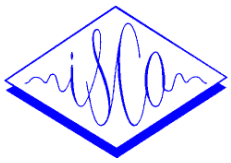

3rd International Conference on
Spoken Language Processing (ICSLP 94)
Yokohama, Japan
September 18-22, 1994

STOCHASTIC CONTEXT-FREE LANGUAGE MODELING WITH EVOLUTIONAL GRAMMARS

Michael K. Brown
Stephen C. Glinski

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Abstract — We describe an efficient implementation of a stochastic context-free language model suitable for real-time processing of speech and handwriting. Several similar prior efforts have some overlap with the methods we present, but in each case some shortcoming was overcome. This paper describes the first integration of a stochastic context-free language model with a stochastic acoustic model in a way that makes the language modeling overhead very small.

This is achieved through the use of an *Evolutional Grammar* (EG) consisting of both ephemeral (Markov) word models and grammatical rules which are dynamically expanded while processing speech input. The efficacy of this approach is tested with the ARPA Naval Resource Management speech application.

1. INTRODUCTION

One of the most important components of a high-performance speech or handwriting recognizer is a language model. This is now well known from prior work in speech recognition (for an early example see Levinson [1]. Consequently, many attempts have been made to incorporate more powerful language models in such recognizers. In particular, recent efforts to use Context-Free Grammars (CFG's) have been reported, but in each case there are shortcomings our system has overcome such as grammar size limitations or approximations, efficiency problems, or suboptimality.

This paper describes the implementation of a large-vocabulary phone-based speaker-independent connected speech recognition system that handles CFG's intrinsically in an *evolving* finite-state *grammar* (EG) using an *Ephemeral Hidden Markov Model* (EHMM) representation. We take advantage or our previous experience with finite-state recognizers by showing that for regular grammars the resulting CFG recognizer yields hypothesis scores that are identical to previous non-CFG recognizers whose performance has been thoroughly tested. Thus, the main contribution of this paper is not to show the recognition accuracy of one more speech recognizer, but to validate a a more powerful and efficient language modeling algorithm.

EHMM's are HMM's with fleeting lifetimes that are dynamically created and destroyed as needed in propagating HMM probabilities through a grammar network.† Each EHMM typically exists for less than a second at real-time processing rates An EHMM is created when its input probability is sufficiently large to potentially be a valid candidate on the correct grammar network path (hypothesis) for the current speech input. Processing of the EHMM then proceeds until the highest probability in the EHMM falls below a useful level, at which point the EHMM vanishes. Thus, computation time and memory consumption are inherently limited by pruning. This is a key feature that makes possible the processing of infinite grammars.

The amount of memory and computation required at any instant of time is dependent on the local entropy (or perplexity) of the grammar, the quality of the acoustic models, and the tightness of the pruning function. Because memory consumption is limited by local properties of the grammar, the global size of the grammar is unimportant.

CFG's with recursive definitions that would normally require infinite HMM representations or finite-state approximations [2] can be fully handled with an EHMM representation because the infinitely recursive parts of the grammar are only expanded to a depth limited only by hypothesis pruning. An underlying finite-state representation of the grammar *evolves* as the speech is processed, hence, the terminology *Evolutional Grammar* (EG).

A similar concept, called "dynamic network grammars", has been previously reported by Moore, et.al. at SRI [3] ‡. Their method used a unification parser and was capable of handling LR(0) grammars. Discussion with one of the SRI authors [5] revealed that the methods were not pursued, in part because of efficiency problems with the parser being used to drive the expansion of the grammar. Also, SRI was experimenting with the ARPA ATIS (Air Travel Information System) task at that time and the grammars being tested were too small and not well suited to that task. This, however, was a function of the language being modeled, not the mechanism. Their method would be useful in other applications. Our expansion method, in contrast, is very efficient, requiring only a small fraction of total processing time, and can provide all other features normally obtainable with finite-state grammars.

Several other methods have been reported in the literature for performing recognition with CFG's [4], [6]. Most of these apply post-processing to the output of a relatively unconstrained recognizer. The post-processor eliminates invalid symbol sequences according to a more restrictive grammar. Ney [7] (see also Young [8]) reports a system using an algorithm, based on the CYK parsing algorithm [9], that performs CFG recognition without conventional post-processing. Closer examination of the algorithm reveals that fine grained post-processing is performed at the frame level. First *all* scores for the current frame are computed for *all* words in the vocabulary using Dynamic Programming (DP). Then the scores are propagated to non-terminals in the grammar using an algorithm with complexity of $O(n^3)$ in the number of input frames. Thus, scores are not propagated according to the grammar at the DP level, but at higher symbolic levels, an interleaved form of post-processing. In contrast, our method builds the CFG constraints directly into the DP process, eliminating grammatically impossible DP's before acoustic processing, with a computational complexity of $O(n)$ in the number of input frames.

The next section describes the implementation of EHMM's in more detail and discusses EG representation and construction. Section 3 briefly covers implementation issues. Validation of the modeling method and some recognition performance with the SNOR grammar are presented in Section 4. Finally we conclude with some remarks about future plans and other applications.

2. EHMM's AND EG's

Conventional beam searching methods limit the attention of the system to parts of the grammar having significant probabilities of a successful match to input speech. Initially, the recognition system starts with no EHMM's (in essence, no initial grammar). An initial input probability (or score, if unnormalized)† is applied to the grammar start or input node,

† To be precise, it is actually the *instance* of the HMM that is ephemeral. Strictly speaking *model* refers to the structure *description* rather than the actual structure. In our application, the distinction between the *model*, which is persistent, and the ephemeral *instance* is that the instance has additional dynamically created structure to hold score, duration and traceback information. We simply refer to these instances as EHMM's to avoid cumbersome terminology.

‡ The root of this concept can be further traced back to the PREDICT knowledge source in the Hearsay-II system [4].

† In the sequel, we will refer to these values as "scores" because, for efficiency reasons, we do not renormalize them in our implementation. Also, to avoid confusion, we will use "state" to refer to HMM states and "node" to refer to a grammar network state, which may

the only initially existing active node. Immediately, all EHMM's needed to represent all starting words in the grammar are created (i.e., memory is allocated into the EHMM data structure, cf. §2.1). Viterbi scoring [10] is then applied to propagate scores in the network. As output scores begin to appear from these EHMM's, additional successor EHMM's in the grammar network are instantiated. While this process continues, earlier EHMM's begin to decline in score value, ultimately being reduced to inconsequential levels, at which point these EHMM's vanish and the associated memory is released. Some of the EHMM output scores with the necessary trace-back information are temporarily stored until all input speech for a sentence is processed. Because of the localized nature of the search and pruning methods (cf. §2.2), the amount of trace-back information stored is small. Finally, trace-back through this data yields the recognition results.

2.1. EHMM Creation and Destruction Criteria

Several methods for determining satisfactory score pruning levels have been commonly used in beam searching algorithms, ratios to best score (or probability), and offsets from best score being most common. We have implemented a dual threshold using linear pruning functions of maximum grammar node score.

Let

$$s_{\max} = \max_{n \in G_a}(s_n) \tag{1}$$

where n is a node in the active part of the EG node set G_a, and s is a score. Then

$$s_{on} = \alpha_{on} + \beta_{on}\, s_{\max} \tag{2}$$

and

$$s_{off} = \alpha_{off} + \beta_{off}\, s_{\max} \tag{3}$$

where α and β are fixed parameters. These score constraints are used to prune lower scoring word models on the next speech frame. Maximum grammar node score typically (but not necessarily) increases monotonically throughout the processing of a valid sentence. Hence, for values of β less than unity, threshold constraints relax with increasing time, and for values of β greater than unity, threshold constraints tighten. Figure 1 and Figure 2 display data with tight pruning parameters of $\alpha_{on} = \alpha_{off} = 0$, $\beta_{on} = 0.99$, and $\beta_{off} = 0.98$.

2.2. Evolutional Grammars

The grammar rules used to build the EG are represented as Recursive Transition Networks (RTN's) [11], the underlying structural component of the more familiar Augmented Transition Networks (ATN's) [12]. RTN's are capable of representing all Context-Free Grammars (CFG's) [11]. RTN's are an efficient choice for implementing this system because they are already in the target representation format, thus requiring almost no transformation overhead and making real-time performance possible.

In the speech processing field, stochastic RTN's have been employed for extracting language modeling statistics [13] and in a limited-depth implementation of a CFG speech recognizer at NEC [14]. Our speech recognition system is capable of handling CFG's intrinsically at essentially unlimited depths in the EG, a substantial improvement over more commonly used post-processing methods [4], [15], [16], which are less efficient and potentially less accurate.

2.3. Cyclic CFG Definitions

One potential problem arises with left-recursive non-terminal definitions. A non-terminal may contain a reference to itself, or an indirect reference through a chain of networks leading back to itself. If a cyclic reference occurs at an input node, an infinite loop occurs in the EG expansion process. Consider the simplest case of a non-terminal expansion rule

$$A \rightarrow A\gamma \tag{4a}$$

where γ is an arbitrary non-null symbol sequence, possibly terminal (constant). There must, of course, also be some other expansion rule for A, such as

$$A \rightarrow \delta \tag{4b}$$

be thought of as the vertex of a directed graph (digraph). A digraph edge, or grammar state transition, will be referred to as an "arc".

where δ is another non-null sequence, not containing A as the leading symbol. Rules (4a) and (4b) express (part of) a valid CFG for an infinite language. This language contains sentences like "$\delta\gamma$", "$\delta\gamma\gamma$", "$\delta\gamma\gamma\cdots$", etc. The problem arises when trying to implement rules for A in the non-terminal index table for an EG. Because rule (4a) replaces non-terminal A in the non-terminal index table with A (and other grammar arcs), when reaccessed, A will again be replaced with A ad infinitum, never creating an EHMM.

The solution is to partially transform the CFG to *Greibach Normal Form* (GNF) [9]. GNF refers to a CFG where every rule is of the form

$$A \rightarrow a\gamma \tag{5}$$

where a is a terminal symbol (i.e. word from the vocabulary). All CFG's can be converted to GNF. GNF representations are generally much larger than Chomsky Normal Form (CNF) representations. Fortunately, we need not complete the entire transformation, only eliminate cycles on the first symbol of the expansion. Note that if γ contains A, there is no additional expansion problem as long as the first symbol of the expansion ultimately leads to an EHMM. The rest of the rule will not be expanded until later in the input sentence. The only requirement is that the expansion of a non-terminal eventually replaces all leading symbols (i.e., arcs from the current grammar node) with terminals.

Hopcroft and Ullman [9] describe a three-step algorithm for the transformation of a CFG to GNF. We are required to perform only the first step of this transformation, which eliminates the cyclic rules by combining and rewriting rules like (4) to form new rules. For example, the first step of the transformation applied to (4) yields

$$A \rightarrow \delta \tag{6a}$$

$$A \rightarrow \delta B \tag{6b}$$

$$B \rightarrow \gamma \tag{6c}$$

$$B \rightarrow \gamma B \tag{6d}$$

which happens to be in GNF if γ and δ are terminals or terminal sequences, but might not be if the rules were more complicated.

2.4. Grammar Compilation

Construction of large grammars of the form described here is quite tedious unless some type of compact symbolic representation can be used. Grammar Specification Language (GSL) [17] is designed specifically to address this problem. GSL expresses finite-state grammars in a natural language form. Indeed, an ordinary English sentence is a valid GSL statement. Disjunctive (exclusive-or) forms make the representation much more compact. A macro facility is also provided to simplify definition of commonly used forms. Alternatively, non-terminals may be defined for these common forms.

RTN's are formed by defining multiple sub-grammars in GSL. Each sub-grammar is assigned to a non-terminal symbol indicated in GSL text by a <...> delimited sequence of characters. Only one definition of each non-terminal symbol is allowed. Thus, multiple CFG rules like the two rules of (4) are combined into a single definition. Multiple grammars for context-switching in response to semantics are a trivial extension of the RTN construction.

Optimization of finite-state grammars was essential for implementation of large finite-state grammars in earlier real-time systems. The need for optimization is somewhat reduced in a system implemented with an EG and EHMM's because the local nature of grammar evolution results in only small incremental increases in grammar size. Most of the grammar is not expanded or processed, so global optimization is not a major issue.

Determinization of each sub-grammar is most important, however, because it eliminates duplication of instantiated EHMM's. A non-deterministic grammar is a grammar for which a node transition, given a particular transition symbol, is not unique. Thus, there is more than one successor node for a given symbol, requiring more than one identical EHMM to represent the grammar at that point. The grammar compiler employs a determinizer to eliminate this possibility.

Optimization still provides some improvement in processing speed because it reduces the size of the non-terminal index table and,

hence, reduces the time required to expand non-terminals. Non-terminal expansion overhead is small, so the savings is not as large as obtained in earlier finite-state implementations. There is also a small chance that a sentence path through the grammar will split and merge again, resulting in the duplication of EHMM's later in the processing. Again, this is not generally of great significance because the number of active EHMM's dwindles rapidly late in the sentence processing as scores along the correct path in the grammar begin to dominate over other path scores (cf. §2.1), thus eliminating most other paths.

3. SYSTEM IMPLEMENTATION OVERVIEW

The large-vocabulary recognizer architecture is partitioned into seven stages [18]. All computation is done in a single pass (frame synchronously) through the input speech signal, permitting the pipeline architecture depicted in the figure. The first four stages are devoted to signal processing. For purposes of validating the language model we use a simple acoustic model consisting of 12 cepstral, and 12 delta-cepstral features. For the DARPA-1000 Naval Resource Management (NRM) task, we use 1759 three-state speaker-independent context-dependent triphones.

We have implemented two versions of the recognizer. The first is a simulation written for research purposes in the C language on a multiprocessor Silicon Graphics machine. This system is specifically designed as a modular simulator to facilitate research. Performance on the order of 10 times slower than real time is achieved with this system for the full finite state grammar version of the NRM task. This version was used for obtaining performance data used in this paper.

The second implementation is on an AT&T Surf Board residing in a Sun Microsystems Sun 4/110 workstation. This version is also written in the C language, but all computationally expensive portions have been rewritten in assembly code for two DSP32C® signal processors residing on the board. Performance is currently around 10 times slower than real-time for the full finite state grammar, but it is anticipated that through further code optimization that less than 10 real time will be achieved.

4. SYSTEM VALIDATION AND RESULTS

We validated the accuracy of the new stochastic language modeling method by regressing the hypothesis scores obtained from the new system against an older baseline system with known performance. After final debugging, the hypothesis scores and segmentation results obtained from the new system on a limited test set were identical to the results of the baseline system in every respect. With this validation we can conclude that the performance of the two systems on large test sets will be identical. Of course, the new system has more language modeling power than the old system, so some additional testing was performed.

Using the research implementation described in §4, we tested several large grammar specifications and one recursive CFG specification (primarily to validate the EG concept for all CFG's) constructed from the DARPA-1000 SNOR grammar.

The four grammar forms described next were tested with various pruning functions and subsets of the testing data. The results shown first illustrate performance with relatively conservative pruning to yield higher recognition accuracy. It will be shown that relaxed pruning functions with these grammar forms can make computational effort appear to lose its dependency on perplexity (cf. Table 5.2), but the dependency can still be demonstrated with these grammars (cf. §5.2).

4.1. DARPA Naval Resource Management Task Grammar

The DARPA Strategic Computing Speech Recognition Program [19] was undertaken with the goal of providing a well-designed, realistic speech database which could be used both to develop and evaluate speech recognition technologies. The database was designed during 1986 with NBS (now NIST) in collaboration with TI, CMU, BBN, and SRI. The proposed task of the project was to model a naval resource management query task.

The SNOR grammar, written in LISP, was distributed by BBN in 1986. The LISP definition contains 132 non-terminals (including the main grammar) and 991 terminals. An examination of the LISP grammar definition reveals that the main grammar has 990 rules, and two of the 131 defined non-terminals (sub-grammars) are never used by the main grammar. The other non-terminals are predominantly very small sub-grammars containing only terminals. A small number of the non-terminals contain nested non-terminals. There are no recursive or cyclic non-terminals.

Four forms of the grammar were prepared in EG form: the word-pair grammar, the full finite-state grammar, the full grammar with non-terminals in place of redundant rules, and a direct translation of the original LISP definition. All of the full grammar definitions yield identical recognition results, far superior to that obtained from the word-pair grammar, but processing speeds vary.

The validation test set, which was not included in the training data, is taken from a speaker whose speech is known to be more difficult to recognize than that taken from most other speakers. The validataion set consists of 30 sentences, 230 words, and 8,904 frames (89 seconds) of speech. Some of these sentences have pauses, "ums", and other artifacts that make accurate recognition difficult.

The 30 validation sentences were processed with pruning parameters $\alpha_{on}=0$, $\alpha_{off}=-180$, $\beta_{on}=0.82$, and $\beta_{off}=1.03$. These parameter values formed moderately tight pruning functions that still accommodate the full 30 sentence test set, yielding the same recognition results as a full search. The test results are summarized in Table 5.2. Table column entries are: the grammar type (explained below), the number of non-terminal symbols expanded in the EG, the number of phone DP's performed per speech frame, the number of word recognition errors, and the recognition error rate.

Table 5.2: 30 Sentence Test Results

Grammar	NT Exp.	DP's/Fr.	Errors	Rate (%)
WPG	1	3040.8	51	22.2
FSG	1	3699.7	7	3.0
FSG w/NT's	1162	3699.7	7	3.0
SNOR	2882	4272.8	7	3.0

4.1.1. Word-Pair Grammar (WPG).

The word-pair grammar is a small finite-state version containing only one instance of each vocabulary word. Null-arcs are used to connect together any word-pairs that exist at least once in the full grammar.

4.1.2. Full Finite-State Grammar (FSG).

The full finite-state grammar was prepared from the original LISP grammar by expanding all originally defined non-terminals into the main grammar and constructing the finite-state representation, which consists of 62,245 nodes and 252,591 arcs. This finite-state grammar was then optimized using methods described by Brown and Wilpon [17], resulting in a finite-state grammar with 5,826 nodes, 29,759 real arcs, and 3,973 null-arcs.

4.1.3. Non-Terminal Definition of Redundant Rules (FSG w/ NT's).

The 5,826 node finite-state grammar is organized into "rules". A rule designates a source node, destination node, and a list of transition symbols (words) connecting those nodes. Many of these rules contain identical lists of words. For instance, 104 ship names are used, in both singular and plural form, in this grammar. Wherever a ship is referenced in the grammar, all ship names are included. This occurs in 153 places in the optimized finite-state grammar (532 places in the original LISP definition). Rather than define all 104 ship names in each of 153 locations in the grammar, we replaced these with references to two non-terminal definitions (one each for singular and plural forms). Similarly, 110 other redundant rule types were replaced with non-terminals. The resulting grammar contains 5,826 nodes in the main grammar, with 4,848 real arcs, 3,552 null-arcs, and 1,919 non-terminal references to the 112 non-terminals.

Since the resulting EG can evolve into the full 5,826 node finite-state grammar, the recognition results are identical to those of the full grammar.

4.1.4. Nested Non-Terminals from Original SNOR Grammar (SNOR).

The original LISP definition was translated directly into an EG definition by inserting uniquely numbered grammar nodes between each symbol (either terminal or non-terminal) in the LISP definition of each of the 132 non-terminals. The main non-terminal was determinized and minimized using the methods of Brown and Wilpon [17], but the remaining 131 non-terminals were used directly since they were small and already deterministic. Refer to Table 5.2 for results.

4.2. Recursive CFG

To validate the concept of EG's for recursive CFG's, including those with recursive non-terminals that could potentially expand without limit, we modified and added some SNOR non-terminals to make them recursive and tested this grammar using a sentence that would fire the new rules. While all of the previously discussed grammars were all ultimately finite-state, this recursive CFG is no longer finite-state.

Non-terminals were altered in the LISP definition and the resulting grammar definition translated to EG form, as in §5.1.4. The following example illustrates how the non-terminals are defined in LISP:

```
(DEF-CFG-RULE <CAT-X>
  ((CAT-2)
   (CAT-3)
   (CAT-4)
   (CATEGORY <234>)))
```

This is a simplification of the original BBN LISP definition, but is representative of the form used in our LISP definition. `DEF-CFG-RULE` is a function (actually a LISP macro) that takes two arguments: the name of the non-terminal being defined, and a list of rules, each of which is a list of symbols. In this instance, `DEF-CFG-RULE` defines the non-terminal `<CAT-X>` to represent the sub-grammar for four possible strings: "CAT-2", "CAT-3", "CAT-4", or "CATEGORY <234>". Non-terminals are represented by symbols with surrounding angle brackets, like `<234>`. The other symbols are all terminals, i.e., words from the 991 word vocabulary. Words like `CAT-2` are verbal abbreviations used by the the U.S. Navy. Thus, this single call to `DEF-CFG-RULE` defines all four CFG rules (cf. §3.2) for `<CAT-X>`.

We were interested only in validating the method for recursive rules in CFG's, not in altering the grammar definition in a way that would change the accuracy substantially. The rule for `<CAT-X>` was replaced by the following three rules:

```
(DEF-CFG-RULE <CAT-X>
  ((<CATEGORY-X> <OPTCAT-X>)))

(DEF-CFG-RULE <OPTCAT-X>
   ((<CAT-X>)
    (nil)))

(DEF-CFG-RULE <CATEGORY-X>
  ((CAT-2)
   (CAT-3)
   (CAT-4)
   (CATEGORY <234>)))
```

where `nil` is equivalent to a grammar null-arc.

This definition allows infinite repetition of the `<CAT-X>` rule. This grammar was tested with the following two sentences from the 30 sentence test set: "which carriers have category two casualty reports" and "does cleveland have any category two or category three problems". We anticipated that, with aggressive pruning, recognition of the later sentence with this new grammar might yield the following (erroneous) string: "does cleveland have any category two category three problems". This did not occur. Both sentences were correctly recognized using the recursive CFG definition.

4.2.1. Full NRM Test

In the course of conducting related experiments we have also tested the full Naval Resource Management test set of February 1989, which contains 300 sentences spoken by 10 speakers not used for training. The word accuracy was 98.2%. Even higher performance would be achieved by using newer acoustic models trained on a larger amount of data.

5. SUMMARY AND FUTURE PLANS

CFG Speech recognition systems with computational complexity comparable to that of finite-state systems have been implemented using "Ephemeral Hidden Markov Models" (EHMM's) and "Evolutional Grammars" (EG's). Our system has overcome shortcomings of prior systems such as poor efficiency and suboptimality. Real-time performance has been achieved in the grammar processing components of the system.

Since the system restarts with no instantiated grammar for each sentence, it is easy to apply semantic constraints to the evolution of the syntax for the next input sentence. This solves three long-standing problems: 1) it eliminates the grammar size barrier making possible real-time implementations with very large or infinite grammars; 2) it implements a context-free grammar intrinsically in the EG, rather than employing commonly used post-processing methods, which are less efficient and potentially less accurate; and 3) it makes the grammars dynamic on a time scale that permits the application of semantic constraints to the grammar syntax at the acoustic processing level, even in midsentence under some conditions.

6. ACKNOWLEDGEMENTS

We thank Chin Lee for providing the speech data and phonemic models used in this study and Maureen McGee for translating the original DARPA-1000 SNOR grammar into the fully expanded finite-state format.

REFERENCES

[1] S.E. Levinson and L.R. Rabiner, "A Task-Oriented Conversational Mode Speech Understanding System," in *Speech and Speaker Recognition*, M.R. Schroeder, Ed. Basil, Switzerland: S. Karger AG, 1985, pp. 149-196.

[2] F.C. Pereira and R.N. Wright, "Finite-State Approximation of Phrase-Structure Grammars," in *29th Annual Meeting of the Association for Computational Linguistics*, Univ. Cal. Berkeley, 1991, pp. 246-255.

[3] R. Moore, F. Pereira, and H. Murveit, "Integrating Speech and Natural-Language Processing," in *Proc. DARPA Speech and Net. Lang. Workshop*, Philadelphia, Feb. 1989, pp. 243-247.

[4] L.D. Erman and V.R. Lesser, "The HEARSAY-II Speech Understanding System: A Tutorial," in *Trends in Speech Recognition (Ch. 16)*, W.A. Lea, Ed. Prentice Hall, 1980, pp. 361-381.

[5] F. Pereira, Private communication. AT&T Bell Laboratories, Murray Hill, 1992.

[6] A. Paeseler, "Modification of Earley's Algorithm for Speech Recognition," in *Proc. NATO ASI Recent Adv. Speech Understanding, Dialog Syst.*, Bad Winsheim, Germany: Springer, July 1987.

[7] H. Ney, "Dynamic Programming Parsing for Context-Free Grammars in Continuous Speech Recognition," *IEEE Trans. on Signal Processing*, vol. 39, no. 2, pp. 336-340, Feb. 1991.

[8] S.J. Young, N.H. Russell, and J.H.S. Thornton, "Speech Recognition in VODIS II," in *Proc. IEEE ICASSP'88*, April 1988, pp. 441-444.

[9] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[10] L.R. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[11] J. Allen, *Natural Language Understanding*. Menlo Park, CA: Benjamin/Cummings Publ. Co., 1987.

[12] W.A. Woods, "Transition Network grammar for Natural Language Analysis," *Comm. ACM*, vol. 13, no. 10, pp. 591-606, Oct. 1970.

[13] J. Kupiec, "A Trellis-Based Algorithm for Estimating the Parameters of a Hidden Stochastic Context-Free Grammar," in *Proc. Speech and Net. Lang. Workshop, DARPA*, Pacific Grove, CA: Morgan Kaufmann, Feb. 1991, pp. 241-246.

[14] K. Yoshida, Continuous Speech Recognition Apparatus, U.S. Patent, No. 5,086,472. Feb. 4, 1992.

[15] M.D. Riley and A. Ljolje, "Lexical Access with a Statistically-Derived Phonetic Network," in *Proc. Speech and Net. Lang. Workshop, DARPA*, Pacific Grove, CA: Morgan Kaufmann, Feb. 1991, pp. 289-292.

[16] S.E. Levinson and A. Ljolje, "Continuous Speech Recognition from Phonetic Transcription," in *Proc. Speech and Net. Lang. Workshop, DARPA*, Cape Cod, Mass.: Morgan Kaufmann, Oct. 1989, p. 292.

[17] M.K. Brown and J.G. Wilpon, "A Grammar Compiler for Connected Speech Recognition," *IEEE Trans. on Signal Processing*, vol. 39, no. 1, pp. 17-28, Jan. 1991.

[18] M.K. Brown and S.C. Glinski, "Context-Free Large-Vocabulary Connected Speech Recognition with Evolutional Grammars," *IEEE ICASSP'94*, vol. 2, pp. 145-148, April 1994.

[19] P. Price, M.W. Fisher, J. Bernstein, and D.S. Pallet, "The Darpa 1000-Word Resource Management Database for Continuous Speech Recognition," *Proceedings IEEE ICASSP'88*, 1988.