AN APPLICATION OF SYNTACTIC PATTERN RECOGNITION TO SEISMIC DISCRIMINATION

H. H. Liu and K. S. Fu

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

TR-EE 81-25
August 1981

This work was supported by the NSF Grant PRF 79-06296 and the ONR Contract N00014-79-C-0574

✓NSF-P [illegible]

AN APPLICATION OF SYNTACTIC PATTERN RECOGNITION TO SEISMIC DISCRIMINATION*

HSI-HO LIU and KING-SUN FU

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

Two syntactic methods for the recognition of seismic waveforms are presented in this paper. The seismic waveforms are represented by sentences (strings of primitives). Primitive extraction is based on a cluster analysis. Finite-state grammars are inferred from the training samples. The nearest-neighbor decision rule and error-correcting finite-state parsers are used for pattern classification. While both show equal recognition performance, the nearest-neighbor rule is much faster in computation speed. The classification of real earthquake / explosion data is presented as an application example.

I. INTRODUCTION

Seismological methods are the most effective and practical methods for detecting nuclear explosions, especially for underground explosions. Position, depth and origin time of the seismic events are useful information for discrimination; so are body wave magnitude and surface wave magnitude of the seismic wave [1,2]. Unfortunately, they are not always applicable and reliable for small events. It would be very helpful if the discrimination

* This work was supported by the NSF Grant PFR 79-06296 and the ONR Contract N00014-79-C-0574. ✓

is based on the short-period waves alone. The application of pattern recognition techniques to seismic wave analysis has been studied extensively [3-5] in the last few years. They all use short-period waves only. Most of these studies concentrated on feature selection. Only simple decision-theoretic approaches have been used. However, syntactic pattern recognition appears to be quite promising in this area. It uses the structural information of the seismic wave which is very important in analysis. In this paper, we present two different methods of syntactic approach to the recognition of seismic waves. One uses the nearest-neighbor decision rule, the other uses the error-correcting parsing.

In the first method, a pattern representation subsystem converts the seismic waveforms into strings of primitives. The string-to-string distances between the test sample and all the training samples are computed and then the nearest-neighbor decision rule is applied. The block diagram is shown in Figure 1(a). The second method contains pattern representation, automatic grammatical inference and error-correcting parsing. The block diagram is shown in Figure 1(b).

The pattern representation subsystem performs pattern segmentation, feature selection and primitive recognition so as to convert the seismic wave into a string of primitives. The automatic grammatical inference subsystem infers a finite-state (regular) grammar from a finite set of training samples. The error-correcting parser can accept erroneous and noisy patterns. Human interaction is required only in the training stage, mostly

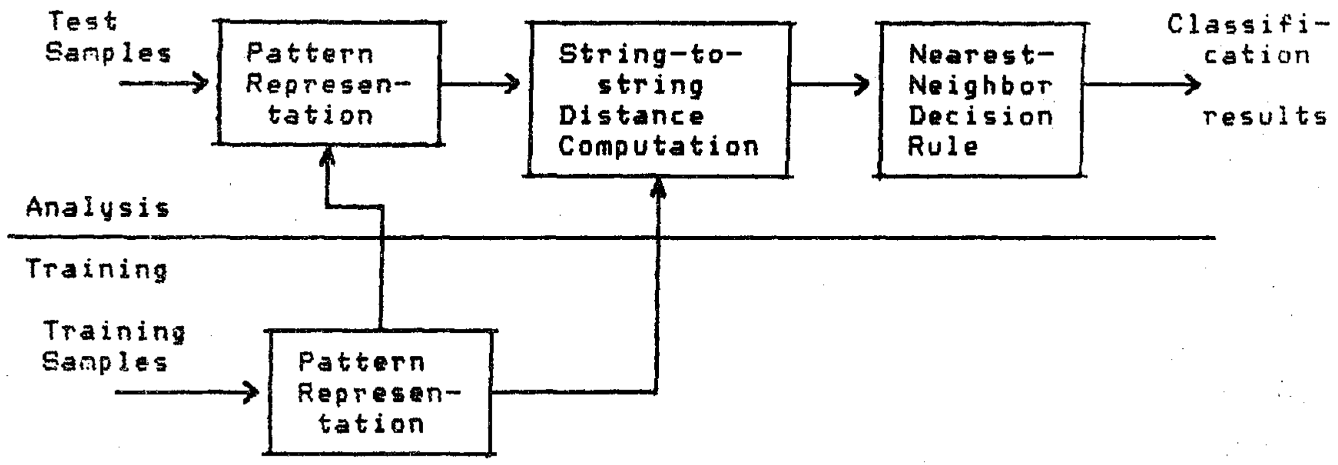

Figure 1(a) Block diagram of the nearest-neighbor decision rule for string patterns.

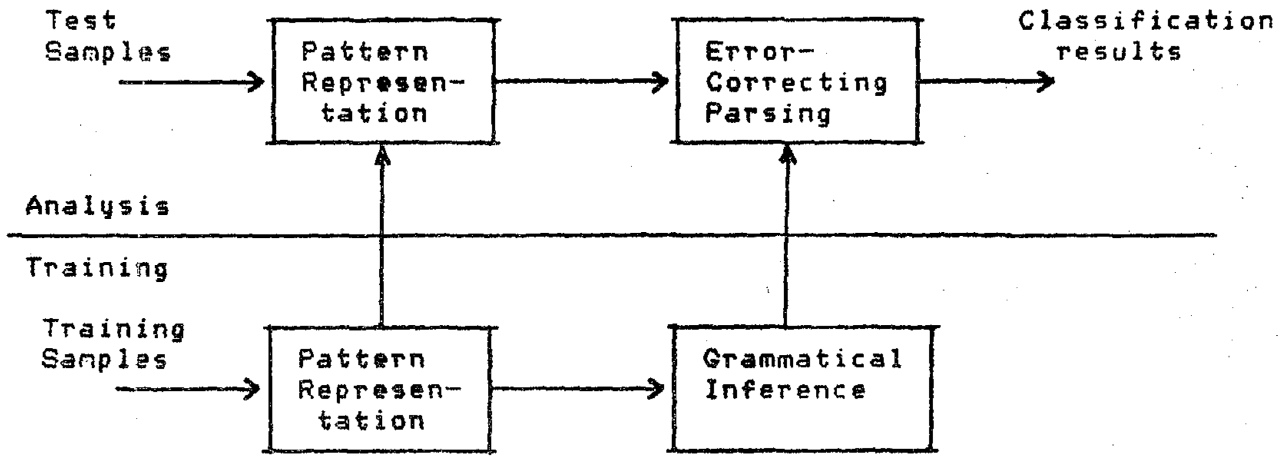

Figure 1(b) Block diagram of the error-correcting parsing system

in pattern representation and slightly in grammatical inference. We use our syntactic patten recognition methods to classify nuclear explosions and earthquakes based on the seismic P-waves. A typical sample from each class is shown in Figure 2(a).

However, extreme cases do exist. A near explosion looks like typical earthquakes while a deep earthquake looks like typical explosions. They are shown in Figure 2(b).

II. PATTERN REPRESENTATION

Seismic records are one-dimensional waveforms. Although there exist several alternatives [6,7] for representing one-dimensional waveforms, it is most natural to represent them by sentences, i.e., strings of primitives. Three steps are required for the conversion -- pattern segmentation, feature selection and primitive recognition.

A. Pattern Segmentation

A digitized waveform to be processed by a digital computer is usually sampled from a continuous waveform which represents the phenomena of a source plus external noise. For some cases, such as EKG wave [8], every single peak and valley are significant. Therefore these waveforms can be segmented according to the shape. For others, like EEG [9] and seismic wave, a single peak or velley does not reveal too much information, especially when the signal to noise ratio is low. Therefore, they should be segmented by length, either a fixed length or variable length. A variable-length segmentation is more efficient and precise in representation, but it is usually very difficult and time consuming to find an appropriate segmentation. A fixed-length segmentation is much easier to implement. If the length is kept short enough it will be adequate to represent the original waveform.

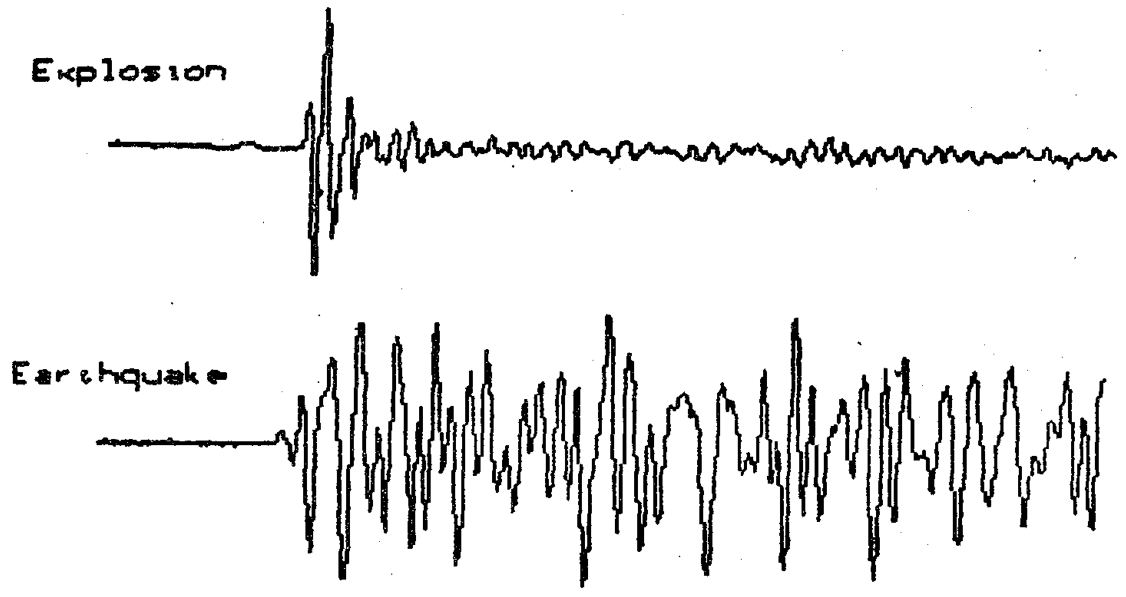

Figure 2(a) Typical samples from each class, explosion (top) and earthquake (bottom).

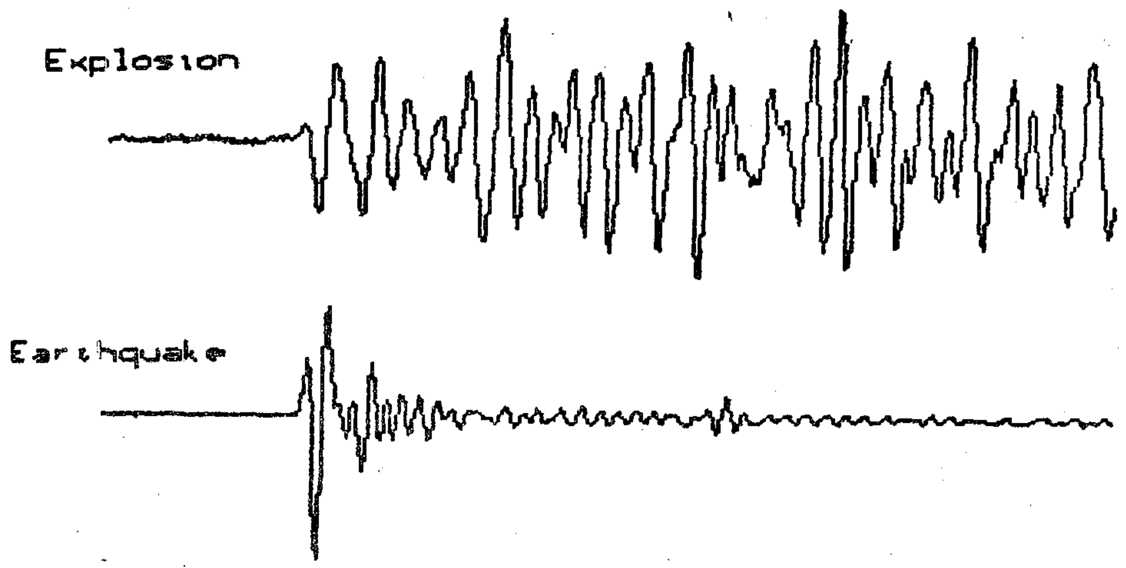

Figure 2(b) Extreme cases from each class, explosion (top) and earthquake (bottom).

The selection of segment length is case dependent. It can be anywhere between the two extremes, i.e., as long as the whole waveform or as short as one point. There are tradeoffs between the representation accuracy and analysis efficency. The shorter the segmentation is, the more accurate the representation will be. But the analysis becomes more inefficient since the string is longer and the parsing time is proportional to the string length. Another problem is the noise. If the segmentation is too short, it will be very sensitive to noise. A rule of thumb is that each segment should contain several periods of the waveforms. In our seismic data base each seismic record contains 1200 sample points. The sampling frequency is 10 points per second. Each record is divided into 20 segments with 60 points in each segment.

B. Feature Selection

This is the most difficult and critical part in pattern recognition. Any linear functions or nonlinear functions of the original measurements may be considered as features provided they give discriminating power. Both time domain features and frequency domain features have been used for seismic discrimination. For example, complexity and autoregressive models are features in time domain; spectral ratio and third moment of frequency are features in frequency domain [2]. Since we segment the seismic wave, complexity and spectral ratio features are implicitely contained in the string structure. Furthermore, the segment may be too short for a model estimation if we use shorter segment. Therefore, we selected a pair of commonly used features -- zero crossing count and log energy of each segment, which are easy to compute and contain significant information. Other features may also serve as good candidates. An advantage of syntactic approach is that feature selection is simpler, since features are extracted from segments and each segment is much simpler in comparison with the whole waveform.

C. Primitive Recognition

After segmentation and feature selection, primitives can be recognized from the analysis of training segments, and an identifier assigned to each segment. This problem can be solved in two ways -- either classified by human experts or by a computer. We choose the latter, since human classifications are not always available and reliable. In addition we need to try different segment lengths in order to find an optimal segmentation. Therefore, we use automatic clustering analysis to classify each segment. In the clustering process, similar samples will be grouped together. The similarity between a pair of samples is usually defined by the distance between them. Each segment is represented by a vector $X = (x_1, x_2, \ldots, x_k)$ where x_i, $1 \leq i \leq k$, is the i-th feature, k is the total number of features. In our case, $k = 2$.

If the number of clusters is known, then the K-means algorithm can be applied to find a clustering which minimizes a performance index. When the number of clusters is unknown there is no universally applicable algorithm to determine the optimal cluster number. We use a bottom-up hierarchical clustering algorithm [10] to find the clustering of a sequence of cluster numbers. The starting cluster number can be arbitrarily selected. It may equal to the number of the training segments, but it is too time consuming even for a moderately large training set. Therefore we start from a smaller number, say 20, to find the clustering using K-means algorithm. The nearest pair of clusters will be merged and the cluster number is decreased by one. The K-means algorithm is applied again for reorganization. This clustering-merging cycle repeats until the cluster number reaches a preset lower bound, say 5, then the procedure stops.

<u>Algorithm 1</u>: <u>Bottom-Up Hierarachical Clustering</u>

Input: A set of n unclassified samples, an upper bound U and a lower bound L.

Ouput: A sequence of optimal clusterings for the number of clusters between U and L.

Method:

1) Let $c = U$, c is the number of clusters, and arbitrarily assign cluster menbership.

2) Reassign membership using K-means algorithm. If $c \leq L$, stop.

3) Find the nearest pair of clusters, say X_i and X_j, $i \neq j$.

4) Merge X_i and X_j, delete X_j and decrease c by one, go to step 2.

The distance between two clusters is defined by

$$d(X_i, X_j) = \| m_i - m_j \|$$

where m_i, m_j is the mean vectors of clusters i, j respectively.

The main problem that still remains unsolved at this point is to determine the optimal cluster number. Some criteria have been suggested for determining the optimal cluster number. However, they are not always applicable. We determine the cluster number by inspecting the increment of merge distance. When a merge of two clusters is natural, the increment of merge distance should be small; otherwise it will be large. This can only be determined from a sequence of cluster numbers. The merge distances of our training samples from 18 clusters down to 7 clusters are shown in Table I. The increments of merge distances are considerably large after ten clusters. Therefore, it is reasonable to select ten to be the optimal number of clusters. After the cluster number had been determined, an identifier was assigned to each cluster. A test segment is assigned to some cluster if the distance between the test segment and that cluster is the smallest. All the seismic waves are thereby converted into strings of primitives, or sentences.

III. SYNTAX ANALYSIS

If the classification is all we need, then the nearest-neighbor decision rule is recommended because of it's computation efficiency. On the other hand, if a complete description of the waveform structure is needed, we have to use (error-correcting) parsing. An error-correcting parser instead of regular parser is

TABLE I
Merge distances of bottom-up
hierarchical clustering process

Cluster number	Merge distance	Increment of merge distance
18	18.7	——
17	29.9	11.2
16	36.6	6.7
15	37.7	1.1
14	43.7	6.0
13	47.6	3.9
12	57.2	9.6
11	67.4	10.2
10	94.5	27.1
9	105.4	10.9
8	144.9	39.5
7	187.1	42.2

required for most practical pattern recognition applications. Since noisy and errors in previous processings usually cause regular parsers to fail. It is not unusual that even a perfect pattern can not be parsed by a regular parser, especially when the grammar is inferred from a small set of samples. In that case, the error-correcting parsing is equivalent to finding the distance between a sentence and a language. The parse of the sentence may contain some error productions.

A. Nearest-Neighbor Decision Rule

The concept of nearest-neighbor decision rule in syntactic approach is the same as that in decision-theoretic approach. The only difference is in distance calculation. The distance between two strings is sometimes called Levenshtein distance [11], which is the minimum number of symbol insertions, deletions and substitutions required in order to transform one string into the other. If different weights are assigned to different symbols and/or operations, then the distance becomes a weighted Levenshtein distance. These distances can be computed using dynamic programming method [12]. Figure 5 shows the shortest path which transforms the string on the left into the string on the top.

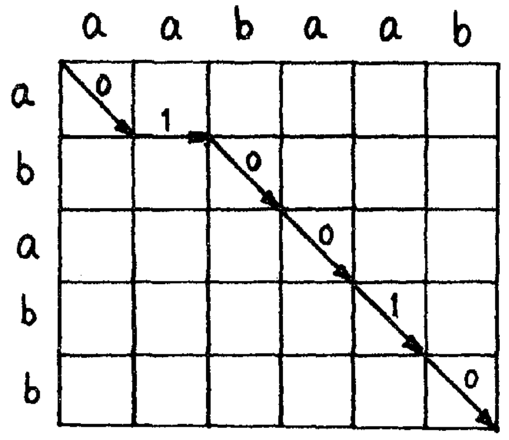

Figure 5 The shortest path which transforms string 'ababb' into string 'aabaab'. The distance between these two strings is 2. Horizontal movement means insertion; vertical movement means deletion; diagonal movement means substitution. Each insertion, deletion and substitution have same weight 1.

B. Error-Correcting Finite-State Parsing

Before parsing can take place we must have a grammar, which can be either heuristically constructed or inferred from a set of training samples. In order to study the learning capability of our syntactic method, we choose the grammatical inference approach.

Phrase Structure Grammar

A phrase structure grammar G is a 4-tuple
$G = (V_N, V_T, P, S)$, where

V_N: finite set of nonterminal symbols

V_T: finite set of terminal symbols, $V_N \cup V_T = V$, $V_N \cap V_T = \phi$.

S : start symbol, $S \in V_N$.

P : finite set of productions or rewrite rules of the form $\alpha \to \beta$, $\alpha, \beta \in V^*$, $\alpha \neq \lambda$; V^* is the set of all finite length strings of symbols from V, including λ, the null string, $V^+ = V^* - \{\lambda\}$.

Let $G = (V_N, V_T, P, S)$ be a grammar. If every production in P is of the form $A \to aB$, or $A \to a$, $A, B \in V_N$, $a \in V_T$, then the grammar G is finite-state or regular [13].

Phrase structure grammars have been used to describe patterns in syntactic pattern recognition [14]. Each pattern is represented by a string of primitives which corresponds to a sentence in a language (tree or graph in high dimensional grammars). All strings which belong to the same class are generated by one grammar.

Grammatical Inference

A set of sentences S^+ is a positive sample of a language L(C), if $S^+ \subseteq L(G)$. A set of sentences S^- is a negative sample of a language L(G), if $S^- \subseteq \bar{L}(G)$.

A positive sample S^+ of a language L(G) is structurally complete if each production in G is used in the generation of at least one string in S [15].

We assume that the set S is structurally complete and $S^+ \subseteq L(G_D)$, where G_D is the inferred grammar. Theoretically, if S^+ is a structurally complete sample of the language L(G) generated by the finite-state grammar G then the canonical grammar G_C can be inferred from S^+. A set of derived grammars can be derived from G_C. The derived grammars are obtained by partitioning the set of nonterminals of the canonical grammar into equivalence classes. Each nonterminal of the derived grammar corresponds to one block of the partition. Since the number of possible partitions is too large it is infeasible to evaluate all the partitions. Therefore some algorithms such as k-tail algorithm [16] has been suggested to reduce the number of derived grammars. These algorithms have one disadvantage. The reduced subset of derived grammars may not contain the source grammar. However, it will be sufficient if one only interests in an estimate of the source grammar. There are at least two situations where a grammatical inference algorithm can be used. In the first case there exists a source grammar which generates a language and we want to infer the source grammar or automaton based on the observed samples. In the second case the exact nature of the source grammar is unknown, the only information we have is some sentences generated by the source. We assume that the source grammar falls into a patricular class and infer a grammar which generates all the training samples, and hopefully will generate some samples belonging to the same class. If a negative sample set is given the inferred grammar must not generate any sample in the negative sample set.

Grammars more complex than finite-state grammars and restricted context-free grammars (in Chomsky hierarchy) can not be inferred efficiently without human interaction. Therefore we choose finite-state grammars to describe the seismic waves. Another reason is that no obvious self-embedding property appears in seismic waves, finite-state grammars will be sufficient in generating power.

The inference of regular grammars has been studied extensively. The k-tail algorithm finds the canonical grammar first and then merges the states which are k-tail equivalent. This algorithm is adjustable, the value of k controls the size of the inferred grammar. Another algorithm called tail-clustering algorithm [17] also finds the canonical grammar first, but then merges the states which have common tails. This algorithm is not as flexible as the k-tail algorithm, but will infer a grammar which is closer to the source grammar in some cases. Since the grammar is inferred from a small set of training samples, we can only expect that the inferred grammar generates all the training samples and will generate other strings which are similar to the training samples.

The generating power of the inferred grammar relies entirely on the merge procedure. If no merge exists then the inferred grammar will generate exactly the same training set, no more no less. Since all the seismic records have the same length in our example, the sentences representing these signals also have the same length. The merge of states does not happen in our experiment when using tail-clustering algorithm.

Error-Correcting Parsing

After a grammar is available, either by inference or construction, the next step is to design a recognizer which will recognize the patterns generated by the grammar. If the grammar G is finite-state, a deterministic finite-state automaton can be constructed to recognize the strings generated by G.

Noise problem and primitives recognition error usually occur in pratice. Conventional parsing algorithms can not handle these situations. A few approaches have been proposed. Error-correcting parsing is one of them [18]. The pattern grammar is first transformed into a covering grammar that generates the correct sentences as well as all the possible erroneous sentences. The errors in string patterns are substitution error, deletion error and insertion error. For nonstochastic grammar, the minimum-distance criterion can be used for error-correcting parsing.

Since all the sentences in our example have the same length, only the substitution error needs to be considered. For each production A -> aB and A -> a in the original grammar we add A -> bB and A -> b respectively to the covering grammar, where A, B $\in V_N$, a, b $\in V_T$, b $\neq$ a. Different weight can be assigned to each error production, therefore, resulting an minimum-cost error-correcting parser. The assignment of weights is very crucial. We use the distance between clusters a and b as the weight for substituting a by b and vise versa. Since a finite-state grammar can be represented by a transition diagram. Thus, a minimum-cost error-correcting parsing is equivalent to finding a minimum cost path from initial state to final state. The parsing time is proportional to the length of the sentence.

<u>Algorithm 2</u>: <u>Minimum-Cost paths</u>

Input. A transition diagram with n nodes numbered 1, 2, ..., n, where node 1 is initial state and node n is final state, and a cost function $c_{ij}(a)$, for $1 \leq i, j \leq n$, $a \in \Sigma$, with $c_{ij}(a) \geq 0$, for all i and j. An input string s.

Output. m_{1n} the lowest cost of any path from node 1 to node n whose sequence is equal to that of the input string s.

Method.

1) Set k = 1.

2) For all $1 \leq j \leq n$, $m_{1j} = \min\{m_{1k} + c_{kj}(b)$, for all $1 \leq k \leq n\}$, where b is the k-th symbol of input string s.

3) If k < |s|, increase k by 1 and go to step (2). If k = |s|, go to step (4).

4) output m_{1n}, which is the lowest cost from node 1 to node n following the move of input string s. Stop.

The production number can be stored with $c_{ij}(a)$, and the parse can be stored with m_{ij}.

If insertion and deletion errors are to be considered, then the parser will still be similar except that we have to compute and store the information V(T, S, a) which is the minimum cost of changing character 'a' into some string which can change the state of the automaton from state T to S [19]. The inclusion of insertion and deletion errors makes the error correction more complete, but assigning appropriate weights to insertion and deletion error could be even more difficult.

IV. EXPERIMENTAL RESULTS

Our seismic data are provided by professor C. H. Chen. They were recorded at LASA in Montana. The original data contains 323 records. Due to some technical problems in data conversion we only get 321 records. Among them 111 records are nuclear explosions and 210 records are earthquakes. The experiment was run on a VAX 11/780 computer using PASCAL programming language. A set of 50 carefully selected samples from each class is used as training samples. The remaining 210 samples are test samples. The weights for substitution errors are shown in Table II. The results shown in Table III and Table IV are the information about the inferred grammar and parsing. The grammars are inferred using K-tail algotithm with different values of k. Table III contains the number of nonterminals, the number of productions and the number of negative samples accepted. Table IV contains average parsing time for one string and the percentage of correct classification. It can be seen that as the value of k becomes smaller, the parsing time becomes shorter but the classification error becomes larger. This results from the uneven merge of the nonterminals. Due to the characteristics of our sample set only those states having the longest tails are merged. The results using nearest-neighbor decision rule are shown in Table V. It compares the string-to-string distance between the test sample and the whole class of training samples. The computation speed of nearest-neighbor rule is much faster than that of error-correcting parsers. Althouth the ultimate performance is about the same. As far as practical computation is concerned, nearest-neighbor decision rule is much faster than the grammatical approach.

TABLE II
Weights for substition error

	a	b	c	d	e	f	g	h	i	j
a	0	0.33	0.45	0.79	0.86	0.76	0.91	0.62	0.60	0.61
b	0.33	0	0.28	0.46	0.56	0.54	0.85	0.33	0.48	0.29
c	0.45	0.28	0	0.46	0.44	0.31	0.57	0.23	0.20	0.46
d	0.79	0.46	0.46	0	0.22	0.41	0.88	0.24	0.56	0.28
e	0.86	0.56	0.44	0.22	0	0.24	0.71	0.24	0.46	0.48
f	0.76	0.54	0.31	0.41	0.24	0	0.47	0.24	0.24	0.58
g	0.91	0.85	0.57	0.88	0.71	0.47	0	0.68	0.37	1.00
h	0.62	0.33	0.23	0.24	0.24	0.24	0.68	0	0.33	0.34
i	0.60	0.48	0.20	0.56	0.46	0.24	0.37	0.33	0	0.64
j	0.61	0.29	0.46	0.28	0.48	0.58	1.00	0.34	0.64	0

V. CONCLUDING REMARKS

TABLE III
The number of nonterminal, production and negative samples accepted of the inferred grammars. The inference algorithm is k-tail algorithm with different values of k.

k	Explosion		Earthquake		No. of negative samples accepted
	Nonterm. No.	Product. No.	Nonterm. No.	Product. No.	
20	748	796	746	794	0
19	748	796	746	794	0
18	741	796	737	794	0
17	722	778	715	772	0
16	694	751	686	743	0
15	656	714	650	708	0
14	610	668	608	666	0
13	560	618	561	619	0
12	510	568	511	569	0
11	460	518	461	519	0
9	360	418	361	419	0
7	262	319	261	319	2
5	166	222	164	220	6

Though the classification results seem satisfactory they are very sensitive to the feature selection, the selection of training samples and the weight assignment of error productions. Although a finite set of samples have some limitations. It still makes sense to pursue more studies about the following problems.

1. Feature selection. How to find a set of distinguishable features is the most important part in practical applications.

TABLE IV
The average parsing time and percentage of correct classification of the error-correcting parsers with different values of k.

k	Average parsing time for one string (sec)	Percentage of correct classification (%)
20	2.6	90.5
19	2.6	90.5
18	2.8	85.5
17	2.7	82.8
16	2.6	75.6
15	2.5	76.0
14	2.4	73.8
13	2.1	73.3
12	1.9	72.9
11	1.7	71.0
9	1.4	70.1
7	1.1	70.6
5	0.8	60.2

The difficulty increases when the class are somewhat overlapped. Possible solution are finding some kind of transformation which will seperate the classes or selecting the most distinguishable feature. Most of the features which are effective for statistical approach can be used for syntactic approach. The selection of feature number also deserves consideration. Some criteria are needed so that a judgement can be made.

2 Selection of training samples. It would be helpful if

TABLE V
Classification results using nearest-neighbor decision rule

Average time for one string (sec)	Percentage of correct classification
0.07	90.5 % 200 records are correctly classified out of 221

human experts are available for consultation. The clustering techniques can be used to get an initial training set, then it can be adjusted to obtain the best results. Clustering techniques can also be used to find good prototypes from a set of samples. A small set of well-selected training samples will certainly reduce computation time and, in the meantime, may improve the classification accuracy.

3. Weight assignment of error productions. This part is very important in error-correcting parsing, and only exists in syntactic approach. Equal weight assignment is very easy to implement and has been used. However, it is not always appropriate since costs should be different for different errors. The similarity between two primitives is a good reference for assigning weights to substitution errors. The weights of insertion and deletion errors are more difficult to assign. Only heuristic approaches have been known so far.

REFERENCES

[1] Bolt, B. A., Nuclear Explosions and Earthquakes, The Parted Veil. , W. H. Freeman and Co., San Francisco, 1976.

[2] Dahlman, O. and Israelson, H., Monitoring Underground Nuclear Explosions , Elsevier Scientific Publishing Co. , Amsterdam, the Netherlands, 1977.

[3] Chen, C. H., "Seismic pattern recognition," Geoexploration , vol. 16, No. 1/2, April 1978.

[4] Tjostheim, D., "Autoregressive representation of seismic P-wave signals with an application to the problem of short-period discriminants," Geophys. J. R. Astr. Soc. , vol. 43, pp. 269-291, 1975.

[5] Sarna C. S. and Stark H., "Pattern recognition of waveforms using modern spectral estimation techniques and its application to earthquake / explosion data," Proc. 5th Intl Conf on Pattern Recognition , Dec. 1-4, 1980.

[6] Pavlidis, T., "Linguistic analysis of waveforms," in Software Engineering, ed. by J. T. Tou, Vol. 2, New York : Acdemic, 1971, pp. 203-225.

[7] Ehrich, R. W. and Foith, J. P., "Representation of random waveforms by relational trees," IEEE Trans. Comput. , Vol. C-25, No. 7, pp. 725-736, July 1976.

[8] Horowitz, S. L., "A syntactic algorithm for peak detection in waveforms with applications to cardiography," Comm. ACM

, Vol. 18, No. 5, pp. 281-285, May 1975.

[9] Giese, D. A., Bourne, J. R. and Ward, J. W., "Syntactic analysis of the electroencephalogram," IEEE Tran. Syst., Man., Cybern., Vol. 9, No. 8, Aug. 1979, pp. 429-435.

[10] Duda, R. D. and Hart, P. E., Pattern Classification and Scene Analysis , New York: Wiley, 1973.

[11] Levenshtein, V. I., "Binary codes capable of correcting deletions, insertions, and reversals," Sov. Phys. Dokl. 10 (Feb. 1966), 707-710.

[12] Wagner, R. A. and Fischer, M. J., "The string-to-string correction problem," J. ACM 21, 1 (Jan 1974), 168-178.

[13] Aho, A. V. and Ullman, J. D., The Theory of Parsing, Translation and Compiling , Prentice-Hall, Inc., Vol. 1, 1972.

[14] Fu, K. S., Syntactic Methods in Pattern Recognition , Academic Press, New York, 1974.

[15] Fu, K. S. and Booth, T. L., "Grammatical inference- introduction and survey," IEEE Trans. Syst., Man, Cybern. , vol. SMC-5, Jan. and July, 1975.

[16] Biermann, A. W. and Feldman, J. A., "On the synthesis of finite-state machine," IEEE Trans. Comput., vol. C-21, June 1972.

[17] Miclet, L., "Regular inference with a tail-clustering

method," IEEE Trans. Syst., Man., Cybern., vol. SMC-10, Nov. 1980.

[18] Fu, K. S., "Error-correcting for syntactic pattern recognition," in Data Structure, Computer Gramphics and Pattern Recognition, Klinger et al., Eds. New York: Academic, 1976.

[19] Wagner, R. A., "Order-n correction of regular languages," Comm. ACM, vol. 17, No. 5, may 1974.