of observing joint states $(S_{s:t}, S_{s:r}, S_{r+1:t}) = (\sigma_i, \sigma_j, \sigma_k)$ for a particular substructure $s : r : t$ now generalises over all such substructures to the expected count of observing the tuple $(\sigma_i, \sigma_j, \sigma_k)$, namely

$$\hat{c}^{(d)}_{ijk} \quad \doteq \quad \sum_{s=1}^{T^{(d)}-1} \sum_{t=s+1}^{T^{(d)}} \sum_{r=s}^{t-1} P(S_{s:r} = \sigma_j, S_{r+1:t} = \sigma_k, S_{s:t} = \sigma_i \mid \mathbf{y}^{(d)}, \Theta)\,. \tag{88}$$

Note that this is just the maximisation step of EM, where we are implicitly using the old estimate of Θ, i.e. Θ' without the prime, to obtain the new estimate $\hat{\Theta}$.

Hence, over all observed data $\mathbb{Y}$, we have total expected counts

$$\hat{c}_{ijk} \quad \doteq \quad \sum_{d=1}^{N} \hat{c}^{(d)}_{ijk}\,. \tag{89}$$

The updated estimate of binary rule probabilities, $\mathbf{A}$, is now obtained via

$$\hat{a}_{ijk} \quad = \quad \frac{\hat{c}_{ijk}}{\hat{c}_{i\cdot\cdot}}\,. \tag{90}$$

More efficiently, we might instead compute

$$\check{c}^{(d)}_{ijk} \quad \doteq \quad \sum_{s=1}^{T^{(d)}-1} \sum_{t=s+1}^{T^{(d)}} \alpha^{(d)}_{s:t}(i) \sum_{r=s}^{t-1} \beta^{(d)}_{s:r}(j)\, \beta^{(d)}_{r+1:t}(k)\,, \tag{91}$$

and

$$\check{c}_{ijk} \doteq \sum_{d=1}^{N} \left\{ \frac{\check{c}_{ijk}^{(d)}}{\sum_{i'=1}^{|\mathcal{S}|} \alpha_{1:T^{(d)}}^{(d)}(i')\, \beta_{1:T^{(d)}}^{(d)}(i')} \right\}, \tag{92}$$

such that

$$\hat{a}_{ijk} = \frac{a_{ijk}\, \check{c}_{ijk}}{\sum_{j'=1}^{|\mathcal{S}|} \sum_{k'=1}^{|\mathcal{S}|} a_{ij'k'}\, \check{c}_{ij'k'}}. \tag{93}$$

Observe from this formulation that any zero probability in a_{ijk} remains zero in $\hat{a}_{ijk}$. Consequently, *structural restrictions* in the binary rules of the grammar $\mathcal{G}$ are preserved. In particular, the head of any binary rule is an intermediate state.

1.3.3 Expected unary rule counts

Similarly to the above case for binary rules, for the unary rules we deduce that the probability of observing the tuple $(S_t, Y_t) = (\sigma_i, \nu_m)$ at position t is given by

$$P(S_t = \sigma_i, Y_t = \nu_m \mid \mathbf{y}^{(d)}, \Theta) = \frac{\delta(y_t^{(d)} = \nu_m)\, \alpha_{t:t}^{(d)}(i)\, \beta_{t:t}^{(d)}(i)}{\sum_{i'=1}^{|\mathcal{S}|} \alpha_{1:T^{(d)}}^{(d)}(i')\, \beta_{1:T^{(d)}}^{(d)}(i')}. \tag{94}$$

Hence, we may define the per-sequence expected counts

$$\hat{c}_{im}^{(d)} \doteq \sum_{t=1}^{T^{(d)}} P(S_t = \sigma_i, Y_t = \nu_m \mid \mathbf{y}^{(d)}, \Theta), \tag{95}$$

and the total expected counts

$$\hat{c}_{im} \doteq \sum_{d=1}^{N} \hat{c}_{im}^{(d)}, \tag{96}$$

such that the unary rule probabilities, $\mathbf{B}$, will be updated via

$$\hat{b}_{im} = \frac{\hat{c}_{im}}{\hat{c}_{i\cdot}}. \tag{97}$$

Now, we observe that if $y_t^{(d)} = \nu_m$ then $\beta_{t:t}^{(d)}(i) = P(Y_t = y_t^{(d)} \mid S_t = \sigma_i, \Theta) = b_{m|i}$. Hence, we may more efficiently compute

$$\check{c}_{im}^{(d)} \doteq \sum_{t=1}^{T^{(d)}} \delta(y_t^{(d)} = \nu_m)\, \alpha_{t:t}^{(d)}(i), \tag{98}$$

and

$$\check{c}_{im} \doteq \sum_{d=1}^{N} \left\{ \frac{\check{c}_{im}^{(d)}}{\sum_{i'=1}^{|\mathcal{S}|} \alpha_{1:T^{(d)}}^{(d)}(i')\, \beta_{1:T^{(d)}}^{(d)}(i')} \right\}, \tag{99}$$

such that

$$\hat{b}_{im} = \frac{b_{im}\, \check{c}_{im}}{\sum_{m'=1}^{|\mathcal{Y}|} b_{im'}\, \check{c}_{im'}}. \tag{100}$$

Observe from this formulation that any zero probability in b_{im} remains zero in $\hat{b}_{im}$. Consequently, structural restrictions in the unary rules of the grammar $\mathcal{G}$ are preserved. In particular, the head of a unary rule is a leaf state.

1.4 Grammatical Restrictions

The inside-outside process involves the finite set $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots\}$ of discrete states that form the *non-terminal* symbols, and the finite set $\mathcal{Y} = \{\nu_1, \nu_2, \ldots\}$ of output tokens that form the *terminal* symbols. Each grammatical structure spanning a contiguous sequence $\mathbf{y}$ of tokens takes the form of a binary parse tree. The stochastic process itself utilises binary rule probabilities, $\mathbf{A} = [a_{ijk}]$, and unary rule probabilities, $\mathbf{B} = [b_{im}]$, where the rules have been defined in a top-down fashion suitable for token generation.

As developed so far, these are the only restrictions placed on the grammar. However, there are situations where it might be useful to exert more control over the grammar. In particular, due to the binary nature of the parse tree, we may further distinguish between the non-terminal states. Thus, we let $\mathcal{S} = \mathcal{S}_{\texttt{root}} \cup \mathcal{S}_{\texttt{int}} \cup \mathcal{S}_{\texttt{leaf}}$, with *root* states $\mathcal{S}_{\texttt{root}}$, *leaf* states $\mathcal{S}_{\texttt{leaf}}$, and *intermediate* states $\mathcal{S}_{\texttt{int}}$. Whether or not these susets overlap depends upon the restrictions placed upon the grammar. By default, with no restrictions, we have $\mathcal{S}_{\texttt{root}} = \mathcal{S}_{\texttt{leaf}} = \mathcal{S}_{\texttt{int}} = \mathcal{S}$.

1.4.1 Root states

The root states $\mathcal{S}_{\texttt{root}} \subseteq \mathcal{S}$ determine which non-terminal symbols may appear at the root of the parse tree of a sequence $\mathbf{y}_{1:T}$. Thus, we suppose that $\alpha_{1:T}(i) = 0$ for every $\sigma_i \in \mathcal{S} \backslash \mathcal{S}_{\texttt{root}}$, and $\alpha_{1:T}(i) > 0$ for every $\sigma_i \in \mathcal{S}_{\texttt{root}}$.

A common usage of the inside-outside algorithm in natural language processing (NLP) is to automatically extract a grammar from a corpus of tokenised sentences. It is typical in this scenario to assume that each sentence $\mathbf{y}_{1:T}$ begins with a single, common state, say $S_{1:T} = \texttt{S}$ with unit probability.

However, what then do we make of imperative sentences like "*Go!*"? We observe that the word "go" here typically takes a verbal form, say `V` for simplicity, which is clearly not `S`. One possibility is to include punctuation in the token vocabulary, such that the appropriate rule might be $\texttt{S} \to \texttt{V} \oplus !$. Another possibility is to allow multiple root states, such as allowing `V` to also indicate an imperative sentence, or allowing another explicit root state like $\texttt{S}_{\texttt{imp}}$ along with a unary rule like $\texttt{S}_{\texttt{imp}} \to \texttt{go}$.

Another issue is whether or not root states are allowed to recur internally within the parse. For example, "*The cat sat.*" and "*The cat sat on the mat.*" are both sentences. Should we allow rules of the form $\texttt{S} \to \texttt{S} \oplus \texttt{PP}$, where `PP` represents a prepositional phrase? Doing so presents a problem for indicating prepositional attachment, e.g. in "*I saw the man with the telescope.*" did I use a telescope (verb attachment), or was the man carrying a telescope (noun attachment)?

A reasonable proposal might be to only allow root states to be used once per parse, i.e. $\mathcal{S}_{\texttt{root}} \cap \mathcal{S}_{\texttt{int}} = \emptyset$ and $\mathcal{S}_{\texttt{root}} \cap \mathcal{S}_{\texttt{leaf}} = \emptyset$. However, what if in response to the question "*What is the dog chasing?*" the answer is "*The cat.*", which is a noun phrase, `NP`? If we let $\texttt{NP} \in \mathcal{S}_{\texttt{root}}$, then it doesn't make sense to prevent it from being reused multiple times, e.g. in "*The cat sat on the mat.*".

1.4.2 Leaf states

The leaf states $\mathcal{S}_{\texttt{leaf}} \subseteq \mathcal{S}$ determine which non-terminal symbols may appear at the leaves of the parse tree of a sequence $\mathbf{y}_{1:T}$. Leaf states are special in that they are solely responsible for the production of the terminal symbols $y_t \in \mathcal{Y}$. Thus, we suppose that $\beta_{t:t}(i) = 0$ for every $\sigma_i \in \mathcal{S} \backslash \mathcal{S}_{\texttt{leaf}}$, which (in order to sum probabilities to unity) might require the implicit use of a non-observable pseudo-token, say $\nu_0 = \square$. Conversely, we suppose that $\beta_{t:t}(i) > 0$ for every $\sigma_i \in \mathcal{S}_{\texttt{leaf}}$.

In practice, the assigned leaf state S_t for each token y_t represents a *category*, or (in NLP) a *part-of-speech* (POS) tag, which specifies the low-level role that token has within the entire parse. For example, in the NLP domain each sentence in the training corpus could (in principle) could be deterministically POS-tagged prior to using the inside-outside algorithm. Thus, these POS-tags could represent the leaf states of the corresponding parse. Alternatively, if we are willing to assume that the assigned POS-tags are always perfectly accurate (which is not true, in practice), then we could instead use the POS-tags themselves as tokens.

Note that POS-tagging the corpus offers the opportunity to automatically build a stochastic mapping from the vocabulary to the POS-tags. In fact, if we subsequently dropped the assigned POS-tags prior to building the grammar, then the use of the stochastic mapping from tokens to leaf states permits some error correction of incorrectly assigned POS-tags.

As discussed in the previous section, using special leaf states causes difficulty for one-token sequences (e.g. from the sentence "*Go!*"), since then the leaf state S_1 and the root state $S_{1:1}$ are identical, implying that $\mathcal{S}_{\texttt{leaf}} \cap \mathcal{S}_{\texttt{root}} \neq \emptyset$. Two-token sequences (e.g. from "*Go home!*") are also potentially problematic, since then the parse tree has no intermediate states (see the next section). Consequently, the appropriate binary rule, say $\sigma_i \to \sigma_j \oplus \sigma_k$, for a two-token sequence must obey $\sigma_i \in \mathcal{S}_{\texttt{root}}$ but have $\sigma_j, \sigma_k \in \mathcal{S}_{\texttt{leaf}}$. This means that rules of the (templated) form $\mathcal{S}_{\texttt{root}} \to \mathcal{S}_{\texttt{leaf}} \oplus \mathcal{S}_{\texttt{leaf}}$ must be permitted.

1.4.3 Intermediate states

The intermediate states $\mathcal{S}_{\texttt{int}} \subseteq \mathcal{S}$ are assigned to nodes in the binary parse tree that are neither the root node nor any of the leaf nodes. Recall that for a sequence of length $T = |\mathbf{y}|$, the binary parse tree has $2T - 1$ non-terminal nodes, with T leaf nodes, 1 root node (which is also the leaf node when $T = 1$), and thus $T - \delta(T > 1) - 1$ intermediate nodes. Consequently, for sequences of length $T \geq 3$, the parse tree has one or more intermediate nodes. Hence, such a tree must contain at least one binary rule, e.g. $\sigma_i \to \sigma_j \oplus \sigma_k$, that results in an intermediate head state, i.e. $\sigma_i \in \mathcal{S}_{\texttt{int}}$.

For example, the three-token sentence "*You go home!*" will (by dropping punctuation) give rise to a parse tree having exactly one intermediate state, with a binary rule taking the form $\mathcal{S}_{\texttt{int}} \to \mathcal{S}_{\texttt{leaf}} \oplus \mathcal{S}_{\texttt{leaf}}$. For the root state, the binary rule must therefore take either the form $\mathcal{S}_{\texttt{root}} \to \mathcal{S}_{\texttt{leaf}} \oplus \mathcal{S}_{\texttt{int}}$, or $\mathcal{S}_{\texttt{root}} \to \mathcal{S}_{\texttt{int}} \oplus \mathcal{S}_{\texttt{leaf}}$.

At this point, it is feasible to separate leaf states from intermediate states, i.e. $\mathcal{S}_{\texttt{leaf}} \cap \mathcal{S}_{\texttt{int}} = \emptyset$. In general, each subparse would then take the form $\mathcal{S}_{\texttt{int}} \to (\mathcal{S}_{\texttt{leaf}} \mid \mathcal{S}_{\texttt{int}}) \oplus (\mathcal{S}_{\texttt{leaf}} \mid \mathcal{S}_{\texttt{int}})$. The corresponding general form for the root state is therefore also $\mathcal{S}_{\texttt{root}} \to (\mathcal{S}_{\texttt{leaf}} \mid \mathcal{S}_{\texttt{int}}) \oplus (\mathcal{S}_{\texttt{leaf}} \mid \mathcal{S}_{\texttt{int}})$.

1.4.4 Marker states

Earlier sections have alluded to some of the grammatical problems caused by parsing very short sequences, i.e. the potential confusion of states between the subsets $\mathcal{S}_{\texttt{root}}$, $\mathcal{S}_{\texttt{leaf}}$ and $\mathcal{S}_{\texttt{int}}$. However, we have overlooked another possible solution, namely the fact that we are parsing *complete* sequences. Notionally, each complete sequence is distinguished from a partial sequence (i.e. a subsequence) by having both an implicit start and and an implicit end.

Hence, we could make the sequence start and end explicit by introducing one or more marker states, $\mathcal{S}_{\texttt{mark}}$. Since the parse preserves token ordering, it is unclear whether we need two markers, e.g. $\mathcal{S}_{\texttt{mark}} = \{\triangleleft, \triangleright\}$, or if one marker suffices, e.g. $\mathcal{S}_{\texttt{mark}} = \{\triangleright\}$.

Assuming the former, the imperative sentence "*Go!*" then becomes the sequence $(\square, \texttt{go}, \square)$, where we have taken $\nu_0 = \square$ to be a non-observable pseudo-token corresponding to both start state $\triangleleft$ and end state $\triangleright$. An appropriate parse might then utilise the rules $\texttt{V}_{\texttt{nosubj}} \to \triangleleft \oplus \texttt{V}$ and $\texttt{S} \to \texttt{V}_{\texttt{nosubj}} \oplus \triangleright$. Alternatively, the grammar might be $\texttt{V}_{\texttt{noobj}} \to \texttt{V} \oplus \triangleright$ and $\texttt{S} \to \triangleleft \oplus \texttt{V}_{\texttt{noobj}}$.

However, if we use only one marker, then the sentence "*Go!*" becomes the sequence $(\texttt{go}, \square)$, with parse $\texttt{S} \to \texttt{V} \oplus \triangleright$.

In either case, for longer sequences the issue arises of whether or not the markers are all allowed to be combined into intermediate states. For example, given the sequence $(\square, \texttt{The}, \texttt{cat}, \texttt{sat}, \square)$ with leaf states $(\triangleleft, \texttt{D}, \texttt{N}, \texttt{V}, \triangleright)$, what does it mean if $(\triangleleft, \texttt{D})$ may combine, or $(\triangleleft, \texttt{NP})$? Essentially, a binary combination with a marker corresponds to a unary type-raising rule, but this would only be permitted at the ends of the sequence.

We could, instead, restrict the grammar such that the markers may only be combined after all observable tokens have been parsed. Even in this restricted case, the final effect of type-raising means that we could now separate root states from leaf states and intermediate states, i.e. $\mathcal{S}_{\texttt{root}} \cap \mathcal{S}_{\texttt{leaf}} = \emptyset$ and $\mathcal{S}_{\texttt{root}} \cap \mathcal{S}_{\texttt{int}} = \emptyset$. For a single marker, the overall parse would then take the simpler form $\mathcal{S}_{\texttt{root}} \to (\mathcal{S}_{\texttt{leaf}} \mid \mathcal{S}_{\texttt{int}}) \oplus \triangleright$.

Note that if this final type-raising is generally undesirable, i.e. for $|\mathcal{S}_{\texttt{root}}| > 1$, then we could instead take two marker states $\mathcal{S}_{\texttt{mark}} = \{\triangleleft, \triangleright\}$, but take $\triangleleft$ as the single, unique root state $\mathcal{S}_{\texttt{root}} = \{\triangleleft\}$ with $\mathcal{S}_{\texttt{root}} \cap \mathcal{S}_{\texttt{leaf}} = \emptyset$. Consequently, the root rule simplifies to the form $\triangleleft \to (\mathcal{S}_{\texttt{leaf}} | \mathcal{S}_{\texttt{int}}) \oplus \triangleright$. Such a situation is discussed further in a later section. In practice, however, this is all just syntactic sugar, since the inner parse corresponds to $\mathcal{S}_{\texttt{root}} = \mathcal{S}_{\texttt{leaf}} \cup \mathcal{S}_{\texttt{int}}$ in the orginal formulation.

1.5 Parsing Versus Generation

The rules for the inside-outside algorithm have been defined in a top-down fashion suitable for token generation. In particular, the unary rules take the form $\sigma_i \to \nu_m$ for state $\sigma_i \in \mathcal{S}$ and token $\nu_m \in \mathcal{Y}$. Likewise, the binary rules take the form $\sigma_i \to \sigma_j \oplus \sigma_k$ for $\sigma_i, \sigma_j, \sigma_k \in \mathcal{S}$.

However, we have repeatedly talked about forming a parse tree that spans a sequence $\mathbf{y}$ of tokens. In this context, we need to map each token into the possible leaf states via reversed unary rules like $\nu_m \to \sigma_i$. Similarly, we need to be able to combine adjacent states into a single state via reversed binary rules like $\sigma_j \oplus \sigma_k \to \sigma_i$.

1.5.1 Unary parsing rules

In a previous section, we discussed the possibility of creating a mapping from tokens to leaf states via POS-tagging the training corpus. This mapping is suitable for parsing as it specifies the probabilities

$$P(\nu_m \to \sigma_i) \quad \doteq \quad P(S_t = \sigma_i \mid Y_t = \nu_m) \tag{101}$$

of the bottom-up unary rules. However, recall that for generation we instead need the probabilities

$$P(\sigma_i \to \nu_m) \quad \doteq \quad P(Y_t = \nu_m \mid S_t = \sigma_i) \quad \doteq \quad \beta_{t:t}(i) \tag{102}$$

of the top-down rules. In order to make this conversion, we first note that

$$P(S_t = \sigma_i, Y_t = y_t) \quad = \quad P(S_t = \sigma_i \mid Y_t = y_t)\, P(Y_t = y_t) \quad = \quad P(Y_t = y_t \mid S_t = \sigma_i)\, P(S_t = \sigma_i) \tag{103}$$

Now, in practice it is difficult to compute $P(Y_t)$, since for parsing we need to allow for new tokens not found in the training corpus $\mathbb{Y}$. Hence, we instead replace usages of $\beta_{t:t}(i)$ by

$$\bar{\beta}_{t:t}(i) \;\doteq\; \frac{P(Y_t = y_t \mid S_t = \sigma_i)}{P(Y_t = y_t)} \;=\; \frac{P(S_t = \sigma_i \mid Y_t = y_t)}{P(S_t = \sigma_i)}\,. \tag{104}$$

This trick works because it turns out that the unknown $P(Y_t)$ terms will cancel out when conditioning on the observed sequence $\mathbf{y}$. Additionally, given a POS-tagged corpus, the same trick can also be used in a one-dimensional HMM to sharpen the probabilities from $P(S_t = \sigma_i \mid Y_t = y_t)$ to $P(S_t = \sigma_i \mid \mathbf{Y} = \mathbf{y})$, although technically the latter probabilities are not independent across $t = 1, 2, \ldots, |\mathbf{y}|$.

It should be noted that if we have pre-specified both $P(S_t = \sigma_i \mid Y_t = \nu_m)$ and $P(S_t = \sigma_i)$, then there is no need to estimate the matrix $\mathbf{B}$ of top-down unary rule probabilities. However, if we do not have a POS-tagged corpus, how do we compute these values?

We recall from a previous section that

$$\hat{b}_{m|i} \;=\; \frac{\hat{c}_{im}}{\hat{c}_{i\cdot}} \;=\; \frac{b_{m|i}\,\check{c}_{im}}{\sum_{m'=1}^{|\mathcal{Y}|} b_{m'|i}\,\check{c}_{im'}}\,, \tag{105}$$

where c_{im} is the expected joint count of (σ_i, ν_m), and $b_{m|i}$ is just our original emission probability b_{im} repurposed to explicitly reflect the required conditionality. Hence, if we define $\gamma_i \doteq P(S_t = \sigma_i)$, then we can estimate this probability via

$$\hat{\gamma}_i \;=\; \frac{\hat{c}_{i\cdot}}{\hat{c}_{\cdot\cdot}} \;=\; \frac{\sum_{m=1}^{|\mathcal{Y}|} b_{m|i}\,\check{c}_{im}}{\sum_{i'=1}^{|\mathcal{S}|}\sum_{m'=1}^{|\mathcal{Y}|} b_{m'|i'}\,\check{c}_{i'm'}}\,. \tag{106}$$

Consequently, we may estimate $P(S_t = \sigma_i \mid Y_t = \nu_m)$ via

$$\hat{b}_{i|m} \;=\; \frac{\hat{c}_{im}}{\hat{c}_{\cdot m}} \;=\; \frac{b_{m|i}\,\check{c}_{im}}{\sum_{i'=1}^{|\mathcal{S}|} b_{m|i'}\,\check{c}_{i'm}}\,. \tag{107}$$

Alternatively, if we modify the inside-outside estimation procedure to always compute and retain $\hat{\mathbf{\Gamma}} = [\hat{\gamma}_i]$, then we may subsequently compute

$$\hat{b}_{i|m} \;=\; \frac{\hat{c}_{im}/\hat{c}_{\cdot\cdot}}{\hat{c}_{\cdot m}/\hat{c}_{\cdot\cdot}} \;=\; \frac{\hat{b}_{m|i}\,\hat{\gamma}_i}{\sum_{i'=1}^{|\mathcal{S}|} \hat{b}_{m|i'}\,\hat{\gamma}_{i'}}\,. \tag{108}$$

Clearly, the more fundamental quantity is that of c_{im}, or perhaps

$$b_{im} \;\doteq\; P(S_t = \sigma_i, Y_y = \nu_m) \;=\; \frac{c_{im}}{c_{\cdot\cdot}}\,, \tag{109}$$

from which we may compute $b_{m|i}$, $b_{i|m}$ and $\gamma_i = b_{i\cdot}$. We may also compute

$$\delta_m \;\doteq\; P(Y_t = \nu_m) \;=\; b_{\cdot m}\,, \tag{110}$$

which presumably should remain constant, since this can be computed directly from the corpus.

We observe that

$$\hat{c}_{im} = b_{m|i}\,\check{c}_{im} = \frac{c_{im}}{c_{i\cdot}}\,\check{c}_{im} \tag{111}$$

$$= \frac{c_{im}/c_{\cdot\cdot}}{c_{i\cdot}/c_{\cdot\cdot}}\,\check{c}_{im} = \frac{b_{im}\,\check{c}_{im}}{b_{i\cdot}}\,, \tag{112}$$

and hence

$$\hat{b}_{im} = \frac{\frac{b_{im}\,\check{c}_{im}}{b_{i\cdot}}}{\sum_{i'=1}^{|\mathcal{S}|}\frac{\sum_{m'=1}^{|\mathcal{V}|} b_{i'm'}\,\check{c}_{i'm'}}{b_{i'\cdot}}}\,. \tag{113}$$

1.6 Sequential Dependencies

It was noted in a previous section that although binary rules of the form $\mathtt{A} \to \mathtt{B} \oplus \mathtt{C}$ specify a left-to-right ordering, they do not specify any dependence between states `C` and `B`. This is in contrast to a HMM, which does not have any hierarchical structure but instead models the sequence $\mathbf{y}$ via sequential dependencies between the states $\mathbf{s}$.

In practice, this lack of sequential dependence reveals itself during parsing. For example, since "*water*" may either be a noun `N` or a verb `V`, the imperative sentence "*Water the garden!*" could either have joint states $(\mathtt{V}, \mathtt{D}, \mathtt{N})$ or $(\mathtt{N}, \mathtt{D}, \mathtt{N})$. This was actually my test sentence for validating third-party POS taggers; it turned out that older POS-taggers erroneously interpreted the sentence as the apposition "*Water, the garden!*" (i.e. "*water*" equals "*the garden*"), due to the empirical fact that $P(S_t = \mathtt{N} \mid Y_t = \mathtt{water}) \gg P(S_t = \mathtt{V} \mid Y_t = \mathtt{water})$. This problem seems to have been fixed in modern POS taggers.

For a context-free grammar, the only disambiguation between these choices comes via the relative probabilities of the rules $\mathtt{VP} \to \mathtt{V} \oplus \mathtt{NP}$ versus $\mathtt{NP} \to \mathtt{N} \oplus \mathtt{NP}$. More generally, we are trying to decide between a rule, say $\mathtt{A} \to \mathtt{B} \oplus \mathtt{C}$, having high probability but with a leaf state, say `C`, having low probability given the token it spans, compared to another rule, say $\mathtt{A} \to \mathtt{D} \oplus \mathtt{E}$, having a lower probability but with the leaf state `D` having high probability with respect to the same token. The parser will tend to waste a lot of time trying to match rules to leaf state `D` before it gets around to leaf state `C`.

One way of improving this situation, as noted in a previous section, is to first sharpen the leaf state probabilities by running a HMM across the tokens $\mathbf{y}$, and then computing the (albeit non-independent) posteriors $P(S_t \mid \mathbf{y})$ for $t = 1, 2, \ldots, |\mathbf{y}|$. Another option is to include explicit sequential dependence in the grammar $\mathcal{G}$.

1.6.1 Binary rule backoff

Recall that the binary rule $\mathtt{A} \to \mathtt{B} \oplus \mathtt{C}$ has probability $P(\mathtt{B}, \mathtt{C} \mid \mathtt{A})$. Hence, we can include explicit sequential dependence in the grammar $\mathcal{G}$ by observing that

$$P(\mathtt{B}, \mathtt{C} \mid \mathtt{A}) = P(\mathtt{B} \mid \mathtt{A})\,P(\mathtt{C} \mid \mathtt{B}, \mathtt{A})\,, \tag{114}$$

in general. Note, however, that the latter term's dependence upon both the child state `B` and the parent state `A` means that the grammar has become (weakly) context-sensitive.

For simplicity, we could reduce the new grammar to being context-free by backing-off from $P(\mathtt{C} \mid \mathtt{B}, \mathtt{A})$ to $P(\mathtt{C} \mid \mathtt{B})$, which is equivalent to defining

$$P(\mathtt{B}, \mathtt{C} \mid \mathtt{A}) \doteq P_{\mathtt{left}}(\mathtt{B} \mid \mathtt{A})\, P_{\mathtt{right}}(\mathtt{C} \mid \mathtt{B})\,. \tag{115}$$

This symbollicaly corresponds to a new rule of the form $\mathtt{A} \overset{\mathtt{left}}{\to} \mathtt{B} \overset{\mathtt{right}}{\to} \mathtt{C}$, where $\mathtt{A} \overset{\mathtt{left}}{\to} \mathtt{B}$ is equivalent to the (expansion or substitution or rewrite) rule $\mathtt{A} \to \mathtt{B} \oplus \cdot$ for some placeholder $\cdot$, and $\mathtt{B} \overset{\mathtt{right}}{\to} \mathtt{C}$ is equivalent to the (concatenation) rule $\mathtt{B} \oplus \cdot \to \mathtt{B} \oplus \mathtt{C}$, which is not in Chomsky normal form.

Let us now use the marker states $\mathcal{S}_{\mathtt{mark}} = \{\triangleleft, \triangleright\}$ to denote the start and end, respectively, of a complete sequence. For simplicity, we assume a single, overall root state $\mathcal{S}_{\mathtt{root}} = \{\triangleleft\}$, with probability $P_{\mathtt{root}}(\triangleleft) = 1$. Similarly, we assume a non-observable, pseudo-token $\nu_0 = \square$ corresponding to the leaf end state $\sigma_0 = \triangleright$, such that $P_{\mathtt{token}}(\square \mid \triangleright) = 1$ and $P_{\mathtt{token}}(\square \mid \sigma_i) = 0$ for all $i > 0$.

Under this modified grammar, the derivation of our example sentence is now shown in the figure below.

Note that the horizontal, dashed edges correspond to sequential dependencies, and the slanted, dotted edges correspond to context-sensitive dependencies, which would be dropped for the simplified, context-free grammar.

The joint probability of this derivation is now

$$\begin{aligned}
P(\mathbf{S} = \mathbf{s}, \mathbf{Y} = \mathbf{y}) \quad = \quad & P_{\mathtt{root}}(\triangleleft)\, P_{\mathtt{left}}(S \mid \triangleleft)\, P_{\mathtt{right}}(\triangleright \mid \mathtt{S})\, P_{\mathtt{left}}(\mathtt{NP} \mid \mathtt{S})\, P_{\mathtt{right}}(\mathtt{VP} \mid \mathtt{NP}) && (116)\\
& \times P_{\mathtt{left}}(\mathtt{D} \mid \mathtt{NP})\, P_{\mathtt{right}}(\mathtt{N} \mid \mathtt{D})\, P_{\mathtt{token}}(\mathtt{The} \mid \mathtt{D})\, P_{\mathtt{token}}(\mathtt{cat} \mid \mathtt{N}) && (117)\\
& \times P_{\mathtt{left}}(\mathtt{V} \mid \mathtt{VP})\, P_{\mathtt{right}}(\mathtt{PP} \mid \mathrm{V})\, P_{\mathtt{token}}(\mathtt{sat} \mid \mathtt{V}) && (118)\\
& \times P_{\mathtt{left}}(\mathtt{P} \mid \mathtt{PP})\, P_{\mathtt{right}}(\mathtt{NP} \mid \mathrm{P})\, P_{\mathtt{token}}(\mathtt{on} \mid \mathtt{P}) && (119)\\
& \times P_{\mathtt{left}}(\mathtt{D} \mid \mathtt{NP})\, P_{\mathtt{right}}(\mathtt{N} \mid \mathrm{D})\, P_{\mathtt{token}}(\mathtt{the} \mid \mathtt{D})\, P_{\mathtt{token}}(\mathtt{mat} \mid \mathtt{N})\, P_{\mathtt{token}}(\square \mid \triangleright) && (120)
\end{aligned}$$

Note that the only effect this binary rule backoff has on the inside-outside algorithm is to replace all occurrences of the probability a_{ijk} by $d_{ij}\, e_{jk}$, say, where $P_{\mathtt{left}}(S_{s:r} = \sigma_j \mid S_{s:t} = \sigma_i) \doteq d_{ij}$, and $P_{\mathtt{right}}(S_{r+1:t} = \sigma_k \mid S_{s:r} = \sigma_j) \doteq e_{jk}$.

The estimates for these probability tables are

$$\hat{d}_{ij} \;=\; \frac{\hat{c}_{ij\cdot}}{\hat{c}_{i\cdot\cdot}}\,, \qquad \hat{e}_{jk} \;=\; \frac{\hat{c}_{\cdot jk}}{\hat{c}_{\cdot j\cdot}}\,, \tag{121}$$

respectively. Note that it might be of interest to determine if there are simpler formulae.

1.6.2 Chunking

Chunking is the process of partitioning a sequence into contiguous subsequences, called *chunks*, such that each chunk has self-consistent semantics with respect to the grammar $\mathcal{G}$. For example, an English sentence could be chunked into noun phrases and verb phrases, et cetera.

As an example, consider the sentence "*The black cat purred.*", with *unchunked* ground-truth leaf states $\mathbf{s}_{1:4}^{\mathtt{leaf}} = \langle \mathtt{D}, \mathtt{J}, \mathtt{N}, \mathtt{V} \rangle$, where '$\langle$' denotes the start of a sequence, and '$\rangle$' denotes the end of a sequence. It is apparent that the phrase "*The black cat*" is a noun phrase (i.e. NP), corresponding to the chunk $\mathbf{s}_{1:3}^{\mathtt{leaf}} = (\mathtt{D}, \mathtt{J}, \mathtt{N})$. Consequently, the remaining chunk must be $\mathbf{s}_{4:4}^{\mathtt{leaf}} = (\mathtt{V})$. At the higher

level, the chunking gives parent (or intermediate) states $\mathbf{s}_{1:4}^{\texttt{int}} = (\texttt{NP}, \texttt{VP})$. Finally, at the highest level, chunking gives the grandparent (or root) state(s) $\mathbf{s}_{1:4}^{\texttt{root}} = (\texttt{S})$.

The context-sensistive, top-down derivation of this nested chunking is shown in the figure below.

Observe that the current context (i.e. the parent state) is carried along downward transitions from parent level to child level, then used across horizontal transitions at the same level, and lastly restored along upward transitions from child level to parent level.

Note that to obtain a context-free approximation to this context-sensitive derivation, the parent context is dropped from each transition, and each node must only depend on its head node. Furthermore, since context is now missing, for every transition from parent to child, the parent can no longer be revisited. For the (mostly) top-down derivation, this causes only a slight change, as shown in the figure below.

However, the chunking process described above follows an alternative, (mostly) bottom-up algorithm that is more suitable for sequence parsing, although it can also be used for sequence generation. Under this model, each parent state is only created once its child chunk has been closed. The derivation for our example sentence is shown in the figure below.

The notion of a chunking grammar is pursued further in another notebook.

1.7 References

[1] J. Baker (1979): "*Trainable grammars for speech recognition*", Speech communication papers presented at the 97th meeting of the Acoustical Society of America.

[2] Karim Lari and Steve J. Young (1990): "*The estimation of stochastic context-free grammars using the inside–outside algorithm*", Computer Speech and Language, 4:35–56. (PDF)

[3] K. Lari and S. J. Young (1991): "*Applications of stochastic context-free grammars using the Inside-Outside algorithm*", Computer Speech and Language, 5:237-257.