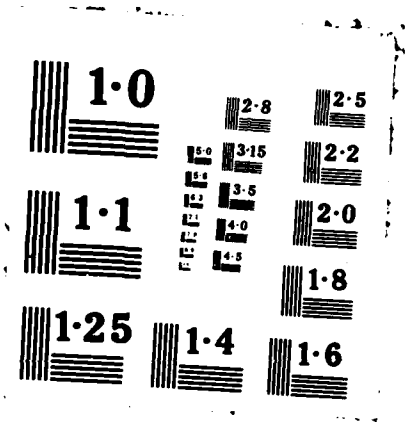
AD-A180 889 FORMAL GRAMMARS AND MARKOV MODELS(U) ROYAL SIGNALS AND 1/1
RADAR ESTABLISHMENT MALVERN (ENGLAND)
J S BRIDLE ET AL. OCT 87 RSRE-MEMO-4051 DRIC-BR-103727
UNCLASSIFIED F/G 25/4 NL


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5
5·0

UNLIMITED

RSRE
MEMORANDUM No. 4051

ROYAL SIGNALS & RADAR ESTABLISHMENT

FORMAL GRAMMARS AND MARKOV MODELS

Authors: J S Bridle and L Dodd

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
R S R E MALVERN,
WORCS.

RSRE MEMORANDUM No. 4051

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 4051

TITLE: FORMAL GRAMMARS AND MARKOV MODELS

AUTHORS: J S Bridle and L Dodd

DATE: October 1987

SUMMARY

The theory of formal grammars is widely used in computer science and linguistics. Hidden Markov models are well established in automatic speech recognition. This expository memorandum sets out the links between the two areas, via stochastic grammars, and points to stochastic context-free grammars as an interesting area for practical application.

CONTENTS

1. INTRODUCTION

The general idea of a grammar is familiar from school English and Latin. In modern Linguistics the grammar of a language is the set of rules which we use to describe the regularities observed in the patterns of word sequences used by speakers and writers of the language. There is the assumption that something equivalent to the grammar is part of the shared knowledge of the language that enables individuals in the particular language community to communicate. Different grammars may be needed for different spoken or written styles.

In contrast, a Formal Grammar is mathematical and has an existence independent of its applicability to any natural language. The application of Formal Grammars to linguistics is a matter of judgement by a linguist. Formal Grammars have been used very successfully in the design and specification of specialist languages, such as those intended for expressing algorithms, to be interpreted by people or computers [e.g. 1].

A Stochastic Model is a very important concept which may be more familiar to gamblers than to Latin scholars. A stochastic model is a mathematical abstraction of a process which is probabilistic in nature. In general, stochastic models deal with situations where some, or all, of the parameters are described by random variables rather than by deterministic quantities. A Markov process is a stochastic system in which the probability of occurrence of an event is conditionally independent on past events when we know what has happened in the recent past. A Markov chain is a particularly simple kind of stochastic model in which we are only concerned with a sequence of discrete and mutually exclusive possibilities. In some applications these possibilities are symbols, but where a Markov chain is part of a more complicated stochastic model the possibilities may be internal states. A transition probability is the probability that a transition will occur between two states at any time. A first-order Markov chain is completely described by a set of transition probabilities (from each state to each state), together with the initial probabilities of starting the process in each state.

This Memorandum is intended to introduce stochastic formal grammars to those who already have some acquaintance with Markov models.

2. MARKOV MODELS

2.1. HIDDEN MARKOV MODELS

In a Hidden Markov Model (HMM) it is assumed that the generating process proceeds through a sequence of states which are not directly observed. The state sequence is a first-order Markov Chain. The output of the model at each position in the sequence is a probabilistic function of the state sequence [2].

We shall refer to the different possible states as x(i) and to the state at

each time (say t) as X(t). (For example, if X(3)=5 then the state at time t=3 is the 5th state, x(5).) To reinforce the distinction we shall use i,j,k,l as indices of x; and r,s,t as indices of X.

Similarly, y(i) is the ith possible output symbol (or observation), and y(Y(t)) is the output symbol at time t.

To say that the sequence of states is a first-order Markov chain is to say that there is a set of 'state transition probabilities' which give the probability distribution of states at any time (t) in terms of previous states, and that this distribution, although dependent on the states at all previous times, is conditionally independent of the past given the state at time (t-1). This is to say that if we know the state at time (t-1), then we get no more information about the state at time (t) by knowing the states at previous times. So for a first-order Markov chain we write this as :

$$P(X(t)|X(1),X(2)...X(t-1)) = P(X(t)|X(t-1))$$

As a shorthand, we introduce the (first-order) state transition probability matrix, A:

$$A = P(X(t)|X(t-1)) \quad \text{for all } t$$

where $a(i,j) = P(X(t)=j|X(t-1)=i)$ is the probability that the state at time t, X(t), is the jth state given that the state at time t-1, X(t-1), is the ith state.

There are two alternative conventions used in Hidden Markov Modelling: in one formulation, each observation depends on the corresponding state; and, in the other formulation, on a transition between two states.

For the outputs-on-states variety, we have:

$$\begin{aligned} &P(Y(t)|X(1)...X(t),Y(1)...Y(t-1)) \\ =&P(Y(t)|X(t)) \\ =&b(X(t),Y(t)). \end{aligned}$$

Alternatively:

$$b(j,k) = P(Y(t)=k|X(t)=j).$$

For the outputs-on-transitions variety, we have:

$$\begin{aligned} &P(Y(t)|X(1)...X(t),Y(1)...Y(t-1)) \\ =&P(Y(t)|X(t-1),X(t)) \\ =&b(X(t-1),X(t),Y(t)). \end{aligned}$$

Alternatively:

$$b(i,j,k) = P(Y(t)=k|X(t-1)=i,X(t)=j).$$

A HMM of one kind can always be transformed into a HMM of the other kind which is equivalent in the sense that it generates the same observed output strings with the same probabilities. (However the number of states will

usually be different.)

Although the HMMs described here have discrete output distributions (multinomial), it is straightforward to extend the definitions to include continuous output distributions with parameters (eg multi-variate Gaussian). The Gaussian probability distribution HMM generalises the whole word template matching as used in automatic word recognition [3].

2.2. HIDDEN SEMI-MARKOV MODELS

In a Hidden Semi-Markov Model (HSMM) (or explicit state dwell HMM, or variable duration HMM), each state visit generates not one but a sequence of observations, and this sequence of observations is drawn from a distribution which depends on the state. As a simple example, consider a HSMM in which each state visit generates an output length, from a distribution characteristic of the state, then chooses each of the symbols in a string of that length independent of one another. If an ordinary (synchronous-output) HMM is allowed transitions from each state to itself, then it is equivalent to such a HSMM in which all the output duration distributions are geometric. HSMMs have been used to advantage in automatic word recognition [4].

2.3. RELATIONSHIP WITH FINITE STATE AUTOMATA

A Finite State Automaton is a system which has a finite set of internal states and can be specified by a state transition matrix and an output matrix. There are close links between probabilistic finite state automata (and their relation to stochastic formal grammars) and Hidden Markov models [5].

3. FORMAL GRAMMARS

3.1. INFORMAL INTRODUCTION

Formal Grammars (FGs) are used to model natural languages, to specify programming languages and to describe physical patterns and data structures. In addition to the descriptive function of FGs there is a generative function. The grammar can be used to generate sentences in a language (or patterns of data) from the grammar's rewrite rules or productions. Work in the area of grammatical inference has shown that sometimes these production rules can be inferred from enough examples of sentences which conform to the rules of the particular grammar [6].

3.2. FORMAL DEFINITION OF FORMAL GRAMMARS

For a Formal Grammar, G, all sentences that can be generated by G are contained in the set L(G), the language generated by G.

The standard notation is as follows :-

$G = (S, V_N, V_T, P)$

where

S is the start symbol $S \in V_N$

V_N is the set of non-terminal symbols : A,B,C

V_T is the set of terminal symbols : a,b,c

P is the set of production rules : $\alpha \rightarrow \beta$

where $\alpha \in (V_N \cup V_T)^*$ and $\beta \in (V_N \cup V_T)^*$

The Kleene star * denotes a repetition of the operator to which it is applied.

The language L(G) is formally defined as :-

$$L(G) = (w \in V_T^* \mid S \overset{*}{\rightarrow} w)$$

where w represents 'words' in the language.

When applied to natural language the terminal symbols are words and the non-terminal symbols are phrase markers, such as NP for noun phrase.
(The convention of denoting terminal symbols by lower case letters and non-terminal symbols by upper case letters will be used throughout.)
Grammars are in general ambiguous in the sense that different sets of production rules may result in the same string of terminals.

3.3. CONTEXT DEPENDENT GRAMMAR

If productions in G are of the form

$$aAb \rightarrow a\gamma b$$

where a,b are terminal symbols and

$\gamma \in (V_N \cup V_T)^*$

then the grammar is context dependent (or context sensitive).

Context dependent grammars are complex and rich enough to formally define many operations on strings of symbols (for example, context sensitive rules have often been used to write phonological rules).

3.4. CONTEXT FREE GRAMMAR

If productions in G are of the form

$$A \rightarrow \gamma$$

where $\gamma \in (V_N \cup V_T)^*$

then the grammar, G, is a context-free (CFG). Any sequence of terminal and non-terminal symbols can appear on the right, but the left hand side must consist of non-terminals only. As we shall see below, it is useful to transform a CFG into an equivalent CFG in a standard form (Chomsky Normal Form) in which the right hand side of each production consists of one terminal symbol or two non-terminal symbols. For example :-

A -> AB
A -> a

An algorithm for transformation of general context free rules into Chomsky Normal Form is given in [7].

3.5. REGULAR GRAMMAR

If the productions of G are of the form
A -> aB
A -> b (to end the process)
then the grammar, G, is regular.

EXAMPLE OF A REGULAR GRAMMAR
The following regular grammar describes the structure of real numbers with optional decimal point (as written in non-exponential form, e.g. -0.876). where

$S = S_0$

$V_N = (S_0, S_1, S_2, S_3)$

$V_T = (\phi, +, -, ., 0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$

and the productions P are :-

$S_0 \rightarrow \phi S_1$

$S_0 \rightarrow +S_1$

$S_0 \rightarrow -S_1$

$S_1 \rightarrow (\text{digit})S_1$

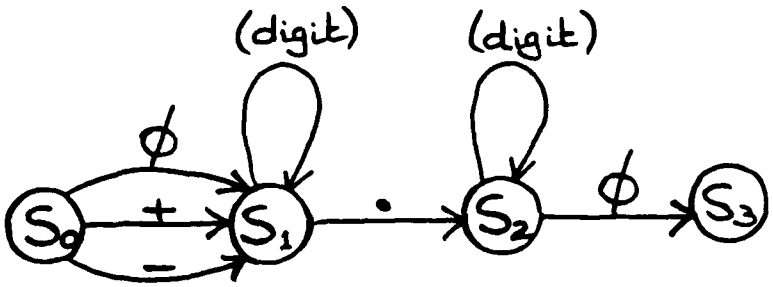
$S_1 \rightarrow .S_2$

$S_2 \rightarrow (\text{digit})S_2$

$S_2 \rightarrow \phi S_3$

(ϕ is the null symbol)

It can be represented as a network of states as follows :-



This example of a regular grammar involves recursion
(e.g. S_1-> (digit)S_1).

When the grammar of a language is recursive the language consists of an infinite number of strings. In this case the infinity is rather trivial. (We simply allow infinite length strings of digits before and after the decimal point.)

4. EXAMPLES OF CONTEXT FREE GRAMMARS AND PARSES

Context-free grammars include regular grammars. In a regular grammar each terminal symbol is directly related to a non-terminal, whereas in a context free grammar, each instance of a non-terminal symbol gives rise to a string of terminal symbols. Hence, the underlying structure of a context free grammar is hierarchical.

4.1. ARITHMETIC EXPRESSIONS

The following context free grammar describes the set of arithmetic expressions using addition and multiplication.

Let E denote the root non-terminal which represents the arithmetic expression. T denotes a term and P a multiplicative factor. The terminal symbols are: y, *, +.

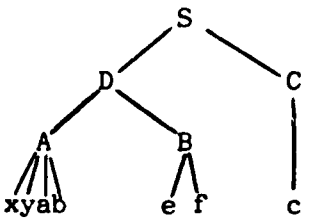
e.g. Expression = y + y*y

has the following productions :-

E -> T
E -> E + T
T -> P
T -> T * P
P -> y

4.2. TREE DIAGRAMS

The derivation of a string from the start symbol (S) by a context free grammar can be represented diagramatically in the form of a tree. The start node (at the root of the tree) is usually labelled S.



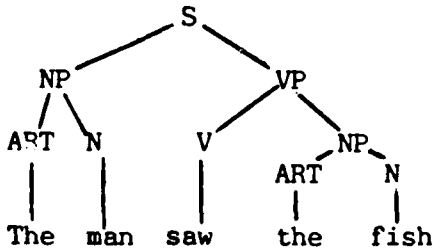
4.3. BRACKET EXPRESSIONS

An alternative notation, more suitable for a text processor, is to use parentheses around the substring derived from each non-terminal. If the non-terminal symbols are included, the resulting string contains enough information to specify a tree diagram (but may be more difficult to comprehend).

For example, the bracketed expression for the above tree diagram example is:-

S(D(A(xyab)B(ef))C(c))

A further example of bracketing is given for a simple English sentence with the following parse tree :



S(NP(ART(THE)N(MAN))VP(V(SAW)NP(ART(THE)N(FISH))))

4.4. NATURAL LANGUAGE

Context free grammars can be used to model simple sentences in natural language where the non-terminal symbols are phrase markers (e.g. noun phrases, etc.) and the words in the language are the terminal symbols. For example, the bracketed sentence in paragraph 4.3 may have been generated by a grammar of the form :

S -> NP VP
VP -> V NP
NP -> ART N
ART -> the
N -> man / fish
V -> saw

(In this case the non-terminal symbols N, V and ART would be called pre-terminals.)

5. A CHANGE OF NOTATION

Up to now we have been using the conventional symbol systems for formal grammars. It will help with the relationships with Markov models and algorithms if we switch to a different system, which is directly related to the notation introduced in 2.1.

We shall use upper and lower case x's for non-terminal symbols (NTSs) and y's for terminal symbols (TSs). Assume an arbitrary ordering of the terminal symbols, and separately of the non-terminals.

y(i) refers to the ith terminal symbol in the set,
x(i) refers to the ith non-terminal symbol in the set.

We shall use upper-case variables for the symbol numbers at particular positions in the output string:

Y(t) is the number (index) of the terminal at position t in the input. (Note that Y(t) is only a number - the actual symbol appearing at position t is y(Y(t)).

For Regular Grammars there is a NTS corresponding to each observed TS. Therefore we can refer to the NTSs in a derivation using a single index, thus:

X(t) is the index of the NTS which was at the end of the extending string immediately after the tth terminal symbol (Y(t)) had been produced. Y(t) must have been produced by productions of the form:

x(X(t-1)) -> y(Y(t)) x(X(t))

For example, if abracaA is transformed into abracadB, then Y(7)=4 and X(7)=2 (assuming alphabetic ordering of TSs and NTSs).

For convenience, define x(0) as the the start symbol. Thus X(0)=0.

In the case of CFGs it is more difficult to refer to the non-terminals in a derivation of a known string in terms of positions in the string of TSs, because of the hierarchical structure of non-terminals. The solution (which is applicable to an important class of CFGs) is to use a double index, to refer to the substring produced by the NTS:

X(s,t) is the index of the NTS which produces the sub-string Y(s),Y(s+1),....Y(t).

This index system, which is important in several algorithms described below, is applicable to CFGs in Chomsky Normal Form. All productions are of one of two kinds :-

```
x(i) -> x(j)x(k)
x(i) -> y(k)
```

6. STOCHASTIC GRAMMARS

The stochastic generalisation of a conventional (formal) grammar is called a stochastic grammar. In a stochastic grammar every production rule has a probability associated with it. The probabilities for all the productions with the same left hand side sum to unity.

Associated with any derivation of a string is a probability which is obtained by multiplying together the probabilities associated with all the productions used in the string derivation.

The probability of the grammar producing a given string is the sum of the probabilities of all the derivations of the string.

6.1. STOCHASTIC REGULAR GRAMMARS

Each production rule is either of the form

```
          p(ijk)
x(i) ---> x(j)y(k)
```

or

```
          p(ik)
x(i) ---> y(k)
```

where the probabilities p(ijk) and p(ik)(for a given i) must sum to unity.

The probability of a given string, Y(1)...Y(N), produced via NTSs X(1)...X(N-1) is

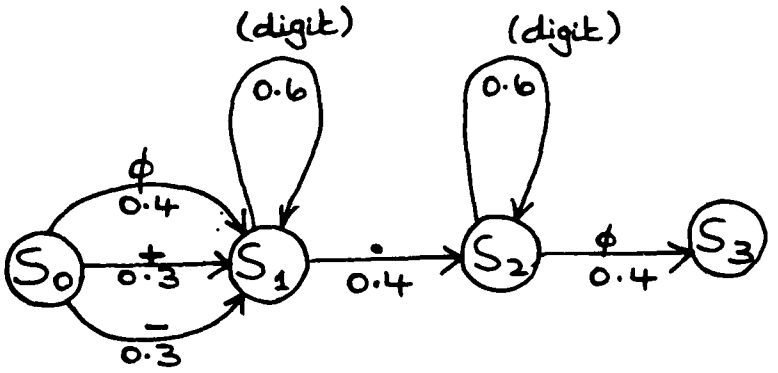
$$\prod_{t=1}^{N-1} p(X(t-1),X(t),Y(t)) \; . \; p(X(N-1),X(N))$$

In the same way stochastic context free grammars (SCFGs) are simply CFGs with probabilities attached to each production rule (see Section 6.3).

6.2. RELATIONSHIPS

Formal grammars of the class called regular grammars are equivalent to finite state automata in the sense that the same set of strings of symbols can be defined by both RGs and finite state automata [5]. It can be shown that a language is regular if and only if it is defined by a finite state automaton.

The example of a regular grammar to generate real numbers can be extended by putting probabilities on the productions to illustrate a stochastic regular grammar.



Stochastic regular grammars are equivalent to probabilistic finite state automata and to hidden Markov models, in the sense that stochastic grammars can be thought of as doubly stochastic processes. The hidden process rewrites the non-terminal symbols while the observed process produces the terminal symbols.

In a stochastic regular grammar the ith non-terminal symbol maps onto the jth non-terminal symbol and the kth terminal symbol (observation) with probability p(ijk):

$$x(i) \xrightarrow{p(ijk)} x(j)\ y(k)$$

where

$$p(ijk) = a(ij).b(ijk)$$

a(ij) is the transition probability between states i and j and b(ijk) is the probability of the observation k given the transition from state i to state j.

Referring back to the outputs-on-transitions type of HMM (in Section 2) it can be seen that the ith and jth non-terminal symbols in the SRG are equivalent to the ith and jth hidden states in the HMM. Also the kth terminal is related in the same way as the kth observed output string in

the HMM. In the same way the probabilities, a(ij) and b(ijk), are equivalent for HMMs and SFGs.

6.3. STOCHASTIC CONTEXT-FREE GRAMMARS

Since CFGs include RGs, SCFGs include SRGs and thus HMMs in the following manner :

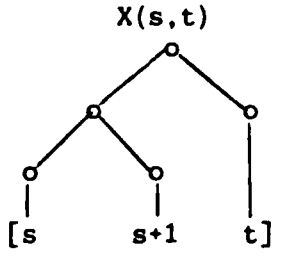
With a SRG the underlying stochastic process can be thought of as a sequence of random variables, each of which is related to a particular observation. This relationship for SRGs can be extended to SCFGs by considering the hidden stochastic process as a hierarchical family (rather than a simple sequence) of random variables X(s,t) where X(s,t) is associated with a substring interval (or span) of terminal symbols Y(s),Y(s+1),...,Y(t).

More formally :

X(s,t) is the non-terminal which 'is responsible' for producing the entire sequence of observations Y(s),Y(s+1),...,Y(t). The probabilities which describe a SCFG are:

a(ijk) = the probability that the NTS i generates NTSs j and k.

b(jk) = the probability that the NTS j generates the TS k.



Baker's "nodal span" principle makes it possible to generalise the techniques used in Hidden Markov modelling so that they can be applied to SCGFs. In particular, Baker's algorithm, known as the Inside/Outside algorithm, provides a method of reestimating the probabilites of the SCFG productions from terminal strings which have been generated according to SCFG rules. This algorithm is a generalisation and extension of the Forward/Backward algorithm used for reestimation of probabilities in HMMs [8].

SCFGs may be useful as language models for Automatic Speech Recognition (ASR) using naturally-occurring English. In particular it should be possible to extend the current HMM techniques currently used in ASR to include syntactic constraints using SCFGs. One disadvantage of SCFGs is that they rely on a well defined set of production rules which are difficult to specify for natural language.

7. SUMMARY OF MAIN ALGORITHMS FOR STOCHASTIC FORMAL GRAMMARS

For completeness we present here the main algorithms for stochastic regular and context free grammars.

7.1. PATTERN GENERATION

The most natural way to operate most stochastic models is to synthesise an instance of a pattern of a given class (driven by a random number generator). For HMMs the generator is obvious enough: choose a new state with a probability determined by the old state and the state transition matrix, then generate an "observation" by sampling from the appropriate output distribution. For more general Markov Random Fields (MRFs) we can use the Gibbs Sampler [9].

7.2. PATTERN RECOGNITION

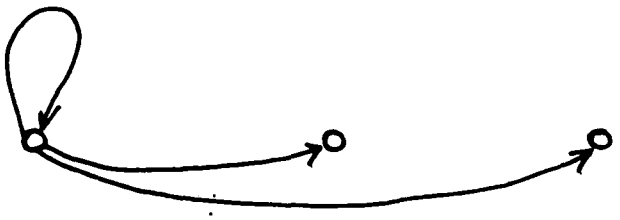
In pattern recognition we infer an explanation of the data in terms of the model. Typically we wish to maximise the probability of the explanation given the observations. Dynamic programming can be used to find the best (most likely) explanation of the data in terms of a given model for HMMs and SCFGs. The forward-backward algorithm for HMMs will compute the total likelihood of the data given a model, and also likelihood of each state at each time [10]. The corresponding algorithm for SCFGs [8] has been called the Inside-Outside algorithm. For more general MRFs the Gibbs Sampler can be applied to sample from the posterior distribution. The maximum a-posteriori interpretation can be found by sharpening this distribution slowly using the control parameter, T (ie Optimisation by Simulated Annealing [11]) .

7.3. PARAMETER TUNING

The other important automatic operation is tuning the model: probabilities or parameters of distributions are adjusted to increase the fit of the model (generated distribution) to the training data (observed distribution). Available algorithms include the Baum-Welch iteration for HMMs, which uses the results of the forward-backward algorithm [10]. The Boltzmann Machine Learning Algorithm [12] will tune the weights of a binary-valued, pairwise-interacting MRF expressed in Gibbs form, and extensions apply to more general MRFs.

7.4. MODEL SPECIFICATION

Finally, we must consider what means exist to allow us to set up the structure and initial values of the parameters based on our knowledge, insight, and predjudices. It is usual to specify the bare minimum when using HMMs (number of states, and form of output distributions). If some values of the state transition matrix are initially zero they will remain zero in the Baum-Welch iteration. One common state transition matrix restriction is tri-diagonal: the states are ordered, and each transition is to the same state, next state or next-but-one:



Other means of using 'a priori' knowledge in existing models form a major area for study in many types of ASR and image understanding research.

REFERENCES

[1] E.T. Irons. "A Syntax directed compiler for ALGOL 60". Comm.ACM 6:11, 1961.

[2] J.K. Baker. "The Dragon System - An Overview". IEEE Transactions on Acoustics, Speech and Signal Processing, Vol ASSP-23, No 1, February 1975.

[3] M.J. Russell. "Hidden Markov Models for Automatic Speech Recognition." Presented at the Leeds Experimental Phonetics Symposium, Leeds. September 1984.

[4] M.J. Russell. "Maximum liklihood Hidden Semi-Markov Modelling parameter estimation for Automatic Speech Recognition". RSRE Malvern Memo 3837, 1985.

[5] J.E. Hopcroft and J.D. Ullman. "Formal Languages and their relation to Automata". Reading, Mass. Addison-Wesley 1969.

[6] K.S. Fu and T.L. Booth. "Grammatical Inference : Introduction and Survey". IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 8 No 3 May 1986.

[7] A.V. Aho and J.D. Ullman. "The Theory of Parsing, Translation and Compiling - Volume 1: Parsing". Prentice Hall 1972.

[8] J.K. Baker. "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, 1979.

[9] S. Geman and D. Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." IEEE Trans. on Pattern Analysis and Machine Intelligence Vol PAMI-6, No.6, pp721-741, November 1984.

[10] F. Jelinek. "Markov Source Modeling of Text Generation". NATO Advanced Study Institute : Impact of Processing Techniques on Communications, MARTINUS NIJHOFF 1985.

[11] S. Kirkpatrick ,C.D. Gelatt, and M.P. Vecchi. "Optimisation by Simulated Annealing." Science pp 671-680 1983.

[12] J.S. Bridle and R.K. Moore. "Boltzmann Machines for Speech Pattern Processing." Proc. Inst. Acoustics Autumn Conference, Windemere, pp 315-322, November 1984.

DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... UNLIMITED .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	MEMO 4051		UNLIMITED

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
	RSRE Saint Andrews Road, Malvern, Worcs WR14 3PS

5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title

FORMAL GRAMMARS AND MARKOV MODELS

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
BRIDLE J S	DODD L		1987	14

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement

UNLIMITED

Descriptors (or keywords)

continue on separate piece of paper

Abstract

The theory of formal grammars is widely used in computer science and linguistics. Hidden Markov models are well established in automatic speech recognition. This expository memorandum sets out the links between the two areas, via stochastic grammars, and points to stochastic context-free grammars as an interesting area for practical application.

END

FILMED

MARCH, 1988

DTIC