APPROXIMATION TECHNIQUES FOR THE LEARNING OF SEQUENTIAL SYMBOLIC PATTERNS

Mark S. Fox and Frederick Hayes-Roth[1]
Computer Science Department[2]
Carnegie-Mellon University
Pittsburgh, Pa. 15213

ABSTRACT

The sequential symbolic pattern types to be learned are ordered sequences of sets consisting of pairs of symbols and weights. The weight defines the likelihood of that symbol occupying the position in the pattern determined by its set. Three methods are described for the learning and recognition of such patterns from training patterns containing substitution, deletion, insertion, and repetition errors. Two methods are based on a network model. Each training pattern is represented by a network (regular grammar). These separate networks are combined to form a single representative network to be used in the recognition process. The third method is based on a positional dependency scheme. This scheme constructs three-tuples from the training patterns of the form <symbol, weight, position in pattern> and combines the tuples into lists containing tuples of the same position. The techniques are applied to the learning and recognition of syllables and words from the HEARSAY II Speech Understanding System.

INTRODUCTION

The sequential symbolic pattern types to be learned are varying length ordered sequences of sets. The set defines what symbols were found to occupy a given position in the pattern. A set consists of pairs of symbols and weights. The weight defines the likelihood of that symbol occupying the position in the pattern determined by its set. Three methods are described for the learning and recognition of such patterns from training patterns containing substitution, deletion, insertion, and repetition errors. A pattern can be defined as follows:

P= <S1, S2, ..., Sn>
Si= {<si_1, wi_1>, <si_2, wi_2>, ..., <si_{n_i}, wi_{n_i}>}
where
P is a pattern
Si is the ith set in P
si_j is the symbol of the jth tuple in the set Si
wi_j is the weight of the jth tuple in the set Si

The domain from which we draw patterns is syllable recognition in the Hearsay II speech understanding system [1]. Using syllables as our basic patterns [2] the goal is to learn and recognize syllables from symbolic sequences produced by a segmenter and labeller. The segmenter divides a speech pattern into segments according amplitude and spectral criteria. It then attempts to match one or more labels to each segment by computing the Euclidean distance to learned phone prototypes.

Present segmentation and labelling schemes, using a 40 label set, operate at an accuracy level of 42% with a 65% chance of the correct label appearing in the top three choices [3]. Example 1 illustrates the differences between segmentation and labelling produced by a trained phonetician and that produced by a machine.

Given training data in the form of I.b for each syllable type, we seek a learning process that exhibits as much as possible the following recognition characteristics.

1) Robustness: should be insensitive to typical data errors.
2) Discriminability: must correctly distinguish between syllables (classes).
3) Speed: must enable fast recognition or be amenable to efficient match procedures.
4) Interpretability: the product of the learning process must be a simple representation that can be easily used and understood.

EXAMPLE I:

Example of human and machine generated segment and acoustic labels for the word "GIVE" which is an example of the syllable type "PIN".

<{<G, 1.0>},
{<IH, 1.0>},
{<V, 1.0>} >

I.a PHONETICIAN'S SEGMENTATION AND LABELLING OF SYLLABLE "PIN"

<{<-, 1.0>},
{<HH1, .336>, <K2, .336>, <T2, .328>},
{<IY1, 1.0>},
{<IH4, .256>, <IX2, .250>, <IH9, .250>, <IH1, .244>},
{<M1, .208>, <W1, .207>, <WW1, .195>, <V1, .195>, <N1,.195>}>

I.b MACHINE'S SEGMENTATION AND LABELLING OF SYLLABLE "PIN"

Related research has been carried out on the inference of grammars from patterns [4-6]. Given a positive (and possibly negative) set of training data, a grammar is constructed that generates a superset of the patterns. This grammar exhibits the basic characteristics of the training set (e.g., repetition of subsequences).

The characteristics of the patterns exhibited in the training data (symbol position, subsequence repetition, etc.) are all of equal importance in the inference of grammars. In our environment error is a major component of training and test patterns. The existence of errors in training data violates the principal assumption of the work in grammatical inference. Contrary to the error free induction problem faced by grammatical inference algorithms, our methods must infer reliable pattern definitions from very errorful data.

There has been recent interest [10] in the recognition of syntactic patterns with substitution, deletion, insertion and repetition errors. This central problem in this research concerns the modification of pre-defined stochastic grammar. It does not approach the subject of how to choose or construct the grammar to be modified. It is this grammar with most of the modifications present that we want to infer with the Network Model.

Previous attempts have been made at solving this problem. The learning paradigm used by Hayes-Roth and Burge [7] constructs pattern templates from conjunctive abstractions of feature descriptions. The resulting templates are used in a wholistic approach to pattern recognition. A wholistic approach utilizes all-or-none pattern matching. If the pattern to be recognized does not contain all the features defined in the template, the match fails. Our methods deviate from the above by using approximating techniques derived from an atomic view of matching. The representation produced by our learning process defines probable elements that may be found in the test pattern. A goodness of fit metric is produced by matching the learned probable elements with the elements contained in the test pattern.

Smith [8] has also worked on the problem of syllable

---

1 Present address: RAND, 1700 Main Street, Santa Monica, Ca. 90406

2 This research was supported in part by the Defense Advanced Research Projects Agency under contract no. F44620-73-C-0074 and monitored by the Air Force Office of Scientific Research. The first author is also supported in part by the National Research Council of Canada.

recognition. Our methods differ in that Smith utilizes domain dependent knowledge to develop specific vowel-centered syllable representations, whereas our algorithms are domain independent. We do not utilize any knowledge of syllables within our methods.

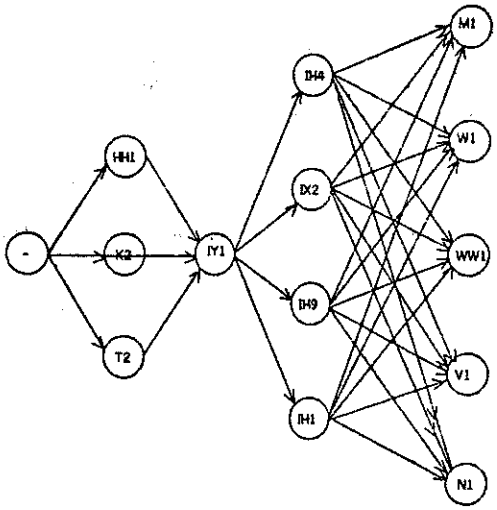
1.0 NETWORK THEORETIC MODEL.

We can represent a class of symbolic sequences (syllable) by a directed network. A node represents a symbol and an arc represents the relation "precedes" with a corresponding weight. Any path through the network determines a sequence in that class. We recognize a test pattern if there exists a path through the network for that pattern. We construct the network by first constructing a separate network for each training pattern and then combining the networks into one.

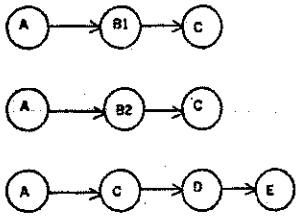
The following describes how the networks are combined once they are constructed from the training patterns. Subsequently, two methods are described for constructing a network from a training pattern.

A pattern can be represented by a path where the nodes represent labels and the arcs represent the relation "precedes" (Example II). A category is a set of symbols that can occupy the same position in the pattern. If we have several patterns, we would represent each by a separate path (Example III).

EXAMPLE II.a: NETWORK OF EX. I.a



EXAMPLE II.b: NETWORK OF EX. I.b



EXAMPLE III: NETWORK REPRESENTATION OF 3 TRAINING PATTERNS (WITHOUT CATEGORIES).

We represent a class of symbolic sequences as a directed network

$$N<V(N), A(N)>$$

where V(N) is the set of vertices in the Network N and A(N) is the set of arcs. Arcs are represented as two-tuples composed of a pair of nodes (symbols) and their corresponding weight. For example, $<<s_i, s_j>, w_{i,j}>$ is an arc specifying a transition from s_i to s_j and $w_{i,j}$ is the weight assigned to this transition. A path through the network represents a symbolic sequence whose probability is dependent upon the weights of the arcs along the path.

Let C_i be the ith class.
Let S be a symbolic sequence.
Then $Path(S) = s_1 s_2 s_3 \ldots s_n$.
Let $w_{i,j}$ be the weight for the transition from s_i to s_j.

$$Pr[Path(S) \mid C_i] = f(w_{1,2}, w_{2,3}, \ldots, w_{n-1,n}).$$

Given a training set T of symbolic sequences we want to construct a network N(T) that represents the class C(T). The construction is carried out by collapsing the training data paths into a Network. By collapsing we mean the superimposition of partial paths which contain identical nodes.

IF $s1_i, s1_j \in V(S1)$ and
$s2_k, s2_m \in V(S2)$
THEN
$V(N(T)) = V(S1) \cup V(S2)$
$A(N(T)) = <<s1_i, s1_j>, f(w1_{i,j}, w2_{k,m})>$
if $(\exists i)(\exists j)(\exists k)(\exists m)((s1_i = s2_k)$
$\wedge (s1_j = s2_m)$
$\wedge (<<s1_i, s1_j>, w1_{i,j}> \in A(S1))$
$\wedge (<<s2_k, s2_m>, w2_{k,m}> \in A(S2)))$

OTHERWISE $A(N(T)) = \{<<s1_i, s1_j>, w1_{i,j}>,$
$<<s2_k, s2_m>, w2_{k,m}>\}$

One case still remains to be handled. If we have two training examples

abc efc

we infer the transitions

a-->b b-->c e-->f f-->c

We also want to infer the cross-over transitions

a-->f e-->b

To do this, we store positional information in our network representation.

We define a node $s_j = <\alpha, p>$ where α is a symbol and p is the position of the symbol in the original training pattern

When using the previous rules we ignore the position factor. The following rule is added.

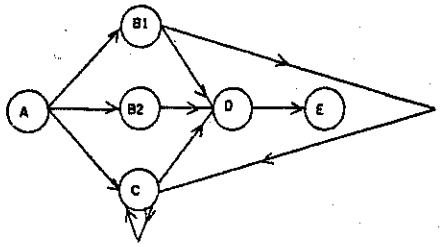
We add the arc

$<<s1_i, s2_m>, h(w1_{i,j}, w2_{k,m})>$
if $<<s1_i, s1_j>, w1_{i,j}> \in A(S1)$
$\wedge <<s2_k, s2_m>, w2_{k,m}> \in A(S2)$
$\wedge (p1_i = p2_k)$
where $s1_i = <\alpha, p1_i>$ and $s2_k = <\beta, p2_k>$

An alternative method using grammars is described in Method II under Ordinal Transitional Grammar.

The "collapsed" network in Example IV shows the regular paths found in all the patterns. It also indicates alternate arcs and paths that can be taken in the representation of that class. The "collapsed" network method achieves both the identification and representation of regularities. It allows the inclusion of less likely but nevertheless probable features of that class.

We turn to formal language theory to represent the Network and allow its use in the recognition process.



EXAMPLE IV: COMBINED NETWORK OF EXAMPLE III.

1.1 METHOD I.

1.1.1 NETWORK METHOD I.1: LLG

Each pattern can be represented by a stochastic left linear grammar that generates only that pattern. A stochastic grammar is a grammar with a weight associated with each production.

e_1 = abc

g_1 =

S-->aA	w11		S-->a[A0]
A-->bB	w12	OR	[A0]-->b[B0]
B-->c	w13		[B0]-->c

As can be seen, the construction of the grammar is simple where the right side of the production is the terminal symbol corresponding to the non-terminal in the right side of the production followed by a non-terminal or nothing depending on whether the terminal is or is not not the last symbol in the pattern string. The start symbol produces the terminal and non-terminal that corresponds to the first terminal in the example string.

A training pattern containing symbol duplications will precipitate a cycle in the grammar. This is avoided by creating a unique non-terminal for each terminal in the pattern.

e_2 = abac

g_2 =

S-->aA		S-->a[A0]	w21
A-->bB	OR	[A0]-->b[B0]	w22
B-->a[A1]		[B0]-->a[A1]	w23
[A1]-->c		[A1]-->c	w24

When a grammar is constructed for each pattern it is combined with the previous grammars to get a grammar G that covers all the g_i i=1, ..., n.

i.e., $G=g_1 \cup g_2 \cup \ldots \cup g_n$

The union of two grammars is like set union where each production is an element of the set, except the weights associated with duplicate productions are summed to get a new weight for that production.

$g_1 \cup g_2$ =

1	S-->a[A0]	w11 + w21
2	[A0]-->b[B0]	w12 + w22
3	[B0]-->c	w13
4	[B0]-->a[A1]	w23
5	[A1]-->c	w24

The result of the union is a left linear grammar G = (Vn, Vt, P!, S) where

Vt = the set of symbols in the pattern.
Vn = the set of symbols generated by the above method.
P! = <P, W> where P is a production with weight W.
S = start symbol.

A stochastic left linear grammar is obtained by dividing the weight of a production by the sum of the weights of production with the same left side of the production.

The maximum likelihood path through the grammar denotes the characteristic pattern for that class (most common syllable pronunciation).

The above representation carries with it almost all the information contained in the patterns. The resultant grammar of the union operation may be of enormous size. This size can be restricted by choosing a suitable threshold frequency (probability) and deleting those productions below that threshold.

The first method postulated a mechanism where a unique non-terminal was constructed for each terminal within a string. These non-terminals must correspond between strings. A problem arises when a terminal is found in more than one place in two or more patterns.

e.g., abacd abdac

In the above the ac's in both would be mapped onto each other; i.e., the production [A1]-->a[C0] would have frequency of 2, even though it occurs in two different contexts. This is a prime example of the power of this method. The insertion of d does not reduce the significance of ac as a substring of the underlying pattern.

1.1.2 METHOD I.2

To include context in the left linear grammar, we construct the following string with new non-terminals.

e_1 = abc

$g_{2,1}$ =

S-->a[SA]
[SA]-->b[AB]
[AB]-->c

The first symbol in the non-terminal on the left side corresponds to the terminal printed in the production two previous to it. The second symbol corresponds to the terminal symbol generated in the immediately preceding production. On the right side of the production, the first symbol in the non-terminal is the terminal symbol printed in the previous production and the second symbol is the symbol to be printed in the next production.

This method includes a context of two previous symbol (second order Markov Model), thus decreasing ambiguity in the inferred grammar. But the problem of repetition of the same terminal occurs here also with sets of two symbols. This method can be extended so that the non-terminal reflects an nth-order Markov process assumption.

1.2 METHOD II.

To facilitate the learning of categories we must add the previously discussed cross-over transitions of productions to the grammar of Method I. To do this we allow ordinal information to be carried in the grammar.

Ordinal Transition Grammar

One of many possible methods is to construct what we will call an Ordinal Transition Grammar (OTG). In essence we take a pattern and change it so that ordinal information is included.

e.g., abc is changed to <T1>a<T2>b<T3>c<T4>

The OTG is constructed like the previous left linear grammars with the exception that the terminal associated with the last <Tn> is blank.

e.g., abc <T1>a<T2>b<T3>c<T4>

S-->a[A]	[T1]-->a[A]
[A]-->b[B]	[T2]-->b[B]
[B]-->c	[T3]-->[C]
	[A]-->[T2]
	[B]-->[T3]
	[C]-->c[T4]
	S-->[T1]

An OTG is produced for each pattern. These OTG's are then combined by what we call a Transitive Convolution. Given two productions [A]-->[T2] and [T2]-->f[F] the transitive convolution will produce [A]-->f[F]. Each OTG is convolved with every other OTG. The transitive convolution of n patterns is an n squared operation.

After all convolved grammars are constructed, any productions still containing the positional information are deleted from their respective OTG. The OTG's are then combined with the grammars constructed in the first part of this outline.

The procedure is now composed of three steps:

1) construction of both the left linear grammar and the ordinal transition grammar;
2) the transitive convolution of the OTG's;
3) union of all the grammars to obtain the resultant stochastic grammar;

The resultant grammar can be used to construct a stochastic finite automaton which accepts members of its class.

2.0 POSITIONAL DEPENDENCY MODEL

2.1 METHOD III

Where the Network theoretic model recognizes transitions from any symbol to another as the primary information, the positional dependency model recognizes the position (time) of a symbol as the primary information.

By grouping symbols according to their position in the patterns, we define a category of symbols for each time.

IF S1, S2, ..., Sn are patterns where $Sj = <Sj_1, Sj_2, ..., Sj_{n_j}>$
and C1, ..., Cm are categories
THEN
$Ci = \{s1_i, s2_i, ..., sn_i\}$

If we associate a weight with each symbol in the pattern then

$Sj = <<sj_1, wj_1>, <sj_2, wj_2>, ..., <sj_{n_j}, wj_{n_j}>>$
and
Ci =
$<ci_j, f(w(ci_j), wk_i)>$ if $(\exists j)(\exists k)(ci_j = sk_i)$
$\wedge (<ci_j, w(ci_j)> \in Ci)$
$\wedge (<sk_i, wk_i> \in Sk)$

OTHERWISE $C_i = <sk_i, wk_i>$
NOTE: f(a, b) is a function for summing two freq.

Recognizing the inherent error in training examples and data to be recognized, we allow the inclusion of a symbol in categories forward and behind in time within an error tolerance proportional to the original position of the symbol.

$<sj_k, wj_k>$ is added to Ci if $|k-i|<\alpha(k)$

The resulting categories are then used in the recognition process as follows. Each symbol in the test pattern is matched with a symbol in the corresponding time category. The probability of accepting that sequence is a function of the weights from each category.

ie., Let S be the test pattern than

$Pr[S \mid Ci] = f(w1_{k1}, w2_{k2}, ..., wn_{kn})$
where $(\forall i)(\exists j)(si = Ci_j)$

3.0 THE EXPERIMENT

A program was written which implements the following methods:

1) Network theoretic model; LLG 1.1
2) Network theoretic model: LLG 1.1 and OTG
3) Positional dependency model

Input to all programs was in the form of patterns containing multiple probabilistic choices for each position in the pattern. An addition was made to both I and II so that the same type of symbol movement as found in III can be made before transitions were formed.

LLG 1.1

The weight of each production was defined as the product of the weights of the two symbols represented in the production. OTG cross-over was carried out within pattern but not between patterns.

LLG and OTG

The same computation as in (1) was performed with the addition that inter-pattern cross-over was carried out. The weight of each production formed in this manner was weighted by a parameter.

Positional Dependency Model

Weights of identical symbols within a category were additively combined to form a new single weight. The temporal generalization of a symbol was decided by a triangular function where $f(n/2)=\alpha n$ and $f(1)=f(n)=\alpha$, n is the sequence length and α is a parameter. The weight of the symbol was the product of the original weight and a parameter p raised to the power of the distance from the symbols original position.

e.g., If si placed in jth position then the new weight is $wi*(p^{|i-j|})$

In all three cases, weights are normalized to a probability ≤ 1.0. In the first two methods, productions having the same left nonterminal are normalized. In the third method normalization is carried out within categories.

3.1 RECOGNITION PROCESS

With any representation careful attention must be paid to how the representation is to be used in the recognition process. The recognition process must utilize all the information in the learned representation.

In all three methods, we compute the following conditional probability Pr[S | Wi]: the probability of the test sequence given it is in the class of word i. Used by itself, this denotes the degree of confidence the acceptor has of the sequence being in class Wi. This metric is usable in the PANDEMONIUM sense [9] of acceptance by which acceptor shouts the loudest. In addition we compute Pr[Wi | S], the probability that we have the class Wi given the input sequence S. This is computed by

$$Pr[Ci \mid S] = \frac{Pr[S \mid Wi]\, Pr[Wi]}{Pr[S]}$$

If $S=s_1 s_2 ... s_n$, we assume a Markov process and estimate Pr[S] as

$Pr[s_2|s_1]*Pr[s_3|s_2]*...*Pr[s_n|s_{n-1}]$ where

$Pr[s_{i+1}|s_i]$ data is computed from all methods by recording all transitions and weights and normalizing to 1.0. Since the occurrence of each pattern in our testing is equiprobable, Pr[Wi] was set to 1.0. In the Hearsay II Speech Understanding system, this parameter can be set by using other knowledge obtained from other sources in the system.

Network Theoretic Model

Pr[S | Wi] is computed using two methods:

1) Exhaustive Parallel: All paths are followed. The probability of the union of all paths that lead to an end state in the grammar (accepting state in the finite automaton) is found [6].
2) Best-n-First: the n most probable paths are followed in the grammar.

In both cases after a symbol is read in the probability of path that is presently in an end state is printed.

Positional Dependency Model

As each symbol is read the corresponding category list is searched and the weight retrieved. The product of each weight from each category is computed. If the corresponding category contains an end of sequence marker, the computed probability defines a possible recognition strength.

In all recognizers, if a transition or symbol is not found a default weight is used (0.0001).

4.0 TEST DATA

The data to be learned and tested were created by the segmenter and labeller in the Hearsay II Speech Understanding system. Syllable boundaries were approximated by comparing the machine generated file with a segmentation and labelling of the same speech by phonetician. Due to the approximate nature of the boundaries, many syllable patterns contain labels from preceding and following syllables. This increases the error in the training and test patterns. Two sets of tests have been carried out. Test set I contains 12 monosyllabic words. Test set II contains 19 syllables. The training set for each syllable ranged

from 1 to 16 patterns. The testing data were not used in training patterns.

The following experiment was run on each of the two sets of test data. Methods I, II and III were applied to each of the syllable categories in the training data. The resultant learned representations were used by a standard recognition program to recognize test exemplars in the learned categories. Thus there logically existed a recognizer for each category in the training set. The recognizer for each learned representation was applied to the same test datum. The datum was defined as being a member of the category of the recognizer that rated it highest.

Each of the learning and recognition methods were tested with various parameter settings. Both the Exhaustive Parallel and Best-n-First recognition procedures were used with learning methods I (LLG) and II (OTG).

5.0 RESULTS

Following are the best results for each method:

TEST I:

Method I: 12 out of 12 correct = 100%
Method II: 12 out of 12 correct = 100%
Method III: 11 out of 12 correct = 91%

Test II: (() denotes correct in top three)

Method I: 14 (17) out of 19 correct = 73% (90%)
Method II: 15 (17) out of 19 correct = 79% (90%)
Method III: 14 (18) out of 19 correct = 73% (95%)

The degree of success in Test I is due in part to the highly distinguishable characteristics of the patterns for each of the monosyllabic words. Test II contains a larger number of less differentiable patterns. As a result, the success rate of each method decreases but does not dip below the accuracy of the segmentation and labelling scheme. We believe that an increase in the recognition rate for Test II can be attained by further tuning of parameters in both the learning and recognition process of each method.

The results of Test II can be compared to those available of Hayes-Roth and Burge [7]. Their tests are based upon a subset (12) of the 19 syllables our tests used. Reevaluating our results without the seven syllables results in a recognition rate of 10 out of 12 for correct first choice with Method II, as compared to 11 out of 12 in the tests of Hayes-Roth and Burge. The difference in results is not significant.

6.0 CONCLUSION

The present results show that all three methods achieve almost the same rates of success. No significant improvement of Method II (OTG) over Method I (LLG) has been found. We believe this is due to the task environment we work in. A training pattern contains enough variation that we essentially do not gain more information from inter-pattern cross-over than was found in the intra-pattern cross-overs.

In both the test sets, the Exhaustive Parallel and Best-n-First do not significantly differ. Selection of the best method will require further testing.

The recognition rates of all three methods are greater than the accuracy of the segmenter-labeller. This implies that both the Network and Positional Dependency methods are appropriate models for this and other errorful environments. The success of each method is due to its ability to handle effectively each of the characteristic errors.

Through the learning of the more probable (higher frequency) subpaths the Network Model is able to learn in the presence of insertion, substitution and deletion errors. The existence of both alternate paths and subpaths allows this learned model to be used for recognition in the same environment that the learning took place.

The Positional Dependency method, through it explicit use of positional categories handles the substitution errors quite well. When temporal generalization is added, the model performs well in the presence of insertion, deletion and repetition errors.

The three methods are not only capable of learning the characteristics of the training data but are able to infer new symbol transitions, positions, and categories by means of the cross-over transition technique and temporal generalization of a symbol within its original pattern. This allows the recognition of patterns containing error traits not found in the training data.

Further testing and parameter tuning need to be carried out before optimal recognition rates are achieved. Higher order Markov assumptions in the construction of the grammar in the Network Model remain to be tested. In addition, an analysis of the effect of sample size on performance should prove to be very useful. Because these methods employ approximate methods of representation and pattern matching, it is anticipated that they will be effective even when training data are few.

REFERENCES

1. Erman, L.D. "Overview of the Hearsay Speech Understanding Research, " Computer Science Research Review, 1974-75, Computer Science Department., Carnegie-Mellon University, Pittsburgh, Pennsylvania.
2. Fujimura, O. "Syllable as a Unit of Speech Recognition, " IEEE Symposium Speech Recognition, Pittsburgh, Pennsylvania, April 1974, 148-153.
3. Goldberg, H.G. "Segmentation and labelling of speech: A comparative performance evaluation, " Ph.D. Thesis, Computer Science Department., Carnegie-Mellon University, Pittsburgh, Pennsylvania., 1975.
4. Biermann, A.W. and J.A.Feldman, "On the synthesis of finite state acceptors, " Stanford AI Project Memo, AIM-114, Stanford University, Stanford, California., 1970.
5. Horning, J.J. "A study of grammatical inference, " Technical Report No. 65-139, Computer Science Department., Stanford University, Stanford, California., 1969.
6. Fu, K.S. Syntactic Methods in Pattern Recognition. Academic Press, New York, 1974.
7. Hayes-Roth, F. and J. Burge "A novel pattern-recognition system applied to the learning of syllable, " Proc. Third Int. Joint Conf. Pattern Recognition, Coronado, Ca., 1976.
8. Smith, A.R. "Word Hypothesization in Hearsay II Speech System, " Proc. IEEE Int. Conf. ASSSP, Philadelphia, Pennsylvania., 1976.
9. Selfridge, D. "Pandemonium: a paradigm for learning" (1956), in L. Uhr(ed.), Pattern Recognition, 1966, pp339-448.
10. Thompson, R.A. "Language Correction using Probabilistic Grammars, " IEEE Trans. Computers, C-25, No. 3, March 1976.
11. Reddy, Raj, (Ed), Speech Recognition:Invited Papers of the IEEE Symposium. New York: Academic Press, 1975.