An Algorithm for Estimating the Parameters of Unrestricted Hidden Stochastic Context-Free Grammars

Julian Kupiec

Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

ABSTRACT

A new algorithm is presented for estimating the parameters of a stochastic context-free grammar (SCFG) from ordinary unparsed text. Unlike the Inside/Outside (I/O) algorithm which requires a grammar to be specified in Chomsky normal form, the new algorithm can estimate an arbitrary SCFG without any need for transformation. The algorithm has worst-case cubic complexity in the length of a sentence and the number of nonterminals in the grammar. Instead of the binary branching tree structure used by the I/O algorithm, the new algorithm makes use of a trellis structure for computation. The trellis is a generalization of that used by the Baum-Welch algorithm which is used for estimating hidden stochastic regular grammars. The paper describes the relationship between the trellis and the more typical parse tree representation.

INTRODUCTION

This paper describes an iterative method for estimating the parameters of a hidden stochastic context-free grammar (SCFG). The "hidden" aspect arises from the fact that some information is not available when the grammar is trained. When a parsed corpus is used for training, production probabilities can be estimated by counting the number of times each production is used in the parsed corpus. In the case of a hidden SCFG, the characteristic grammar is defined but the parse trees associated with the training corpus are not available. To proceed in this circumstance, some initial probabilities are assigned which are iteratively re-estimated from their current values, and the training corpus. They are adjusted to (locally) maximize the likelihood of generating the training corpus. The EM algorithm (Dempster, 1977) embodies the approach just mentioned; the new algorithm can be viewed as its application to arbitrary SCFG's. The use of unparsed training corpora is desirable because changes in the grammar rules could conceivably require manually re-parsing the training corpus several times during grammar development. Stochastic grammars enable ambiguity resolution to performed on the rational basis of most likely interpretation. They also accommodate the development of more robust grammars having high coverage where the attendant ambiguity is generally higher.

Previous approaches to the problem of estimating hidden SCFG's include parsing schemes in which all derivations of all sentences in the training corpus are enumerated (Fujisaki et al., 1989; Chitrao & Grishman, 1990)). An efficient alternative is the Inside/Outside (I/O) algorithm (Baker, 1979) which like the new algorithm, is limited to cubic complexity in both the number of nonterminals and length of a sentence. The I/O algorithm requires that the grammar be in Chomsky normal form (CNF). The new algorithm has the same complexity, but does not have this restriction, dispensing with the need to transform to and from CNF.

TERMINOLOGY

The training corpus can be conveniently segmented into sentences for puposes of training; each sentence comprising a sequence of words. A typical one may consist of $Y+1$ words, indexed from 0 to Y:

$$(w_0, w_1, w_2 ... w_Y)$$

The lookup function $W(y)$ returns the index k of the vocabulary entry v_k matching the word w_y at position y in the sentence.

The algorithm uses a extension of the representation and terminology used for "hidden Markov models"(hidden stochastic regular grammars) for which the Baum-Welch algorithm (Baum, 1972) is applicable (and which is also called the Forward/Backward (F/B) algorithm). Grammar rules are represented as networks and illustrated graphically, maintaining a correspondence

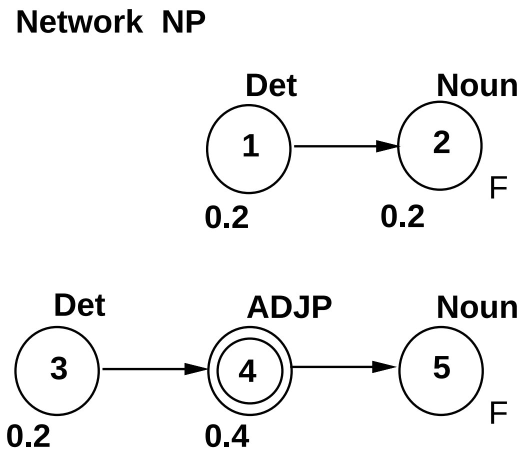

Figure 1: Networks for Lexical Rules

with the trellis structure on which the computation can be conveniently represented. The terminology is closely related to that of Levinson, Rabiner & Sondhi (1983) and also Lari & Young (1990).

A set of $\mathcal{N}$ different nonterminals are represented by $\mathcal{N}$ networks. A component network for the nonterminal labeled n has a parameter set (A, B, I, N, F, Top, n). To uniquely identify an element of the parameter set requires that it be a function of its nonterminal label (e.g. $A(n)$, $I(n)$ etc.). However this notation has been dropped to make formulae less cumbersome. A network labeled NP is shown in Figure 1 which represents the following rules:

NP $\Longrightarrow$ **Noun (0.2)**
NP $\Longrightarrow$ **Det Noun (0.2)**
NP $\Longrightarrow$ **Det ADJP Noun (0.2)**
NP $\Longrightarrow$ **ADJP Noun (0.4)**
Noun $\Longrightarrow$ **"cat" (0.002)**
Noun $\Longrightarrow$ **"dog" (0.001)**
Det $\Longrightarrow$ **"the" (0.5)**
Det $\Longrightarrow$ **"a" (0.2)**

The rule **NP** $\Longrightarrow$ **Noun (0.2)** means that if the NP rule is used, the probability is 0.2 that it produces a single **Noun**. In Figure 1, states are represented by circles with numbers inside to index them. *Nonterminal* states are shown with double circles and represent references to other networks, such as $ADJP$. States marked with single circles are called *terminal* states and represent part-of-speech categories. When a transition is made to a terminal state, a word of the current training sentence is generated. The word must have the same category as the state that generated it. Rules of the form **Noun** $\Longrightarrow$ **"cat" (0.002)** and **Noun** $\Longrightarrow$ **"dog" (0.001)** are collapsed into a state-dependent probability vector $b(j)$,

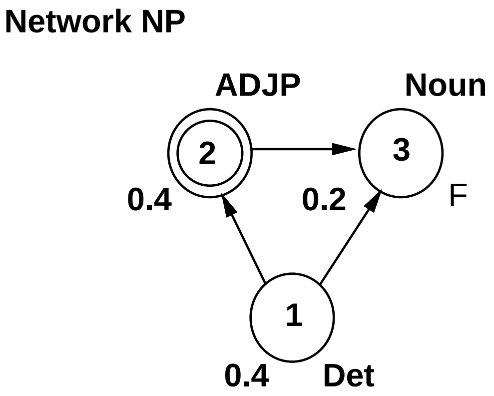

Figure 2: Equivalent Network

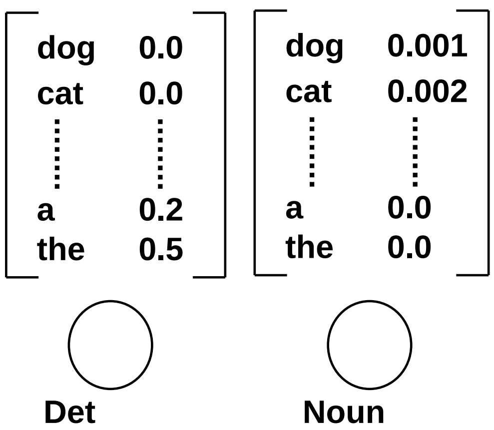

Figure 3: Representation for Terminal Productions

which is an element of the output matrix B. Elements of the vector such as $b(j, W(y))$ represent the probability of seeing word w_y in terminal state j. A transition to a nonterminal state does not in itself generate any words, however terminal states within the referenced network will do so. The parameter N is a matrix which indicates the label (e.g. n, NP, $ADJP$) of the network that a nonterminal state refers to. The probability of making a transition from state i to state j is labeled $a(i, j)$ and collectively these probabilities form the transition matrix A. The initial matrix I contains the production probabilities for rules that are modelled by the network. They are indicated in Figure 1 as numbers beneath the state, if they are non-zero. $I(i)$ can be equivalently viewed as the probability that some subsequence of n is started at state i. The parameter F is the set of final states; any sequence accepted by the network must terminate on a final state. In Figure 1 final states are designated with the annotation "F". The boolean value Top indicates whether the network is the top-level network. Only one network may be assigned as the top-level network, which models productions involving the root symbol of a grammar.

An equivalent network for the same set of rules is shown in Figure 2. The lexical rules can be written compactly as networks, with fewer states. The transitions from the determiner state each have probability 0.5 (i.e $a(1,2) = a(1,3) = 0.5$). It should be noted that the algorithm can operate on either network.

TRELLIS DIAGRAM

Trellis diagrams conveniently relate computational quantities to the network structure and a training sentence. Each network n has a set of $Y+1$ trellises for subsequences of a sentence $w_0 ... w_Y$, starting at each different position and ending at subsequent ones. A single trellis spanning positions 0...2 is shown in Figure 4 for network NP. Nonterminal states are associated with a row of *start* nodes indicating where daughter constituents may start, and a row of *end* nodes that indicate where they end. A pair of start/end nodes thus refer to a daughter nonterminal constituent. In Figure 4, the ADJP network is referenced via the start state at position 0. An adjective is then generated by a terminal state in the trellis for the ADJP network, followed by a transition and another adjective. The ADJP network is left at position 1, and a transition is made to the noun state where the word "cat" is generated. Terminal states are associated with a single row of nodes in the trellis (they represent terminal productions that span only a single position). The path taken through the trellis is shown with broken a line. A path through different trellises has a corresponding unique tree representation, as exemplified in Figure 5. In cases where parses are ambiguous, several paths exist corresponding to the alternative derivations. We shall next consider the computation of the probabilities of the paths. Two basic quantities are involved, namely *alpha* and *beta* probabilities. Loosely speaking, the alphas represent probabilities of subtrees associated with nonterminals, while the betas refer to the rest of the tree structure external to the subtrees. Subsequently, products of these quantities will be formed, which represent the probabilities of productions being used in generating a sentence. These are summed over all sentences in the training corpus to obtain the expected number of times each production is used, based on the current production probabilities and the training corpus. These are used like frequency counts would be for a parsed corpus, to form ratios that represent new estimates of the production probabilities. The procedure is iterated several times until the estimates do not change between iterations (i.e. the overall likelihood of producing the training corpus no longer increases).

The algorithm makes use of one set of trellis diagrams to compute alpha probabilities, and another for beta probabilities. These are both split into *terminal*, *nonterminal-start* and *nonterminal-end* probabilities, corresponding to the three different types of nodes in the trellis diagram. The alpha set are labeled α_t, α_{nts} and α_{nte} respectively. The algorithm was originally formulated using solely the trellis representation (Kupiec, 1991) however the definitions that follow will

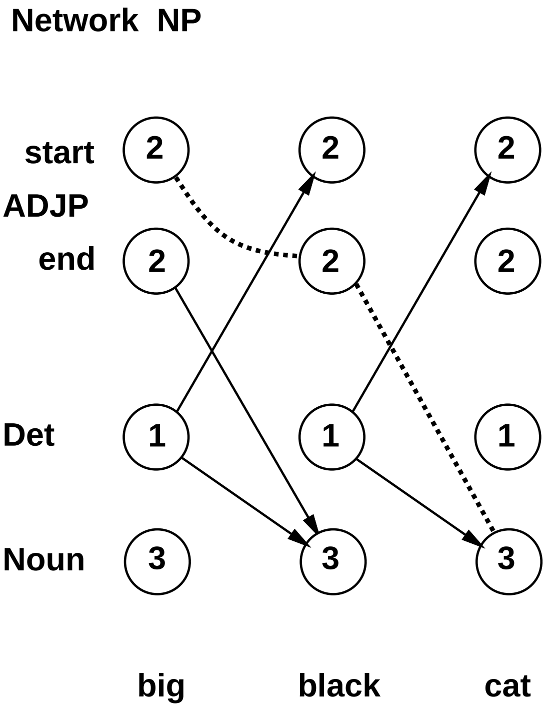

Figure 4: A Path through a Trellis Diagram

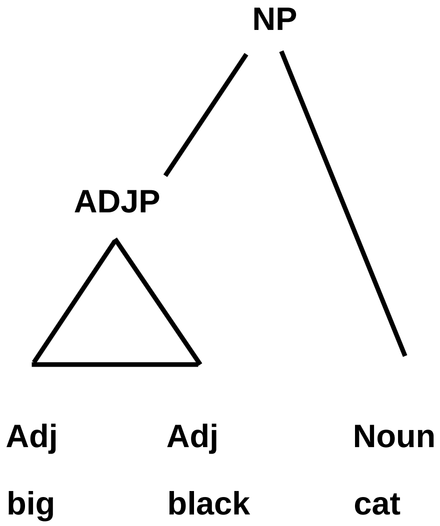

Figure 5: The Equivalent Tree

also be related to the consituent structures used in the equivalent parse trees. In the following equations, three sets will be mentioned:

1. $Term(n)$ The set of terminal states in network n.
2. $Nonterm(n)$ This is the set of nonterminal states in network n.
3. $Final(n)$ The set F of final states in network n.

$\alpha_t(x, y, j, n)$: The probability that network n generates the words $w_x...w_y$ inclusive and is at the node for terminal state j at position y.

$$\begin{aligned}\alpha_t(x,y,j,n) = \\ &\left[\sum_i \alpha_t(x,y-1,i,n)a(i,j)\right] b(j,W(y)) \\ &+ \left[\sum_q \alpha_{nte}(x,y-1,q,n)a(q,j)\right] b(j,W(y))\end{aligned}$$

$$\begin{array}{ll} 0 < y \leq Y & j, i \in Term(n) \\ 0 \leq x < y & q \in Nonterm(n) \end{array} \quad (1)$$

$$\begin{array}{lll} \alpha_t(x,x,j,n) & = & I(j)b(j,W(x)) \\ 0 \leq x \leq Y & & j \in Term(n) \end{array} \quad (2)$$

$\alpha_t(x, y, j, n)$ represents a constituent for nonterminal n spanning positions $x...y$. It is formed by extending an incomplete constituent for n, by addition of the terminal w_y at state j. The two terms indicate cases where the constituent previously ended on either a terminal or another constituent completed at $y - 1$ (as in Figure 5, where the complete ADJP constituent is followed by the noun "cat"). If j is a final state the extended constituent is complete.

$\alpha_{nts}(x, y, p, n)$: The probability that network n generates the words $w_x...w_{y-1}$ inclusive, and is at the start node of nonterminal state p at position y.

$$\begin{aligned}\alpha_{nts}(x,y,p,n) &= \sum_i \alpha_t(x,y-1,i,n)a(i,p) \\ &\quad + \sum_q \alpha_{nte}(x,y-1,q,n)a(q,p) \\ 0 < y \leq Y &\qquad p, q \in Nonterm(n) \\ 0 \leq x < y &\qquad i \in Term(n) \end{aligned} \quad (3)$$

$$\begin{aligned}\alpha_{nts}(x,x,p,n) &= I(p) \\ 0 \leq x \leq Y &\qquad p \in Nonterm(n)\end{aligned} \quad (4)$$

$\alpha_{nts}(x, y, p, n)$ represents an incomplete constituent for nonterminal n whose left subtrees span $x...y - 1$, and whose next extension will involve trees dominated by $N(p, n)$, the nonterminal referred to by state p.

$\alpha_{nte}(x, y, p, n)$: The probability that network n generates the words $w_x...w_y$ inclusive, and is at the end node of nonterminal state p at position y.

$$\begin{aligned}\alpha_{nte}(x,y,p,n) = \\ &\sum_{x \leq v \leq y} \alpha_{nts}(x,v,p,n)\alpha_{total}(v,y,N(p,n))\end{aligned}$$

$$\begin{array}{ll} 0 \leq y \leq Y & p \in Nonterm(n) \\ 0 \leq x \leq y & \end{array} \quad (5)$$

$$\alpha_{total}(v,y,n) = \sum_i \alpha_t(v,y,i,n) + \sum_p \alpha_{nte}(v,y,p,n)$$

$$\begin{array}{ll} 0 \leq y \leq Y & i \in Term(n)\ \&\ i \in Final(n) \\ 0 \leq v \leq y & p \in Nonterm(n)\ \&\ p \in Final(n) \end{array} \quad (6)$$

$\alpha_{nte}(x, y, p, n)$ represents the probability of a constituent for n that spans $x...y$, formed by extending the various constituents $\alpha_{nts}(x, v, p, n)$ (ending at $v - 1$) with corresponding completed constituents starting at v, ending at y and dominated by $N(p, n)$.

The quantity $\alpha_{total}(v, y, n)$ refers to the probability that network n generates the words $w_v...w_y$ inclusive and is in a final state of n at position y. Equivalently it is the probability that nonterminal n dominates all derivations that span positions $v...y$. The α_{total} probabilities correspond to the "Inner" (bottom-up) probabilities of the I/O algorithm. If a network corresponding to Chomsky normal form is substituted in equation (6), the recursion for the inner probabilities of the I/O algorithm will be produced after further substitutions using equations (1)-(6).

In the previous equations (5) and (6) it can be seen that the α_{nte} probabilities for a network are defined recursively. They will never be self-referential if the grammar is cycle-free, (i.e. there are no derivations $A \overset{+}{\Longrightarrow} A$ for any nonterminal production A). Although only cycle-free grammars are of interest here, it is worth mention that if cycles do exist (with associated probabilities less than unity), the recursions form a geometric series which has a finite sum.

The alpha probabilities are all computed first because the beta probabilities make use of them. The latter are defined recursively in terms of trees that are external to a given constituent, and as a result the recursions are less obvious than those for the alpha probabilities. The basic recursion rests on the quantity β_{nte} which involves the following functions β_{above} and β_{side}:

$$\beta_{above}(x,y,n) =$$

$$\sum_{m\in\mathcal{N}} \sum_{r:N(r,m)=n} \sum_{0\le v\le x} \alpha_{nts}(v,x,r,m)\beta_{nte}(v,y,r,m)$$

$$r \in Nonterm(m) \tag{7}$$

$$\beta_{side}(x,y,l,n) =$$

$$\sum_i a(l,i)\beta_t(x,y+1,i,n)b(i,W(y+1)) +$$

$$\sum_q a(l,q) \sum_{y<w\le Y} \alpha_{total}(y+1,w,N(q,n))\beta_{nte}(x,w,q,n)$$

$$i \in Term(n)$$
$$q \in Nonterm(n) \tag{8}$$

Given a constituent n spanning $x...y$, $\beta_{above}(x,y,n)$ indicates how the constituents spanning $v...y$ and labeled m that immediately dominate n relate to the constituents that are in turn external to m via $\beta_{nte}(v,y,r,m)$. This situation is shown in Figure 6, where for simplicity $\beta_{nte}(v,y,r,m)$ has not been graphically indicated. Note that m can dominate $x...y$ as well as left subtrees.

$\beta_{side}(x,y,l,n)$ defines another recursion for a constituent labeled n that spans $x...y$, and is in state l at time y. The recursion relates the addition of right subtrees (spanning $y{+}1...w$) to the remaining external tree, via $\beta_{nte}(x,w,q,n)$. This is depicted in Figure 7 (again the external trees represented by $\beta_{nte}(x,w,q,n)$ are not shown). Also omitted from the figure is the first term of Equation 8 which relates the addition of a single terminal at position $y+1$ to an external tree defined by $\beta_t(x,y+1,i,n)$. β_t and the various other probabilities for a beta trellis are defined next:

$\beta_t(x,y,j,n)$: The probability of generating the prefix $w_0...w_{x-1}$ and suffix $w_{y+1}...w_Y$ given that network n generated $w_x...w_y$ and is in terminal state j at position y. The indicator function $Ind()$ is used in subsequent equations. Its value is unity when its argument is true and zero otherwise.

$$\beta_t(x,y,j,n) \;=\; \beta_{side}(x,y,j,n)$$
$$+\; Ind(j \in Final(n))\beta_{above}(x,y,n)$$

$$0 \le y < Y \qquad j \in Term(n)$$
$$0 \le x \le y \tag{9}$$

$$\beta_t(x,Y,j,n) \;=\; \beta_{above}(x,Y,n)$$

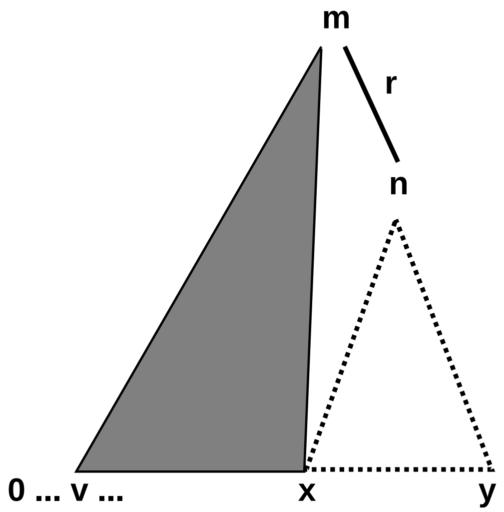

Figure 6: Part of $\beta_{above}(x,y,n)$

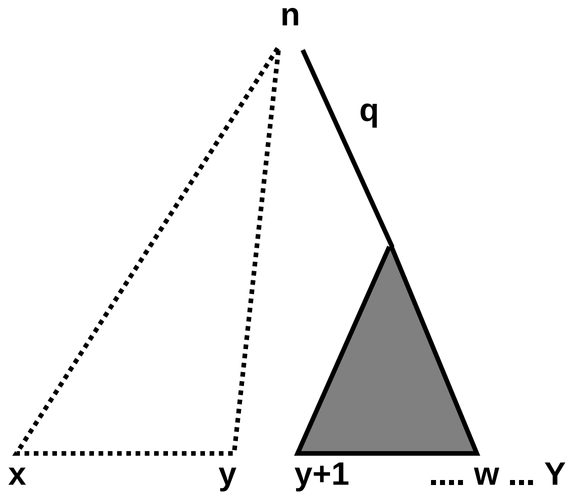

Figure 7: Part of $\beta_{side}(x,y,l,n)$

$$0 \le x \le Y \qquad j \in Term(n)$$
$$j \in Final(n) \;\&\; \sim Top(n) \tag{10}$$

$$\beta_t(0,Y,j,n) \;=\; 1.0$$
$$j \in Term(n)$$
$$j \in Final(n) \;\&\; Top(n) \tag{11}$$

The first term in Equation 9 describes the relationship of the tree external to $x...y+1$ to the tree external to $x...y$, with respect to state j generating the terminal w_{y+1} at time $y+1$. If the constituent n spanning $x...y$ is complete, the second term describes the probability of the external tree via constituents that immediately dominate n.

$\beta_{nte}(x,y,p,n)$: The probability of generating the prefix $w_0...w_{x-1}$ and suffix $w_{y+1}...w_Y$ given that network n generated $w_x...w_y$ and the end node of state p is reached at position y.

$$\begin{aligned}\beta_{nte}(x,y,p,n) &= \beta_{side}(x,y,p,n) \\ &\quad + Ind(p \in Final(n))\beta_{above}(x,y,n)\end{aligned}$$

$$\begin{array}{ll} 0 \le y < Y & i \in Term(n),\ p \in Nonterm(n) \\ 0 \le x \le y & \end{array} \tag{12}$$

$$\begin{array}{rcl} \beta_{nte}(x,Y,p,n) &=& \beta_{above}(x,Y,n) \\ 0 \le x \le Y && p \in Nonterm(n) \\ && p \in Final(n)\ \&\ \sim Top(n) \end{array} \tag{13}$$

$$\begin{array}{rcl} \beta_{nte}(0,Y,p,n) &=& 1.0 \\ && p \in Nonterm(n) \\ && p \in Final(n)\ \&\ Top(n) \end{array} \tag{14}$$

$\beta_{nte}(x,y,p,n)$ has the same form as the previous formula for β_t, but is used with nonterminal states. Via β_{above} and β_{side} it relates the tree external to the constituent n (spanning $x...y$) to the trees external to $v...y$ and $x...w$. During the recursion, the trees shown in Figures 6 and 7 are substituted into each other (at the positions shown with shaded areas). Thus the external trees are successively extended to the left and right, until the root of the outermost tree is reached. It can be seen that the values for $\beta_{nte}(x,y,p,n)$ are defined in terms of those in other networks which reference n via β_{above}. As a result this computation has a top-down order. In contrast, the $\alpha_{nte}(x,y,p,n)$ probabilities involve other networks that are referred to by network n and so assigned in a bottom-up order. If the network topology for Chomsky normal form is substituted in equation (12), the recursion for the "Outer" probabilities of the I/O algorithm can be derived after further substitutions. The β_{nte} probabilities for final states then correspond to the outer probabilities.

$\beta_{nts}(x,y,p,n)$: The probability of generating the prefix $w_0...w_{x-1}$ and suffix $w_y...w_Y$ given that network n generated $w_x...w_{y-1}$ and is at the start node of state p at position y.

$$\beta_{nts}(x,y,p,n) = \sum_{y \le v \le Y} \alpha_{total}(y,v,N(p,n))\beta_{nte}(x,v,p,n)$$

$$\begin{array}{ll} 0 \le x \le Y & p \in Nonterm(n) \\ x \le y \le Y & \end{array} \tag{15}$$

RE-ESTIMATION FORMULAE

Once the alpha and beta probabilities are available, it is a straightforward matter to obtain new parameter estimates $(\bar{A},\bar{B},\bar{I})$. The total probability P of a sentence is found from the top-level network n_{Top}.

$$\begin{array}{rcl} P &=& \alpha_{total}(0,Y,n_{Top}) \\ && Top(n_{Top}) \end{array} \tag{16}$$

There are four different kinds of transition:

1. *Terminal* node i to *terminal* node j.
2. *Terminal* node i to *nonterminal start* node p.
3. *Nonterminal end* node p to *nonterminal start* q.
4. *Nonterminal end* node p to *terminal* node i.

The expected total number of times a transition is made from state i to state j conditioned on the observed sentence is $E(\psi_{i,j})$. The following formulae give $E(\psi)$ for each of the above cases:

$$\begin{aligned} E(\psi_{i,j}) &= \frac{1}{P}\sum_x\sum_y \alpha_t(x,y,i,n)a(i,j) \times \\ &\quad b(j,W(y+1))\beta_t(x,y+1,j,n) \end{aligned} \tag{17}$$

$$\begin{aligned} E(\psi_{i,p}) &= \frac{1}{P}\sum_x\sum_y \alpha_t(x,y,i,n)a(i,p) \times \\ &\quad \beta_{nts}(x,y+1,p,n) \end{aligned} \tag{18}$$

$$\begin{aligned} E(\psi_{p,q}) &= \frac{1}{P}\sum_x\sum_y \alpha_{nte}(x,y,p,n)a(p,q) \times \\ &\quad \beta_{nts}(x,y+1,q,n) \end{aligned} \tag{19}$$

$$\begin{aligned} E(\psi_{p,i}) &= \frac{1}{P}\sum_x\sum_y \alpha_{nte}(x,y,p,n)a(p,i) \times \\ &\quad b(i,W(y+1))\beta_t(x,y+1,i,n) \end{aligned} \tag{20}$$

$$\begin{array}{rl} 0 = x & Top(n) \\ 0 \le x < Y & \sim Top(n) \\ x \le y < Y & \end{array}$$

A new estimate $\bar{a}(i,j)$ for a typical transition is then:

$$\bar{a}(i,j) = \frac{E(\psi_{i,j})}{\sum_j E(\psi_{i,j})} \tag{21}$$

Only B matrix elements for terminal states are used, and are re-estimated as follows. The expected total number of times the k'th vocabulary entry v_k is generated in state i conditioned on the observed sentence is $E(\eta_{i,k})$. A new estimate for $\bar{b}(i,k)$ can then be found:

$$E(\eta_{i,k}) = \frac{1}{P}\sum_x \sum_{y:W(y)=k} \alpha_t(x,y,i,n)\beta_t(x,y,i,n)$$

$$\begin{array}{rl} 0 = x & i \in Term(n)\ \&\ Top(n) \\ 0 \le x \le Y & i \in Term(n)\ \&\ \sim Top(n) \end{array}$$

$$x \leq y \leq Y \tag{22}$$

$$\bar{b}(i,k) = \frac{E(\eta_{i,k})}{\sum_k E(\eta_{i,k})} \tag{23}$$

The initial state matrix I is re-estimated as follows:

$$\bar{I}(i) = \frac{1}{P}\sum_x \alpha_t(x,x,i,n)\beta_t(x,x,i,n)$$

$$0 = x \qquad i \in Term(n) \ \& \ Top(n)$$

$$0 \leq x \leq Y \qquad i \in Term(n) \ \& \ \sim Top(n) \tag{24}$$

$$\bar{I}(p) = \frac{1}{P}\sum_x \alpha_{nts}(x,x,p,n)\beta_{nts}(x,x,p,n)$$

$$0 = x \qquad p \in Nonterm(n) \ \& \ Top(n)$$

$$0 \leq x \leq Y \qquad p \in Nonterm(n) \ \& \ \sim Top(n) \tag{25}$$

DISCUSSION

The preceding equations have been implemented as a computer program and this section describes some practical issues with regard to a robust implementation. The first concerns the size of the B matrix. For pedagogical reasons individual words were shown as elements of this matrix. A vocabulary exceeding 220,000 words is actually used by the program so it is not practical to try to reliably estimate the B matrix probabilities for each word individually. Instead, only common words are represented individually; the rest of the words in the dictionary are partitioned into *word equivalence classes* (Kupiec, 1989) such that all words that can function as a particular set of part-of-speech categories are given the same label. Thus "type", "store" and "dog" would all be labelled as *singular-noun-or-uninflected-verb*. For the category set that is used, only 250 different equivalence classes are necessary to cover the dictionary.

It is important that the initial guesses for parameter values are well-informed. All productions for any given nonterminal were intially assumed to be equally likely, but the B matrix values were conveniently copied from a trained hidden Markov model (HMM) used for text-tagging. The HMM was also found very useful for verifying correct program operation. The algorithm has worst-case cubic complexity in both the length of a sentence and the number of nonterminal states in the grammar. An index can be used to efficiently update terminal states. For any word (or equivalence class) the index determines which terminal states require updating. Also when all probabilities in a column of any trellis become zero, no further computation is required for any other columns in the trellis. Grammars are currently being developed, and initial experiments have typically used eight training iterations, on training corpora comprising 10,000 sentences or more (having an average of about 22 words per sentence).

To complement the training algorithm, a parser has also been constructed which is a corresponding analogue of the Cocke-Younger-Kasami parser. The parser is quite similar to the training algorithm, except that maximum probability paths are propagated instead of sums of probabilities. Trained grammars are used by the parser to predict the most likely syntactic structure of new sentences. The applications for which the parser was developed make use of incomplete parses if a sentence is not covered by the grammar, thus top-down filtering is not used.

REFERENCES

[1] 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 & J.J. Wolf, eds), pp. 547-550.

[2] Baum, L.E. (1972). An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of a Markov Process. *Inequalities*, **3**, pp. 1-8.

[3] Chitrao, M.V. & Grishman, R. (1990). Statistical Parsing of Messages. *Proceedings of the DARPA Speech and Natural Language Workshop.*

[4] Dempster, A.P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, **B39**, pp. 1-38.

[5] Fujisaki, T., Jelinek, F., Cocke, J., Black, E. & Nishino, T. (1989). A Probabilistic Parsing Method for Sentence Disambiguation. *International Workshop on Parsing Technologies*, Pittsburgh, PA. pp. 85-94.

[6] Jelinek, F. (1985). Markov Source Modeling of Text Generation. *Impact of Processing Techniques on Communication* (J.K. Skwirzinski, ed), Nijhoff, Dordrecht.

[7] Kupiec, J.M. (1989). Augmenting a Hidden Mar kov Model for Phrase-Dependent Word Tagging. *Proceedings of the DARPA Speech and Natural Language Workshop*, Cape Cod, MA pp. 92-98. Morgan Kaufmann.

[8] Kupiec, J.M. (1991). A Trellis-Based Algorithm for Estimating the Parameters of a Hidden Stochastic Context-Free Grammar. *Proceedings of the DARPA Speech and Natural Language Workshop*, Pacific Grove, CA pp. 241-246. Morgan Kaufmann.

[9] Lari, K. & Young, S.J. (1990). The Estimation of Stochastic Context-Free Grammars Using the Inside-Outside Algorithm. *Computer Speech and Language*, **4**, pp. 35-56.

[10] 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**, pp. 1035-1074.