A Trellis-Based Algorithm For Estimating The Parameters Of A Hidden Stochastic Context-Free Grammar

Julian Kupiec

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

ABSTRACT

The paper presents a new algorithm for estimating the parameters of a hidden stochastic context-free grammar. In contrast to the Inside/Outside (I/O) algorithm it does not require the grammar to be expressed in Chomsky normal form, and thus can operate directly on more natural representations of a grammar. The algorithm uses a trellis-based structure as opposed to the binary branching tree structure used by the I/O algorithm. The form of the trellis is an extension of that used by the Forward/Backward algorithm, and as a result the algorithm reduces to the latter for components that can be modeled as finite-state networks. In the same way that a hidden Markov model (HMM) is a stochastic analogue of a finite-state network, the representation used by the new algorithm is a stochastic analogue of a recursive transition network, in which a state may be simple or itself contain an underlying structure.

INTRODUCTION

The algorithm described in this paper is concerned with using hidden Markov methods for estimation of the parameters of a stochastic context-free grammar from free text. The Forward/Backward (F/B) algorithm (Baum, 1972) is capable of estimating the parameters of a hidden Markov model (i.e. a hidden stochastic regular grammar) and has been used with success to train text taggers (Jelinek, 1985). In the tagging application the observed symbols are words and their underlying lexical categories are the hidden states of the model.

A context-free grammar comprises both lexical (terminal) categories and grammatical (nonterminal) categories. One iterative method of estimation in this case involves parsing each sentence in the training corpus and for each derivation, accumulating counts of the number of times each rule is used. This method has been used by Fujisaki et al. (1989), and Chitrao & Grishman (1990). A more efficient method is the Inside/Outside algorithm, devised by Baker (1979) for grammars that are expressed in Chomsky normal form. The algorithm described in this paper relaxes the requirement for a grammar to be expressed in a normal form, and it is based on a trellis representation that is closely related to the F/B algorithm, and which reduces to it for finite-state networks.

The development of the algorithm has various motivations. Grammars must provide a large coverage to accommodate the diversity of expression present in large collections of unrestricted text. As a result they become more ambiguous. A stochastic grammar provides the capability to resolve ambiguity on a probabilistic basis, providing a practical approach to the problem. It also provides a way of modeling conditional dependence for incomplete grammars, or in the absence of any specific structural information. The latter is exemplified by the approach taken in many current taggers, which have a uniform model of second-order dependency between word categories. Kupiec (1989) has experimented with the inclusion of networks to model mixed-order dependencies.

The use of hidden Markov methods is motivated by the flexibility they afford. Text corpora from any domain can be used for training, and there are no restrictions on a grammar due to conventions used during labeling. The methods also lend themselves to multi-lingual application.

The representation used by the algorithm can be related to constituent structures used in other parsers such as chart parsers, providing a means of embedding this technique in them.

REPRESENTATION

The representation of a grammar and the basic trellis structure are discussed in this section. The starting point is the conventional HMM network in which symbols are generated at states (rather than on transitions) as described in Levinson et al. (1983). Such a network is represented by the parameter set (A, B, I) comprising the transition, output and initial matrices. The states in this kind of network will be referred to as *terminal* states from now on, and will be represented pictorially with single circles. As a shorthand convenience in what follows, if the circle contains a symbol, then it is assumed that only that symbol is ever generated by the state. (The probability of generating it is then unity, and zero for all other symbols.) A single symbol is generated by a transition to a terminal state. For the grammars considered here, terminal states correspond to lexical categories.

To this parameter set we will add four other parameters (N, F, Top, L). The boolean Top indicates whether the network is to be considered as the top-level network. Only one network may be assigned as the top-level network, and it is analogous to the root symbol of a grammar. The parameter F is the set of final states, specifying the allowable states in which a network can be considered to have accepted a sequence of observations. A different type of state will now be introduced, called a *nonterminal* state. It represents a reference to another network and is indicated diagrammatically with two concentric circles. When a transition is made to a nonterminal state, the state does not generate any observations *per se*, but terminal nodes within the referred network do. A nonterminal state may be associated with a sequence of observation symbols, corresponding to the sequence accepted by the underlying network. The parameter N is a matrix which indicates whether a state is a terminal or nonterminal state. Terminal states have a null entry in the matrix, and nonterminal states have a reference to the network which they represent. A grammar is usually composed of several networks, so each one is referred to with a unique label L.

Figure 1 shows how rules in Chomsky normal form are represented as networks using the above scheme. The lexical form of the rules is included, illustrating how the left hand side of a rule corresponds to a network label, and the network structure is associated with the right-hand side. Terminal states are labeled in lower case and nonterminals in upper case. The numbers associated with the states are their initial probabilities which are also rule probabilities. For terminal nodes in the top-level network, initial probabilities have the same meaning as in the F/B algorithm. For all other networks, an initial probability corresponds to a production probability. States which have a non-zero initial probability will be termed "Initial states" from now on. Any sequence recognized by a network must start on an initial state and end on a final state. In Figure 1, final states are designated with the annotation "F". Figure 2 shows how the terminal symbols in Figure 1 may be represented in a more compact style, by a single state having different B matrix probabilities for the symbols x and y.

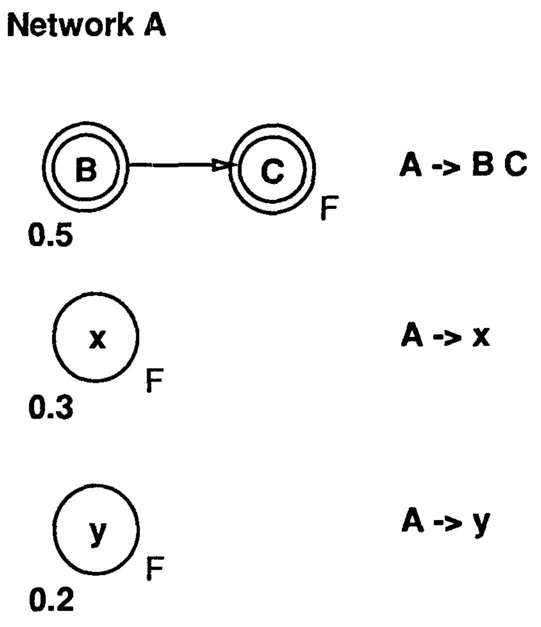

Figure 1: Network and Rules for Chomsky Normal Form

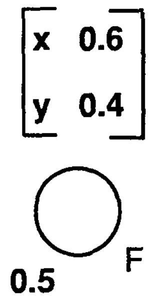

Figure 2: Representation for Terminal Symbols

Terminology

A grammar is represented as a set $\mathcal{N}$ of networks, and a component network labeled n is composed of parameters (A, B, I, N, F, Top, n). To strictly identify an element in the parameter set each element must be a function of its associated network (e.g. $A(n)$, $I(n)$ etc.). In the following sections however, where the reference is obvious this notation has been omitted to make formulae less cumbersome. Thus, given a network $n \in \mathcal{N}$, an element of its transition matrix A, from state i to state j is written $a(i, j)$. Likewise the initial probability for state i is $I(i)$. Assuming that sentences are used as text units, an observation sequence may consist of $Y + 1$ words, indexed from 0 to Y:

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

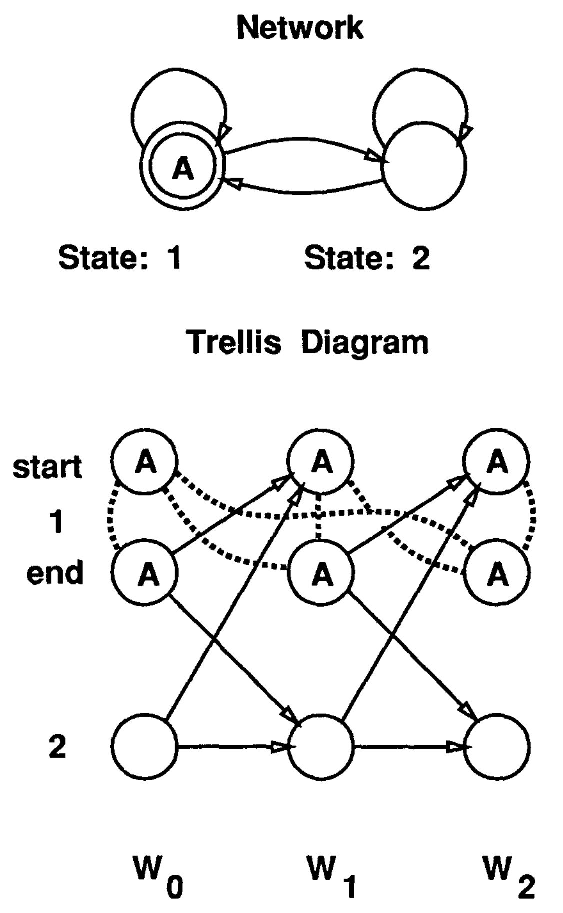

Figure 3: An Example Network and Trellis Diagram

It is useful to define a lookup function $W(y)$ which returns the index k of the vocabulary entry v_k matching the word w_y at position y in the sentence. The vocabulary entry may be a word or an equivalence class based on categories (Kupiec, 1989). An element of the output matrix B, representing the the probability of seeing word w_y in terminal state j is then $b(j, W(y))$. In addition, 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.

The predicate $Top(n)$ indicates that n is a top-level network, and $\sim Top(n)$ indicates it isn't. Finally, the function $N(p, n)$ returns the network to which state p in network n refers to. (If p is a terminal state it returns a null value).

TRELLIS DIAGRAM

Figure 3 shows the form of the trellis diagrams that are used for the computation of probabilities. In the F/B algorithm a single trellis is used, whose dimensions are the number of states in the network and the length of the sentence. A single trellis spans the whole sentence. In the new algorithm each network has an associated set of trellises, for subsequences starting at different positions in a sentence and ending at subsequent ones. (Only a single trellis starting at w_0 is shown in Figure 3.) It can be seen that terminal state *2* has corresponding nodes in the trellis diagram, but nonterminal state *1* is represented by pairs of nodes. One node of the pair is called the ***start*** node and the other is termed the *end* node. Paths exist in the trellis for possible state transitions between successive words. However, it is also implicitly understood that paths also exist between the start node and subsequent end nodes for each nonterminal state. These implicit paths are shown as broken lines in Figure 3 and correspond to paths that enter network A at some time, and return from it at the same or a later time. The probabilities associated with the implicit paths are assigned by reference to the trellis diagrams of the appropriate network. An implicit path from a start node at position x to an end node at position y for a nonterminal state p can be thought of as a constituent labeled p, that dominates the words from positions x through to y (inclusive) in a sentence. A network n is deemed to ***include*** the sequence $w_x \ldots w_y$ if paths exists through the network which will generate this sequence or a longer one which includes it as a prefix. Thus it is not necessary to be at a final state of n at word w_y to include $w_x \ldots w_y$.

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. For the alpha set, these are labeled α_t, α_{nts} and α_{nte} respectively.

$\alpha_t(x, y, j, n)$: The probability of generating the words $w_x \ldots w_y$ inclusive and network n ***includes*** them, and being at the node for terminal state j at position y.

$$\begin{aligned}\alpha_t(x,y,j,n) &= \left[\sum_i \alpha_t(x,y-1,i,n)a(i,j)\right] b(j,W(y)) \\ &\quad + \left[\sum_q \alpha_{nte}(x,y-1,q,n)a(q,j)\right] b(j,W(y)) \\ 0 < y \le Y &\qquad j,i \in Term(n) \\ 0 \le x < y &\qquad q \in Nonterm(n) \qquad (1)\end{aligned}$$

$$\begin{aligned}\alpha_t(x,x,j,n) &= I(j)b(j,W(x)) \\ 0 \le x \le Y &\qquad j \in Term(n) \qquad (2)\end{aligned}$$

It can be seen that if $x = 0$ and there are no nonterminal states, the previous expressions are as in the F/B algorithm.

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

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

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

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

$$\begin{aligned} \alpha_{nte}(x, y, p, n) &= \sum_{x \leq v \leq y} \alpha_{nts}(x, v, p, n) \alpha_{total}(v, y, N(p, n)) \\ 0 \leq y \leq Y &\quad p \in Nonterm(n) \\ 0 \leq x \leq y & \end{aligned} \tag{5}$$

$$\begin{aligned} \alpha_{total}(v, y, n) &= \sum_i \alpha_t(v, y, i, n) + \sum_p \alpha_{nte}(v, y, p, n) \\ 0 \leq y \leq Y &\quad i \in Term(n) \ \& \ i \in Final(n) \\ 0 \leq v \leq y &\quad p \in Nonterm(n) \ \& \ p \in Final(n) \end{aligned} \tag{6}$$

The quantity $\alpha_{total}(v, y, n)$ refers to the probability that network n generates the words $w_v ... w_y$ inclusive and being in a final state of n at position y. The α_{total} probabilities correspond to the "Inner" (bottom-up) probabilities of the I/O algorithm. If the network topology for Chomsky normal form shown in Figure 1 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 in terms of other ones. 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). In the new algorithm cycles can be detected and self-referencing avoided. This is a similar situation to a chart parser where once a constituent with a given label, start and end position is built, no further instances of it are added.

The alpha probabilities are all computed first. The beta probabilities can then be calculated, which unlike the F/B algorithm involve the alpha probabilities because prefixes of a sentence must be accounted for as well as suffixes. The beta probabilities are described below. For convenience in later equations the following functions β_{above} and β_{side} are first defined:

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

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

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

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

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

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

$$\begin{aligned} &i \in Term(n) \\ &q \in Nonterm(n) \end{aligned} \tag{8}$$

$\beta_t(x, y, j, n)$: The probability of generating the prefix $w_0 ... w_{x-1}$ and suffix $w_{y+1} ... w_Y$ given that network n includes $w_x ... w_y$ 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. In addition, elements that are not explicitly referenced by the ranges in the equations are assumed to be zero.

$$\begin{aligned} \beta_t(x, y, j, n) &= \beta_{side}(x, y, j, n) \\ &\quad + Ind(j \in Final(n)) \beta_{above}(x, y, n) \\ 0 \leq y < Y &\quad j \in Term(n) \\ 0 \leq x \leq y & \end{aligned} \tag{9}$$

$$\begin{aligned} \beta_t(x, Y, j, n) &= \beta_{above}(x, Y, n) \\ 0 \leq x \leq Y &\quad j \in Term(n) \\ &\quad j \in Final(n) \ \& \ \sim Top(n) \end{aligned} \tag{10}$$

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

The previous equations reduce to the definitions for β in the F/B algorithm when $x = 0$ and there are no nonterminal states in the network.

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

$$\begin{aligned} \beta_{nte}(x,y,p,n) &= \beta_{side}(x,y,p,n) \\ &\quad + Ind(p \in Final(n))\beta_{above}(x,y,n) \\ 0 \le y < Y &\quad i \in Term(n),\ p \in Nonterm(n) \\ 0 \le x \le y &\quad q \in Nonterm(m) \end{aligned} \tag{12}$$

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

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

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 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 includes $w_x...w_{y-1}$ and is at at the start node of state p at position y

$$\begin{aligned} \beta_{nts}(x,y,p,n) &= \sum_{y \le v \le Y} \alpha_{total}(y,v,N(p,n))\beta_{nte}(x,v,p,n) \\ 0 \le x \le Y &\quad p \in Nonterm(n) \\ x \le y \le Y & \end{aligned} \tag{15}$$

RE-ESTIMATION FORMULAE

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

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

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* node 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:

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

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

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

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

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

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:

$$\begin{aligned} E(\eta_{i,k}) &= \frac{1}{P}\sum_x \sum_{y:W(y)=k} \alpha_t(x,y,i,n)\beta_t(x,y,i,n) \\ 0 = x &\quad i \in Term(n)\ \&\ Top(n) \\ 0 \le x \le Y &\quad i \in Term(n)\ \&\ \sim Top(n) \\ x \le y \le Y & \end{aligned} \tag{22}$$

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

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

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

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

$$I(p) = \frac{1}{P} \sum_{x} \alpha_{nts}(x,x,p,n)\beta_{nts}(x,x,p,n)$$
$$0 = x \qquad p \in Nonterm(n) \ \& \ Top(n)$$
$$0 \leq x \leq Y \qquad p \in Nonterm(n) \ \& \ \sim Top(n) \qquad (25)$$

IMPLEMENTATION

Inspection of the preceding equations indicates that in similar fashion to the I/O algorithm, this algorithm has cubic complexity in both the length of a sentence and the number of states in the grammar. It has been implemented as a computer program, and verification was conducted in four stages, to facilitate debugging:

1. Using top-level networks having only terminal states, check for exact numerical agreement of re-estimated parameters with those obtained by applying the F/B algorithm to the same examples.
2. Create examples involving nonterminals, but which have finite-state equivalents, and verify as in stage 1.
3. Create examples with several references to a given network, then build a finite-state equivalent in which the references are supplanted by network copies having tied parameters. Verify as in stage 1.
4. Test using examples in Chomsky normal form and compare with results from the I/O algorithm.

Unscaled arithmetic was employed to simplify the initial implementation. Subsequent versions will include logarithmic scaling to prevent inaccuracies due to arithmetic underflow. The representation would also benefit from the inclusion of a probability matrix for final states, rather than their use simply as constraints.

As the representation used by the algorithm is a superset of that used by the F/B algorithm, it conveniently permits "Staged Training". Components that are finite-state networks can be pre-trained using the F/B algorithm, and then inserted into a context-free superstructure. This may be done to obtain improved initial estimates for the algorithm, and/or to reduce the total amount of computation involved. Lari and Young (1990) describe experiments using the I/O algorithm in which such pre-training was found useful. Using the algorithm, the parameters of a context-free grammar can be trained from a corpus of untagged text. Values for the production probabilities are directly available, and no conversion of the rules to or from Chomsky normal form is needed. Once trained, a grammar can be used to predict the most likely syntactic structure of new sentences using a corresponding analogue of the Cocke-Younger-Kasami parser.

CONCLUSION

An iterative algorithm for estimating the parameters of a hidden stochastic context-free grammar has been described, which is a generalization of the F/B algorithm and the I/O algorithm. The algorithm reduces to the F/B algorithm for finite-state grammars, and to the I/O algorithm when a context-free grammar is expressed in Chomsky normal form.

ACKNOWLEDGEMENTS

I would like to thank Phil Chou and John Maxwell of Xerox PARC, for their helpful comments on this paper.

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] 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.

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

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

[7] 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.

[8] 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.