Computer Speech and Language (1990) **4**, 35–56

The estimation of stochastic context-free grammars using the Inside–Outside algorithm

K. Lari and S. J. Young

Cambridge University Engineering Department, Trumpington Street Cambridge CB2 1PZ, U.K.

Abstract

Using an entropy argument, it is shown that stochastic context-free grammars (SCFG's) can model sources with hidden branching processes more efficiently than stochastic regular grammars (or equivalently HMM's). However, the automatic estimation of SCFG's using the Inside–Outside algorithm is limited in practice by its $O(n^3)$ complexity. In this paper, a novel pre-training algorithm is described which can give significant computational savings. Also, the need for controlling the way that non-terminals are allocated to hidden processes is discussed and a solution is presented in the form of a grammar minimization procedure.

1. Introduction

In recent years, considerable success has been achieved in a variety of speech recognition tasks by using Hidden Markov Models (HMM's) to represent the speech production process (Levinson, Rabiner & Sondhi, 1983; Kubala *et al.*, 1988; Lee & Hon, 1988). Since a HMM is directly equivalent to a stochastic regular grammar which is the weakest class of grammar in the Chomsky Hierarchy, it would seem reasonable to consider the merits of employing the more powerful stochastic context-free grammars (SCFG's) to perform similar tasks. Since a SCFG can be modelled by a multitype Galton–Watson branching process (Harris, 1963; Sevast'yanov, 1970; Sankoff, 1971), the theory of statistical inference of Markov processss can be applied to infer the production probabilities (Anderson & Goodman, 1957).

The potential advantages of SCFG's lie in their ability to capture embedded structure within speech data. Such embedded structure appears explicitly at the word level where context-free grammar's are often used to model task languages. However, the effectiveness of phonological rule systems such as that described by Oshika, Zue, Weeks, Neu & Aubach (1975) suggest that the ability to directly model embedding may be useful at lower levels also. Furthermore, a simple extension of the standard Baum–Welch re-estimation procedure, known as the "Inside–Outside algorithm", enables a SCFG to be estimated from training data in a similar manner to that used in the HMM case (Baker, 1979). A more detailed derivation of this algorithm along with its extension to training on multiple observations is given in Section 2 of this paper.

0885–2308/90/010035+22 $03.00/0

Since Baker first proposed the Inside–Outside algorithm in 1979, there has been surprisingly little subsequent work reported in the literature. The IBM group have reported on its use for estimating the production rule probabilities of a context-free language model (Jelinek, 1985) and the RSRE Speech Research Unit have reported on its use in estimating spelling rules (Dodd, 1988). Other than these, no further applications of the Inside–Outside algorithm are known to us.

The limited interest in the application of the Inside–Outside algorithm in speech and language is probably due to two factors. Firstly, the increased power of CFG's compared to regular grammars results from their ability to model recursive embedding. However, if all of the sentences of a language are assumed to be finite, then any covering CFG can always be reduced to an equivalent regular grammar. Hence, given that English sentences are in practice finite, there would seem to be little need for adopting the complexity of the CFG formalism. Whilst this is certainly true for deterministic grammars, for the stochastic case, the ability of the grammar to determine language membership is less important than its ability to model derivation probabilities accurately. In Section 3 of this paper, we demonstrate that the predictive power of a SCFG as measured by its prediction entropy is greater than that of a regular grammar with the same number of free parameters.

The second barrier to using the Inside–Outside algorithm is its inherent computational complexity which is $O(n^3)$ (see Appendix) both in terms of input string length and the number of grammar symbols. In Section 4, we describe a novel pre-training algorithm which can give significant computational savings and in Section 5 we discuss the need for minimizing the number of non-terminals and controlling the way that they are allocated to hidden processes during grammar estimation. To this end, the use of a grammar minimization procedure within the Inside–Outside algorithm is described. Finally, in Section 6 we briefly comment on our implementation of the algorithm using a parallel transputer array.

2. The Inside–Outside algorithm

The Inside–Outside algorithm assumes that the source can be modelled as a context-free, Hidden Markov Process (Baker, 1979). The algorithm allows the estimated grammar to have an arbitrary degree of ambiguity.

Let $\mathbf{O} = O_1, O_2, \ldots, O_T$ be the observation sequence generated by a stochastic context free grammar G, with rules of the form:

$$i \rightarrow jk \quad \text{and} \quad i \rightarrow m$$

where i,j,k are unique integer numbers corresponding to each of the non-terminal symbols and m is an integer corresponding to a terminal symbol. The matrices of parameters which describe this stochastic context-free grammar are **A** and **B**, where:

$$a[i,j,k] = P(i \Rightarrow jk/G) \tag{1}$$

$$b[i,m] = P(i \Rightarrow m/G). \tag{2}$$

Therefore $a[i,j,k]$ is the probability that the non-terminal symbol i will generate the pair of non-terminal symbols j and k. Similarly, $b[i,m]$ represents the probability that the non-

terminal symbol i will generate a single terminal symbol m. Since any context-free grammar (Chomsky, 1956) may be reduced to Chomsky Normal Form (Chomsky, 1959), these parameters are sufficient to describe any stochastic context-free language.

Note that for consistency the following constraint must always be satisfied:

$$\sum_{j,k} a[i,j,k] + \sum_{m} b[i,m] = 1, \quad \text{for all } i. \tag{3}$$

This constraint simply means that all non-terminals must generate either a pair of non-terminal symbols or a single terminal symbol.

In applying a stochastic context-free grammar we have to address two specific problems, namely that of recognition and training. The recognition problem is concerned with the computation of the probability of the start symbol S generating an observation sequence $\mathbf{O}$ given the grammar G:

$$P(S \overset{*}{\Rightarrow} \mathbf{O}/G).$$

Where an asterisk (*) denotes a derivation sequence consisting of one or more steps. The training problem is concerned with determining a set of grammar rules G given a training sequence $O^{(1)},O^{(2)},\ldots,O^{(Q)}$.

Analogous to the *forward* (α) and *backward* (β) probabilities of conventional Markov model algorithms, we define *inner* (e) and *outer* (f) probabilities to facilitate the analysis of stochastic context-free Markov grammars.

The quantity $e(s,t,i)$ is defined as the probability of the non-terminal symbol i generating the observation $O(s),\ldots,O(t)$:

$$e(s,t,i) = P(i \overset{*}{\Rightarrow} O(s)\ldots O(t)/G) \tag{4}$$

In determining a recursive procedure for calculating e, two cases must be considered:

CASE 1: $(s = t)$

Only one observation is emitted and therefore a transition rule of the form $i \rightarrow m$ applies:

$$\begin{aligned} e(s,s,i) &= P(i \Rightarrow O(s)/G) \\ &= b[i,O(s)]. \end{aligned} \tag{5}$$

CASE 2: $(s \neq t)$

In this case we know that rules of the form $i \rightarrow jk$ must apply since more than one observation is involved. Referring to Fig. 1, it is clear that $e(s,t,i)$ can be expressed as follows:

$$e(s,t,i) = \sum_{j,k} \sum_{r=s}^{t-1} a[i,j,k]e(s,r,j)e(r+1,t,k), \quad \text{for all } i. \tag{6}$$

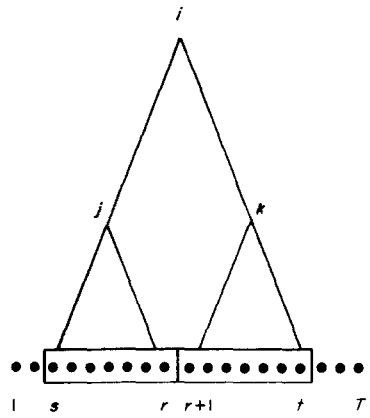

Figure 1. Calculation of inner probabilities.

The quantity e can therefore be computed recursively by determining e for all sequences of length 1, then all sequences of length 2, and so on.

Next we define the *outer* probabilities as follows:

$$f(s,t,i) = P(S \overset{*}{\Rightarrow} O(1)\ldots O(s-1), i, O(t+1)\ldots O(T)/G). \tag{7}$$

$f(s,t,i)$ may be thought of as the probability that in the re write process, i is generated and that the strings not dominated by it are $O(1)\ldots O(s-1)$ to the left and $O(t+1)\ldots O(T)$ to the right (see Fig. 2). In this case, the non-terminal symbol i could be one of two possible settings $j \rightarrow ik$ or $j \rightarrow ki$ as shown in Fig. 3, hence:

$$f(s,t,i) = \sum_{j,k} \left[\sum_{r=1}^{s-1} f(r,t,j)\, a[j,k,i]\, e(r,s-1,k) + \sum_{r=t+1}^{T} f(s,r,j)\, a[j,i,k]\, e(t+1,r,k) \right] \tag{8}$$

$$\text{and} \quad f(1,T,i) = \begin{cases} 1, & \text{if } i = S; \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

After the inner probabilities have been computed bottom-up, the outer probabilities can therefore be computed top-down.

For recognition purposes, the e and the f values can be used to compute the sentence probability as follows:

$$P(S \overset{*}{\Rightarrow} \mathbf{O}/G) = \sum_i e(s,t,i) f(s,t,i) \tag{10}$$

for any $s \leqslant t$. Setting $s = 1$, $t = T$ in (10) gives:

$$\begin{aligned} P(S \overset{*}{\Rightarrow} \mathbf{O}/G) &= \sum_i e(1,T,i) f(1,T,i) \\ &= e(1,T,S). \end{aligned} \tag{11}$$

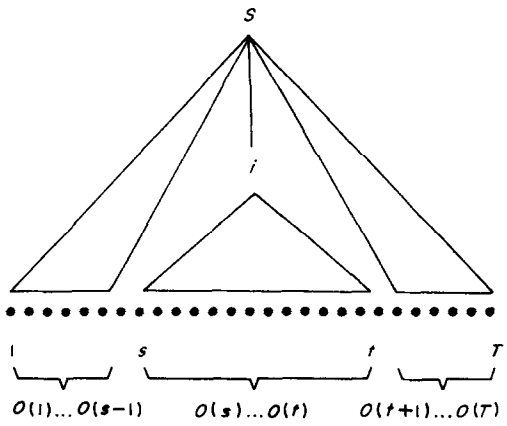

Figure 2. Definition of outer probabilities.

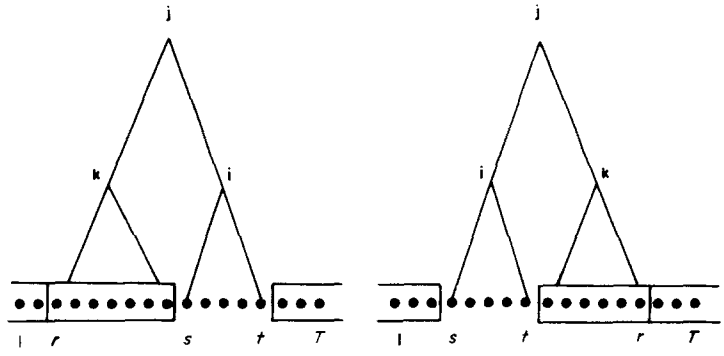

Figure 3. Calculation of outer probabilities.

So $P(S\overset{*}{\Rightarrow}\mathbf{O}/G)$ can be computed from the inner probabilities alone. A similar formula may be derived in terms of outer probabilities by setting $s=t$ in (10):

$$\begin{aligned} P(S\overset{*}{\Rightarrow}\mathbf{O}/G) &= \sum_i e(s,s,i)f(s,s,i) \\ &= \sum_i b[i,O(s)]\, f(s,s,i). \end{aligned} \tag{12}$$

The problem of training a stochastic context-free grammar is more complicated. We start by considering the product:

$$\begin{aligned} e(s,t,i)f(s,t,i) &= P(S\overset{*}{\Rightarrow}\mathbf{O}, i\Rightarrow O(s)\ldots O(t)/G) \\ &= P(S\overset{*}{\Rightarrow}\mathbf{O}/G)\cdot P(i\overset{*}{\Rightarrow}O(s)\ldots O(t)/S\overset{*}{\Rightarrow}\mathbf{O},G) \end{aligned} \tag{13}$$

The last step is due to Bayes's theorem. Let

$$P = P(S \overset{*}{\Rightarrow} \mathbf{O}/G),$$

then from (13):

$$P(i \overset{*}{\Rightarrow} O(s)\ldots O(t)/S \overset{*}{\Rightarrow} \mathbf{O},G) = \frac{1}{P} e(s,t,i) f(s,t,i), \tag{14}$$

therefore

$$P(i \text{ used in derivation}) = \sum_{s=1}^{T} \sum_{t=s}^{T} \frac{1}{P} e(s,t,i) f(s,t,i). \tag{15}$$

Now consider an application of the rule $i \rightarrow jk$ in a derivation. Substitute (6) into (14) to obtain:

$$P(i \overset{*}{\Rightarrow} jk \overset{*}{\Rightarrow} O(s)\ldots O(t)/S \overset{*}{\Rightarrow} \mathbf{O},G) =$$

$$= \frac{1}{P} \sum_{r=s}^{t-1} a[i,j,k] e(s,r,j)\, e(r+1,t,k) f(s,t,i) \quad \text{for all } j,k \text{ and } t > s. \tag{16}$$

Hence from (15) and (16):

$$P(i \rightarrow jk, i \text{ used}) = \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \frac{1}{P} \sum_{r=s}^{t-1} a[i,j,k] e(s,r,j) e(r+1,t,k) f(s,t,i). \tag{17}$$

By definition:

$$a[i,j,k] = P(i \rightarrow jk / i \text{ used}) = \frac{P(i \rightarrow jk, i \text{ used})}{P(i \text{ used})}$$

therefore a re-estimation formula for $a[i,j,k]$ is given by:

$$\hat{a}[i,j,k] = \frac{\frac{1}{P} \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \sum_{r=s}^{t-1} a[i,j,k] e(s,r,j) e(r+1,t,k) f(s,t,i)}{\frac{1}{P} \sum_{s=1}^{T} \sum_{t=s}^{T} e(s,t,i) f(s,t,i)} \tag{18}$$

for all i,j,k.

By similar reasoning a re-estimation formula for $b[i,m]$ may be expressed as follows:

$$\hat{b}[i,m] = \frac{\frac{1}{P}\sum_{t \in O(t)=m} e(t,t,i) f(t,t,i)}{\frac{1}{P}\sum_{s=1}^{T}\sum_{t=s}^{T} e(s,t,i) f(s,t,i)} \tag{19}$$

In a practice, a single observation is insufficient to estimate the parameters of a SCFG accurately. Therefore the above equations must be extended to handle an arbitrary number of observations.

Assume we have a set of Q observations:

$$O \equiv [O^{(1)}, O^{(2)}, \ldots, O^{(Q)}].$$

Let

$$w_q(s,t,i,j,k) = \frac{1}{P_q}\sum_{r=s}^{t-1} a[i,j,k] e_q(s,r,j) e_q(r+1,t,k) f_q(s,t,i)$$

and

$$v_q(s,t,i) = \frac{1}{P_q} e_q(s,t,i) f_q(s,t,i).$$

Assuming that the observations are independent, we can sum the contribution from each of w_q and v_q to the numerators and denominators of Equations (18) and (19) to give

$$\hat{a}[i,j,k] = \frac{\sum_{q=1}^{Q}\sum_{s=1}^{T_q-1}\sum_{t=s+1}^{T_q} w_q(s,t,i,j,k)}{\sum_{q=1}^{Q}\sum_{s=1}^{T_q}\sum_{t=s}^{T_q} v_q(s,t,i)} \tag{20}$$

$$\hat{b}[i,m] = \frac{\sum_{q=1}^{Q}\sum_{t \in O(t)=m} v_q(t,t,i)}{\sum_{q=1}^{Q}\sum_{s=1}^{T_q}\sum_{t=s}^{T_q} v_q(s,t,i)} \tag{21}$$

The Inside–Outside algorithm implements Equations (11), (20) and (21) in an iterative fashion as follows:

1. Choose suitable initial values for the **A** and **B** matrices subject to the constraint specified by (3).

2. REPEAT

   A = . . .{Equation 20}
   B = . . .{Equation 21}
   P = . . .{Equation 11}

   UNTIL change in P is less than a set threshold.

3. Regular versus context-free grammars

The role of a grammar or automata in speech recognition is to reduce the uncertainty inherent in the identification of the input data. One way of measuring this uncertainty is to calculate the entropy of the language generated by the grammar. Two grammars estimated from the same finite size data set, in general, generate different languages with the training data as a common subset (this is strictly because there is never enough example sentences in the training set to capture the complete structure of the language) and therefore have different language entropies. One would expect that the grammar with the lowest language entropy would yield fewest recognition errors. In this paper, entropy is regarded as a measure of the efficiency with which a stochastic grammar models the target language. Entropy therefore provides us with an important tool for comparing regular and context-free grammars, or even two context-free grammars; as far as language modelling is concerned.

Suppose a random variable X can take on values $x_1,x_2,\ldots,x_n$ with respective probabilities $p_1,p_2,\ldots,p_n$. It can be shown that $-\log_2 p_i$ is the amount of information registered when X takes the value x_i (Ross, 1984; Shannon, 1948). The entropy $H(p_1,\ldots,p_n)$ is defined to be the expected amount of information received upon learning the value of X:

$$H(p_1,\ldots,p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{22}$$

The quantity H can also be interpreted as the amount of *uncertainty* that exists as to the value of X. It follows that H is maximized when all n events x_i have equal probabilities of occurrence (i.e. $p_i = 1/n$, for all i), which is intuitively the most uncertain situation. In this case H is equal to $\log_2 n$ (Shannon, 1948).

Entropy can be computed for natural languages (Shannon, 1951), deterministic languages (Kuich, 1970; Levinson, 1985) and stochastic languages (Soule, 1974; Wright, 1988). The entropy of a natural language such as English can be approximated from the statistics of English. The accuracy increases as more and more of the statistics are taken into account (Shannon, 1951). In the case of deterministic languages some assumption must be made about the probability of occurrence of sentences. Two possible assumptions are that sentences are either equiprobable or distributed so as to maximize entropy (Levinson, 1985). However, in the case of stochastic languages no such assumptions are necessary since the probabilities associated with the production rules are known. For a non-ambiguous grammar, a probability can be assigned to each sentence in the language by multiplying the probabilities of the production rules used in its derivation (Wright, 1988). For *ambiguous* grammars, the inner probability defined by Equations (4), (5) and (6) in Section 2 can be used since its method of calculation explicitly takes multiple derivation sequences into account.

Suppose L is a stochastic bounded language, (i.e. L contains a finite number of sentences) and further assume that an accurate estimate of the probability of each sentence $P(S)$ can be computed, it follows that the entropy per sentence is given by

$$H(S) = -\sum_{S \in L} P(S) \log_2 P(S). \tag{23}$$

This is the uncertainty associated with the selection of a sentence from the language. The entropy of the language L is defined to be the quotient of the sentence entropy and the average sentence length:

$$H(L) = -\frac{\Sigma_{S \in L} P(S) \log_2 P(S)}{E\{|S|\}}. \tag{24}$$

The quantity $E\{|S|\}$ is simply the expected number of words per sentence, which is computed by summing individual sentence lengths weighted by their respective probabilities of occurrence:

$$E\{|S|\} = \sum_{S \in L} P(S)|S|. \tag{25}$$

However, many languages of interest are not bounded and may contain an infinite number of sentences. The entropy of such unbounded languages can be estimated by making an *ε-representation* of L in the sense of Booth & Thompson (1973) and using this representation to compute $H_\varepsilon(L)$ via Equation (24). A set of sentences L_1 *strongly* ε-approximates L if

$$L_1 \subseteq L$$

$$\sum_{S \in L_1} P(S) \geqslant 1 - \varepsilon$$

$$P(S/L_1) = P(S/L) \text{ for all } S \in L_1$$

The value of ε itself varies from language to language and depends on how good an approximation is required. L_1 generally contains sentences which have a relatively high probability of occurrence (with no repetition). This follows from the fact that an approximation to L may be obtained by choosing a subset of L consisting of all sentences of length less than or equal to some fixed integer N_ε (Booth & Thompson, 1973):

$$L_1 = \{S | S \in L, \text{ length}(S) \leqslant N_\varepsilon\}.$$

All sentences longer than this upper bound make insignificant contribution to the value of entropy (mainly because longer sentences use more production rules in their derivation, hence, have a smaller probability of occurrence) and therefore may be omitted from L_1. It is this fact that makes Equation (24) computable in practice.

An alternate approach is to compute the sentence probabilities associated with a number of randomly generated sentences, and compute the *empirical entropy* (Wright, 1988) as follows:

$$H_e(L) = -\frac{\Sigma_{k=1}^{K} \log_2 P(S_k)}{\Sigma_{k=1}^{K} |S_k|} \tag{26}$$

The set of K sentences will contain many repetitions of the most likely sentences, but provided K is large enough, it can nevertheless yield a reasonable estimate of entropy.

To illustrate the effectiveness of these measures in comparing stochastic Regular versus stochastic context-free grammars, we have used standard Baum–Welch re estimation and the Inside–Outside algorithm to estimate grammars for the simple language of palindromes defined by

$$L(G)=\{xy|x \text{ is a mirror image of } y\}.$$

For this experiment, a simple grammar for a two-symbol palindrome was constructed (see below), which acted as a source to generate 200 random sentences of the language. These sentences were used as training data for (1) the Inside–Outside algorithm to compute SCFG's and (2) the Baum–Welch algorithm to construct HMM's. Each HMM was given the same number of parameters as the SCFG, (it is unfair to compare a five non-terminal SCFG with a five-state HMM, since a five non-terminal SCFG uses more parameters to express its grammar). Generally, an N-non-terminal, M-terminal SCFG uses N^3+NM parameters and a K-state, M-terminal HMM uses $K^2+(M+2)K$ parameters (including transition probabilities to and from confluent states). Therefore, the following condition should hold for comparable models:

$$K^2+(M+2)K \approx N^3+NM. \tag{27}$$

The source grammar used to generate the data for this experiment is shown in Fig. 4. It is a five non-terminal, two terminal SCFG with eight non-zero parameters. We used the

Source SCFG

$S\rightarrow A\ C$	(0·3)	$A\rightarrow a$	(1·0)
$B\ D$	(0·3)	$B\rightarrow b$	(1·0)
$A\ A$	(0·2)	$C\rightarrow S\ A$	(1·0)
$B\ B$	(0·2)	$D\rightarrow S\ B$	(1·0)

Estimated SCFG

$S\rightarrow A\ B$	(0·47)		
$D\ C$	(0·45)		
$B\ B$	(0·02)		
$D\ D$	(0·07)		
$A\rightarrow B\ S$	(0·60)	$C\rightarrow S\ D$	(0·66)
b	(0·40)	a	(0·34)
$B\rightarrow$ b	(1·00)	$D\rightarrow a$	(1·00)

Estimated regular grammar

$S\rightarrow aA_4|bA_9$

$A_1\rightarrow aA_2|bA_2|aA_4|a|b$

$A_2\rightarrow aA_1|bA_1|aA_3|aA_5|aA_6|aA_7$

$A_3\rightarrow aA_4|bA_9$

$A_4\rightarrow aA_3|bA_1|aA_3$

$A_5\rightarrow aA_4|aA_8|bA_{10}|b$

$A_6\rightarrow aA_8|bA_{10}$

$A_7\rightarrow aA_9|bA_{10}$

$A_8\rightarrow aA_1|bA_1|aA_6|aA_7$

$A_9\rightarrow aA_8|bA_{10}$

$A_{10}\rightarrow aA_4|bA_9|b$

Figure 4. Grammar estimated for the language of two-symbol palindromes.

same number of non-terminals and terminals for the estimated SCFG but allowed all parameters to be non-zero (135 in total). Using (27) this fixed the estimated HMM to have 10 states. The final SCFG obtained using the Inside–Outside algorithm and the regular grammar derived from the estimated HMM are also shown in Fig. 4. Inspecting these grammars, it can be seen that although the estimated SCFG is not identical to the source it is nevertheless equivalent in the sense that it generates palindromes and only palindromes.

Table I shows the entropies of the languages associated with both the SCFG and HMM estimated using (24) and (26), also shown in the table is the entropy of the source which for this simple grammar can be calculated exactly. From the table, it is clear that the entropy of the language associated with the estimated SCFG is much closer to the source than that of the HMM. Thus, for this example at least, the SCFG is able to model the source more efficiently than the comparable HMM even though the source data is finite.

Some additional simple tests confirmed this further. Firstly, using a set of 100 distinct sentences half of which were palindromes, the SCFG classified all 100 sentences correctly whereas the HMM classified 53 correctly. Secondly, using both the estimated grammars as generators, all sentences generated by the SCFG were palindromes whereas only 56% of those generated by the HMM were palindromes.

TABLE I. Entropies of source and estimated languages

	$H(L)$	$H_\varepsilon(L)$*	$H_e(L)$*
Source	0·985	—	—
Estimated SCFG	—	0·995	0·989
Estimated HMM	—	1·196	1·402

* Estimated from 1000 generated sentences.

4. Pre-training

As with standard HMM re-estimation, there is no guarantee that the Inside–Outside algorithm will converge towards a global optimum. Furthermore, as noted in the introduction, it is an $O(n^3)$ algorithm. Hence, it is vital to start off the algorithm with good initial estimates.

In this section, a pre-training algorithm will be described which has proved effective in both reducing the number of re-estimation cycles required and facilitating the generation of a good final model.

The key idea is that instead of starting with random **A** and **B** matrices, we first use the Baum–Welch algorithm to obtain a set of regular grammar rules, and use these (with some modifications to be explained later) as a starting point for the Inside–Outside algorithm.

Suppose for a given data set, we train a HMM, with N states, and an explicit starting state S and a final state E (note that in order to describe a HMM in terms of regular grammar rules, we need two confluent states S and E, otherwise we will not obtain strings containing only terminal symbols). The transition matrix **A**, the emission matrix

B, the initial state probability vector π and the final state probability vector **F** can be used to define a Chomsky Normal Form grammar as follows:

(a) For each b_{jk}, define $Y_j \rightarrow k$ with probability b_{jk}.

(b) For each a_{ij}, define $X_i \rightarrow Y_i X_j$ with probability a_{ij}.

(c) For each π_j, define $S \rightarrow X_j$ with probability π_j.

(d) For each F_i, define $X_i \rightarrow Y_i$ with probability F_i.

Rules of type (a)–(d) allow a HMM to be converted into a stochastic grammar. However, note that the rules resulting from (a) and (b) are in Chomsky Normal Form whereas those resulting from (c) and (d) are not. Since, the Inside–Outside algorithm requires us to have the rules in Chomsky Normal Form, some further conversion is necessary:

(e) For each rule of the form:

$$S \rightarrow X_j \text{ with probability } \pi_j$$

locate all rules with X_j on their left and then substitute the right-hand side of each rule for X_j in the original rule, i.e. if

$$X_j \rightarrow Y_j X_l \text{ with probability } a_{jl}$$

then create a new rule:

$$S \rightarrow Y_j X_l \text{ with probability } \pi_j a_{jl}.$$

(f) For each rule of the form:

$$X_i \rightarrow Y_j \text{ with probability } F_i$$

substitute all right-hand side of rules with Y_j on their left-hand side, for Y_j, i.e. if

$$Y_j \rightarrow k \text{ with probability } b_{ik}$$

then create a new rule

$$X_i \rightarrow k \text{ with probability } b_{ik} F_i.$$

(e) and (f) ensure that all the rules are in the required Chomsky Normal Form.

The above algorithm allows a HMM to be transformed into an equivalent Chomsky Normal Form grammar. If this grammar was used directly as the starting point for the Inside–Outside algorithm, all of the zero parameters would remain zero thus forcing the grammar to remain regular. To prevent this, all parameters of the initial grammar are raised above some floor value and then the matrices are renormalized. This allows the re-

estimation procedure to transform the grammar into a full SCFG. The floor value we use is proportional to the number of non-zero parameters (typically around 0·01). To illustrate the operation of this pre-training algorithm, Fig. 5 shows log P plotted against iteration number for the palindrome estimation problem described earlier. Curves are shown for both random initialization and the HMM-based pre-training described above. The substantially more rapid convergence in the pre-trained case is evident. In this case, the reduction in total compute time achieved by using the pre-training technique is approximately 40%. Similar savings have also been achieved across a range of other problems.

As an aside, it is interesting to note the shape of the curves in Fig. 5 which appear to be typical in that there seem to be two distinct learning phases. We have no explanation for this at present but its occurrence means that care needs to be taken to ensure proper convergence of the algorithm.

The SCFG estimated for the palindrome problem using pre-training is shown in Fig. 6 and its entropy is compared with the randomly initialized case in Table II. As can be seen, the estimated grammar using pre-training is again equivalent to the source grammar and there is no statistically significant difference between its entropy and that obtained earlier. Thus, the entropy measurements suggests that the pre-trained SCFG is as efficient as the randomly initialized SCFG. In fact, more direct comparisons of the two grammars suggests that the pre-trained SCFG is actually a closer estimate to the source. For example, Table III shows the probabilities of generating sentences of length 4 or less. For this case, the pre-trained grammar is clearly closer to the source grammar.

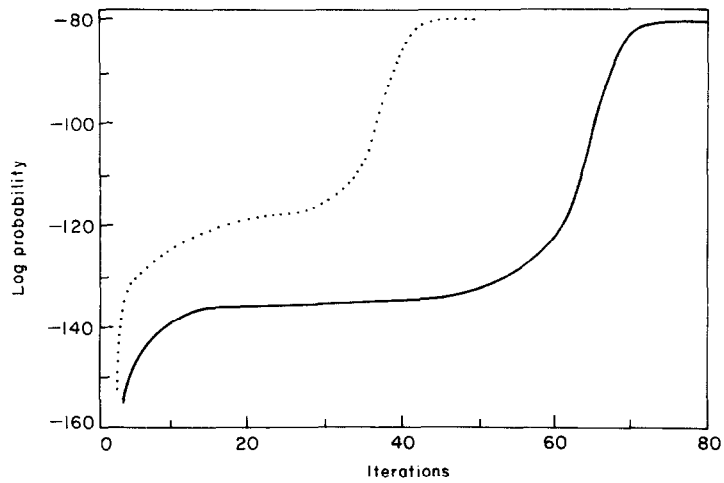

Figure 5. Learning rate for pre-trained (·····) and random (———) SCFGs.

$S \rightarrow A\ C$	(0·35)	$B \rightarrow D\ S$	(0·72)
$B\ D$	(0·41)	$B \rightarrow b$	(0·28)
$C\ C$	(0·17)	$C \rightarrow a$	(1·00)
$D\ D$	(0·07)	$D \rightarrow b$	(1·00)
$A \rightarrow C\ S$	(1·00)		

Figure 6. SCFG obtained with pre-training.

TABLE II. Effects on entropy due to pre-training

	$H(L)$	$H_s(L)$*	$H_e(L)$*
Source	0·985	—	—
SCFG (random)	—	0·995	0·989
SCFG (pre-trained)	—	0·981	0·994

* Estimated from 1000 generated sentences.

TABLE III. Comparison of sentence length distributions

	$P(\|S\| \leqslant 4)$
Source	0·64
Pre-trained	0·59
Random	0·46

5. The use of grammar minimization during estimation

The palindrome example of Sections 3 and 4 was drawn from the alphabet $\{a,b\}$ which is the simplest of its kind. More complex palindromes can be formed if the alphabet set is expanded to contain more terminal symbols. However, increasing the number of terminals requires an increase in the number of non-terminals, which in turn leads to a more complex grammar. In order to formulate grammar rules for the language of M-symbol palindromes a minimum of $2M+1$ non-terminal symbols is required; one start-symbol, M non-terminals to describe the hidden process and M non-terminals to describe the observable process. The grammar shown in Fig. 7 generates palindromes drawn from the alphabet $\{a,b,c\}$. Where S is the start-symbol, B,D,F describe the hidden process and A,C,E describe the observable process.

A stochastic version of the above grammar was used to generate 400 example sentences of the language. These sentences acted as training data for the Inside–Outside algorithm, which produced the rules shown in Fig. 8. In this case, the Inside–Outside algorithm failed to produce a satisfactory set of grammar rules. Only the terminal "a" is generated in the desired self-embedded fashion, "b" and "c" are generated sequentially. The symbol "a" acts as a *dominant* symbol and freezes some of the non-terminals. For example, non-terminals C, D and F play the same role; only one of them is required for the generation of "a"s. A similar experiment was performed whereby the number of non-terminals was increased from 7 to 12. The computed grammar rules were slightly better but still not very satisfactory, symbols "a" and "b" were now generated in a self-embedded fashion and only "c" was generated sequentially. Increasing the number of non-terminals to 18 finally resulted in a set of production rules, which correctly modelled the language of palindromes and generated only sentences of that language.

We have found that the above experimental results are typical over a wide range of problems. In order to ensure that sufficient non-terminals are available to model all of the hidden processes within the source, it is necessary to have many more non-terminals than are theoretically necessary. Given the $O(n^3)$ complexity of the algorithm, this aspect of the Inside–Outside algorithm renders it computationally intractable for realistic problems.

$S \rightarrow A\ B$	$B \rightarrow S\ A$
$C\ D$	$D \rightarrow S\ C$
$E\ F$	$F \rightarrow S\ E$
$A\ A$	$A \rightarrow a$
$C\ C$	$C \rightarrow b$
$E\ E$	$E \rightarrow c$

Figure 7. Source grammar for three-symbol palindromes.

$S \rightarrow A\ A$	$B \rightarrow S\ C$
$C\ B$	$S\ D$
$C\ C$	$S\ F$
$D\ B$	a
$D\ C$	$C \rightarrow a$
$D\ D$	$D \rightarrow a$
$E\ E$	$E \rightarrow S\ E$
$F\ B$	b
$A \rightarrow S\ A$	$F \rightarrow a$
c	

Figure 8. Grammar inferred for the three-symbol palindrome.

From the above, it is clear that the Inside–Outside training algorithm would be improved if redundant and/or useless symbols could be detected and eliminated. This would in effect involve attempting to minimize the grammar. Although the lack of any formal procedure for testing for the equivalence of two context-free grammars prevents an exact minimization technique to be defined, the use of approximate techniques may nevertheless be of value. We have therefore experimented with extended versions of the Inside–Outside algorithm in which the conventional re-estimation is interleaved with an approximate Grammar Minimization (GM) procedure.

A GM procedure may be incorporated into the Inside–Outside algorithm in two ways. Firstly, the Inside–Outside could start with some fixed maximum number of non-terminals, and a grammar minimization procedure applied periodically to detect and eliminate *useless* and *redundant* symbols (e.g. for the above example the algorithm would start with 20 non-terminals and end up with 7). However, this approach would be computationally intractable if the grammar used many non-terminals. A second and more practical approach would be to start with the desired number of non-terminals (i.e. 7 in the example above) and apply a grammar minimization procedure periodically (or when the change in log-probability falls below a certain threshold) to detect and *reallocate* redundant symbols. This approach is described further in Section 5.2. Before that however, it is worth noting that for simple grammars like palindromes, the grammar minimization process may be avoided by introducing suitable constraints. We will discuss this first before dealing with the more general case.

5.1 *Constraining the Inside–Outside algorithm*

In the experiments described so far, the Inside–Outside algorithm was given complete freedom in formulating the grammar rules, and it was this freedom that encouraged "greedy symbols" to take too many non-terminals. One method of decreasing this freedom is to introduce *constraints* (which is so commonly done with HMM's, e.g. the use of left to right models). A typical constraint might be to specifically allocate a non-terminal to each terminal symbol and then force the remaining non-terminals to model

the hidden branching process. This may be done by setting certain parameters to zero. For example, allocating the non-terminal A to terminal symbol a in the palindrome case is done by assigning a probability of 1·0 to that rule (therefore 0·0 to the remaining derivation parameters for A). Figure 9 shows such a constraint along with the resulting grammar inferred by the Inside–Outside algorithm for the language of three symbol palindromes where it can be seen that the required embedding has been achieved now for all three terminal symbols without having a large excess of non-terminals. This type of constraint appears to work well for simple languages, but it is clearly not suitable for realistic problems that use many terminals. For example, recognizing speech encoded by a 64-codeword Vector Quantizer requires 64 non-terminals dedicated entirely to generating the terminal symbols. With regard to the complexity of the algorithm, this type of constraint would be completely infeasible for speech. The next section therefore describes an approach based on grammar minimization which is practical for larger scale problems.

(a)

$S \rightarrow a\|b\|c$	(0·0)
$A \rightarrow a$	(1·0)
$B \rightarrow b$	(1·0)
$C \rightarrow c$	(1·0)
$D \rightarrow a\|b\|c$	(0·0)
$E \rightarrow a\|b\|c$	(0·0)
$F \rightarrow a\|b\|c$	(0·0)

(b)

$S \rightarrow A\ A$	$A \rightarrow a$
$B\ B$	$B \rightarrow b$
$C\ C$	$C \rightarrow c$
$D\ A$	$D \rightarrow A\ S$
$E\ B$	$E \rightarrow B\ S$
$F\ C$	$F \rightarrow C\ S$

Figure 9. (a) Constraining the algorithm; (b) grammar obtained with constraint.

5.2 *A Grammar Minimization algorithm*

This section outlines how we have incorporated a grammar minimization process into the Inside–Outside algorithm. The main aim of this extension is to detect and reallocate redundant non-terminals which are dominated the "greedy" symbols, that is, the symbols which have an excessive number of parent non-terminals. We have developed the following algorithm to perform this task:

1. Locate all redundant non-terminal symbols. This is done in a bottom-up fashion by first examining the terminal symbols that are generated by more than one non-terminal, and then recursively searching backwards through the possible derivation sequences recording all rules which are redundant. The redundant non-terminals corresponding to a particular greedy terminal are then replaced by a single non-terminal throughout the grammar thereby transforming the redundant grammar rules into several occurrences of the same rule. These are then collapsed into a single rule by adding their rule probabilities. This step results in the availability of one or more *free* non-terminals which were previously dominated by greedy symbols.

2. Locate non-terminals (other than S) that successfully generate parts of a sentence. These are assumed to be the remaining non-terminals involved in the generation of greedy symbols since we know that the hidden processes relating to these symbols have been more than adequately covered by non-terminals. The matrices of parameters associated with these non-terminals are then fixed i.e. they are excluded from the parameter updates defined by steps (3) and (4).

3. Randomize the matrices corresponding to the remaining free non-terminals with the following constraints:

(i) The probability of any free non-terminal generating a greedy symbol is set to zero and

(ii) rows and columns corresponding to the non-terminals located by step (2) are also set to zero.

These conditions reallocate the free non-terminals to the stochastic processes not involved in the generation of the greedy symbols.

4. Randomize those rows and columns of the remaining non-terminals (which describe the under-represented stochastic processes) corresponding to the free non-terminals.

To illustrate the effects of the above GM process, consider the grammar inferred by the Inside–Outside algorithm for the language of three-symbol palindromes (see Fig. 8). After applying step (1), non-terminals D and F are set "free" and the grammar is reduced to that of Fig. 10. At the second step of GM non-terminals B and C are fixed since they are both involved in the generation of greedy symbol a. Steps (3) and (4) randomize parts of the matrices corresponding to non-terminals S, A, D, E and F, such that D and F are re-allocated to S, A and E for the generation of the non-greedy symbols b and c.

$S \rightarrow A\ A$
$\quad C\ B$
$\quad C\ C$
$\quad E\ E$
$A \rightarrow S\ A$
$\quad c$
$B \rightarrow S\ C$
$\quad a$

$C \rightarrow a$
$D \rightarrow ???$
$E \rightarrow S\ E$
$\quad b$
$F \rightarrow ???$

Figure 10. Elimination of D's and F's.

$S \rightarrow A\ A$
$\quad C\ B$
$\quad C\ C$
$\quad D\ E$
$\quad E\ E$
$\quad F\ E$
$A \rightarrow S\ A$
$\quad c$
$B \rightarrow S\ C$
$\quad a$

$C \rightarrow a$
$D \rightarrow E\ S$
$E \rightarrow b$
$F \rightarrow E\ S$

Figure 11. Grammar for three-symbol palindromes after the first application of the grammar minimization algorithm.

$S \rightarrow A\ A$	$B \rightarrow S\ C$
$C\ B$	a
$C\ C$	$C \rightarrow a$
$D\ E$	$D \rightarrow E\ S$
$E\ E$	$E \rightarrow b$
$F\ A$	$F \rightarrow A\ S$
$A \rightarrow c$	c

Figure 12. Grammar for three-symbol palindromes after two applications of the grammar minimization algorithm.

After 34 iterations the Inside–Outside algorithm converged on the new set of rules shown in Fig. 11. By examining the grammar of Fig. 11 it is clear that non-terminals D and F are now doing the same job. However, the grammar has improved since non-terminals D (or F) and E together now generate the b's in the desired self-embedding fashion. The grammar minimization algorithm was applied again by replacing all F's by D's and keeping the parameters associated with D and E (as well as B and C) fixed. The non-terminal F was then reallocated to S and E to cover the generation of c's.

After another 22 iterations the Inside–Outside algorithm finally produced the "correct" grammar as shown in Fig. 12.

So far, our testing of the above GM procedure has been limited to "toy" examples. Although the rules we use for identifying "useful" and "useless" symbols are currently somewhat "ad hoc", our result clearly show that it is both possible and profitable to attempt to redistribute the function of non-terminals during the re-estimation process. Indeed, for realistic tasks, a procedure such as this appears to be mandatory.

6. Implementation

Implementation of the Inside–Outside algorithm on a conventional serial computer is practical only if the number of non-terminals is small (less than eight or so on a MicroVAX II workstation). As noted earlier, the Inside–Outside algorithm is $O(n^3)$ whereas the Baum–Welch algorithm is $O(n^2)$. The graph below (Fig. 13) shows how the time taken per iteration varies with the number of non-terminals in (arbitrary) MicroVAX II units.

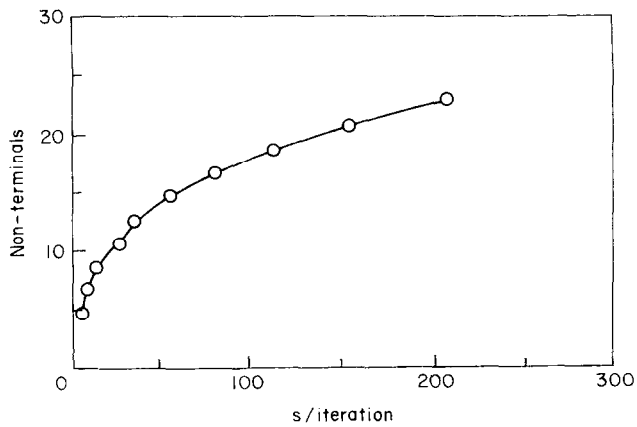

Figure 13. The $O(n^3)$ complexity of the Inside–Outside algorithm.

One can see that when a large number of non-terminals is used, we have an n^3 relationship, but for a small number of non-terminals it approaches n^2 since the computation overhead tends to dominate for small n.

Although the Inside–Outside algorithm is computationally rather too complex for a conventional serial-computer, it is fortunately suitable for parallel implementation. Since training time is proportional to the number of data strings in the training set, it is simple to split the problem into smaller sub-problems by partitioning the training data. We have implemented the Inside–Outside algorithm on the ParSiFal (Knowles, 1986) transputer array which consists of a reconfigurable array of 64 transputers linked in a serial fashion as shown in Fig. 14.

The array is attached to a SUN host computer via a control board, which itself is a transputer. The training data is split into 64 subsets and each transputer works independently on its own data. All 64 transputers send their contributions to the control transputer which computes the updated parameter set and transmits it down the chain to all the other transputers. This method of implementation is 100 times faster than an optimized C (serial) version running on a MicroVAX II workstation. Using the transputer array, we are able to tackle problems with around 20–30 non-terminals and around 60–100 terminals. All of our algorithms are written in C and are configured to run both on a conventional workstation and the parallel transputer array. So far, using double length floating point arithmetic, we have not encountered underflow problems. If such problems do occur, we will switch to the logarithmic representation developed by Kingsbury & Rayner (1971) rather than attempt explicit scaling.

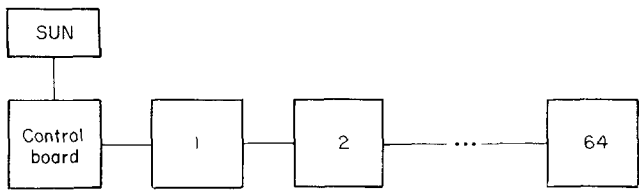

Figure 14. The ParSiFal transputer array.

7. Conclusions

This paper has reviewed the Inside–Outside algorithm and described various problems that arise in its practical application. As with all iterative procedures, it is highly desirable to start the Inside–Outside algorithm with a good initial estimate. We have shown that it is possible to derive such an estimate by using standard HMM estimation in a "pre-training" phase. Not only does this seem to lead to good final solutions, it also typically reduces the number of iterations needed by half. We have also drawn attention to the need for controlling the way non-terminal symbols are allocated to the underlying hidden process during re-estimation. We have found that typically a three-fold excess of non-terminals is needed to ensure an adequate distribution of non-terminals to each of the hidden processes and that given the $O(n^3)$ complexity of the algorithm, this is potentially crippling. Fortunately, however, we have discovered that this problem can be avoided by interleaving the Inside–Outside re-estimation with a Grammar Minimization procedure which reallocates useless and/or redundant symbols to under-represented stochastic processes.

So far the bulk of our work on the Inside–Outside algorithm has been restricted to "toy" problems, that is, estimating models from data generated from a known source. Our real interest, however, is in modelling speech data at the subword level. Whilst it is not yet known whether SCFG's can in practice provide better subword models than HMM's, we have shown in this paper, via an entropy argument, that where branching processes do exist in data, SCFG's trained by the Inside–Outside algorithm are superior. We are therefore currently building SCFG models trained from real speech data to pursue this further.

References

Aho, A. V., Hopcroft, J. E. & Ullman, J. D. (1983). *Data Structures and Algorithms*. Addison Wesley, Reading, MA.

Anderson, T. W. & Goodman, L. A. (1957). Statistical inference about Markov chains. *Annals of Mathematical Statistics*, **28**, 89–100.

Bahl, L. R., Jelinek, F. & Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Matching and Machine Intelligence*, **5**, 179–190.

Baker, J. K. (1979). Trainable grammars for speech recognition. *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America* (D. H. Klatt and J. J. Wolf, eds), pp. 547–550.

Booth, T. L. & Thompson, R. A. (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers*, **22**, 442–450.

Chomsky, N. (1956). Three models for the description of languages. *IRE Transactions on Information Theory*, **2**, 113–124.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, **2**, 137–167.

Dodd, L. (1988). Grammatical inference for automatic speech recognition: an application of the Inside/Outside algorithm and the spelling of English words. *Proceedings of the 7th FASE Symposium*, Edinburgh, pp. 1061–1068.

Harris, T. E. (1963). *The Theory of Branching Processes*. Springer-Verlag, Berlin.

Jelinek, F. (1985). Markov source modeling of text generation. NATO Advanced Study Inst. *Impact of Processing Techniques on Communication*, pp. 569–598. Martinus Nijhoff, Amsterdam.

Kingsbury, N. G. & Rayner, P. J. W. (1971). Digital filtering using logarithmic arithmetic. *Electronics Letters*, **7**, 56–58.

Knowles, A. E. (1986). Specification for T-Rack, ParSiFal Document PSF/MU/87/AEK/3, Department of Computer Science, University of Manchester.

Kubala, F., Chow, Y., Derr, A., Feng, M., Kimball, O., Makhoul, J., Price, P., Rohlicek, J., Roucos, S., Schwartz, R. & Vandegrift, J. (1988). Continuous speech recognition results of the BYBLOS system on the DARPA 1000-word resource management database. *IEEE ICASSP*, **1**, 291–294.

Kuich, W. (1970). On the entropy of context-free languages. *Information and Control*, **16**, 173–200.

Lee, K. F. & Hon, H. W. (1988). Large-vocabulary speaker-independent continuous speech recognition using HMM. *IEEE ICASSP*, **1**, 123–126.

Levinson, S. E., Rabiner, L. R. & Sondhi, M. M. (1983). An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *Bell System Technical Journal*, **62**, 1035–1074.

Levinson, S. E. (1985). Structural methods in automatic speech recognition. *Proceedings of the IEEE*, **73**, 1625–1650.

Oshika, B. T., Zue, V. W., Weeks, R. V., Neu, H. & Aubach, J. (1975). The role of phonological rules in speech understanding research. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **23**, 104–112.

Ross, S. (1984). *A First Course in Probability*. Macmillan, New York.

Sankoff, D. (1971). Branching processes with terminal types: applications to context-free grammars. *Journal of Applied Probability*, **8**, 233–240.

Sevast'yanov, B. A. (1970). Theory of branching processes. *Progress in Mathematics*, **7**, 1–51.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, **27**, 379–423.

Shannon, C. E. (1951). Prediction and entropy of printed English. *Bell System Technical Journal*, **29**, 50–64.

Soule, S. (1974). Entropies of probabilistic grammars. *Information and Control*, **25**, 57–74.

Wright, J. H. (1988). Linguistic modelling for application in speech recognition. *Proceedings of the 7th FASE Symposium*, Edinburgh pp. 391–398.

Appendix: algorithmic complexity of the Inside–Outside algorithm

The execution time $T(n)$ of a program as a function of input size n is said to be $O(n^k)$ if there exist positive constants c and n_0 such that

$$T(n) \leqslant cn^k,$$

for $n \leqslant n_0$. We now seek to show that for the Inside–Outside training algorithm $T(n) = O(n^3)$. In the following, the "rule of sums" will be used to calculate the running time of a sequence of program steps. Suppose $T_1(n)$ and $T_2(n)$ are the running times for two program fragments P_1 and P_2, and that $T_1(n)$ is $O(f(n))$ and $T_2(n)$ is $O(g(n))$. Then $T_1(n) + T_2(n)$, the running time of P_1 followed by P_2 is $O(\max(f(n),g(n)))$ (Aho, Hopcroft & Ullman, 1983).

It is clear that the most complex part of the Inside–Outside algorithm is the computation of the inner (and the outer) probabilities. Therefore, by the "rule of sums" their running time will determine the complexity of the whole training algorithm. The inputs to the program are the number of non-terminal symbols N, the number of terminal symbols M and observation sequence(s) O with length T. Our aim is to determine the complexity of the algorithm in terms of the input parameters N and T.

The inner probabilities defined by Equation (6) is computed in two steps. First, all substrings of length 2 are considered. This eliminates the inner summation of (6) and simplifies the equation as follows:

$$e(s,t,i) = \sum_{i=1}^{N} \sum_{s=1}^{T-1} \sum_{j,k} a[i,j,k]b[j,O(s)]b[k,O(s+1)]$$

The product of the three probability terms is taken to be $O(1)$. The inner summation with indices j and k which range over all N non-terminals and therefore is executed N^2 times. The middle summation ranges over the length (T) of the observation and is executed $T-1$ times. The outer summation repeats the whole process for every N making the running time $O(N^3)$ in non-terminals and $O(T)$ in length of the observation sequence. The analysis for all the remaining substrings (length greater than 2) is more complex. The inner probabilities are now computed from:

$$e(s,t,i) = \sum_{l=2}^{T-1} \sum_{i=1}^{N} \sum_{s=1}^{T-1} \sum_{j,k} \sum_{r=s+1}^{t-2} a[i,j,k]e(s,r,j)e(r+1,t,k), \quad t = s+l.$$

The inner summation is executed $t-s-2$ times or simply $l-2$ times. By counting the number of times each loop (or summation) is executed the running time may be expressed as follows:

$$\sum_{i=2}^{T-1} N^3(T-l)(l-2) = N^3 \left\{ -\sum_{l=2}^{T-1} l^2 + (T+2) \sum_{i=2}^{T-2} l - 2T(T-3) \right\}$$

$$= N^3\{-\tfrac{1}{6}T(T-1)(T-\tfrac{1}{2}) + 1 + \tfrac{1}{2}T(T-1)(T+2) - (T+2) - 2T(T-3)\}$$

which is $O(N^3)$ and $O(T^3)$. By applying the "rule of sums", the computation of inner probabilities is cubic both in terms of number of non-terminals and in terms of length of the observation sequence. The running time of the outer probability computation can be computed in an identical fashion (and it leads to the same conclusion).