THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 57, No. 5, May–June 1978
Printed in U.S.A.

The Effects of Syntactic Analysis on Word Recognition Accuracy

By S. E. LEVINSON

(Manuscript received May 18, 1977)

In this paper we examine the effects of an algorithm for syntactic analysis on word recognition accuracy. The behavior of the algorithm is studied by means of a computer simulation. We describe the syntactic analysis technique, the problem domain to which it was applied, and the details of the simulation. We then present the results of the simulation and their implications. We find, for example, that an acoustic word error rate of 10 percent is reduced to 0.2 percent after syntactic analysis, resulting in a sentence error rate of 1 percent. These figures are based on a 127-word vocabulary and an average of 10.3 words per sentence for 1000 sentences. We expect that these results are indicative of the performance which will be attained by a real speech recognition system which uses the syntactic analysis algorithm described herein.

I. INTRODUCTION

The utility and flexibility of a speech recognition system can be substantially expanded if it can accept sentence length utterances rather than single words. Simultaneously, accuracy can be greatly improved by exploiting the grammatical constraints of language on the input sentences.[1,2,3]

The purpose of this investigation is to establish, by means of a computer simulation, how much improvement in reliability can be obtained by using a particular optimal method for syntactic analysis in conjunction with an isolated word recognition system.

This paper is in five sections. First we give a description of the method of syntax analysis under consideration. In the second section we describe the content and the semantic and grammatical structure of the problem domain to which we intend to apply the analysis. The third section is devoted to a description of the simulation, particularly of the acoustic recognizer and the procedure for generating random sentences. In the

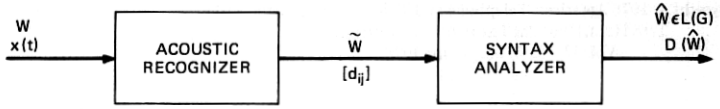


Fig. 1—Block diagram of the speech recognition system.

fourth section we present the results of the simulation. We conclude with an evaluation of the results and a brief discussion of directions for further investigations.

It should be said at the outset that the results of this experiment are encouraging. We find that, by using the grammatical constraints for our problem domain, the syntax analyzer reduces the acoustic error rate from 10 percent to 0.2 percent. This results in a sentence error rate of 1 percent. The sentences were composed from a 127-word vocabulary and contained an average of 10.3 words per sentence over 1000 randomly generated sentences.

We believe that these figures, with the more detailed results given in Section IV, are indicative of those which will be attained when the method of syntax analysis described herein is incorporated into a real speech recognition system.

II. AN OPTIMAL ALGORITHM FOR SYNTAX ANALYSIS

The type of speech recognition system we are evaluating is shown in Fig. 1, and its operation may be formally described as follows.

Let the language, L, be the subset of English used in a particular speech recognition task. Sentences in L are composed from the vocabulary, V, consisting of the M words $v_1, v_2, \ldots v_M$. Let W be an arbitrary sentence in the language. Then we write $W \in L$ and

$$W = w_1 w_2 \cdots w_k \tag{1}$$

where each w_i is a vocabulary word which we signify by writing $w_i \in V$ for $1 \leqslant i \leqslant k$. Clearly W contains k words, and we will often denote this by writing $|W| = k$. Similarly the number of sentences in L will be denoted by $|L|$.

The sentence W of eq. (1) is encoded in the speech signal $x(t)$ and input to the acoustic recognizer from which is obtained the probably corrupted string

$$\tilde{W} = \tilde{w}_1 \tilde{w}_2 \cdots \tilde{w}_k \tag{2}$$

where $\tilde{w}_i \in V$ for $1 \leqslant i \leqslant k$ but $\tilde{W}$ is not, in general, a sentence in L.

The acoustic recognizer also produces the matrix $[d_{ij}]$ whose ijth entry, d_{ij}, is the distance, as measured by some metric in an appropriate pattern space, from the ith word, $\tilde{w}_i$ to the prototype for the jth vocabulary word, v_j, for $1 \leqslant i \leqslant k$ and $1 \leqslant j \leqslant M$.

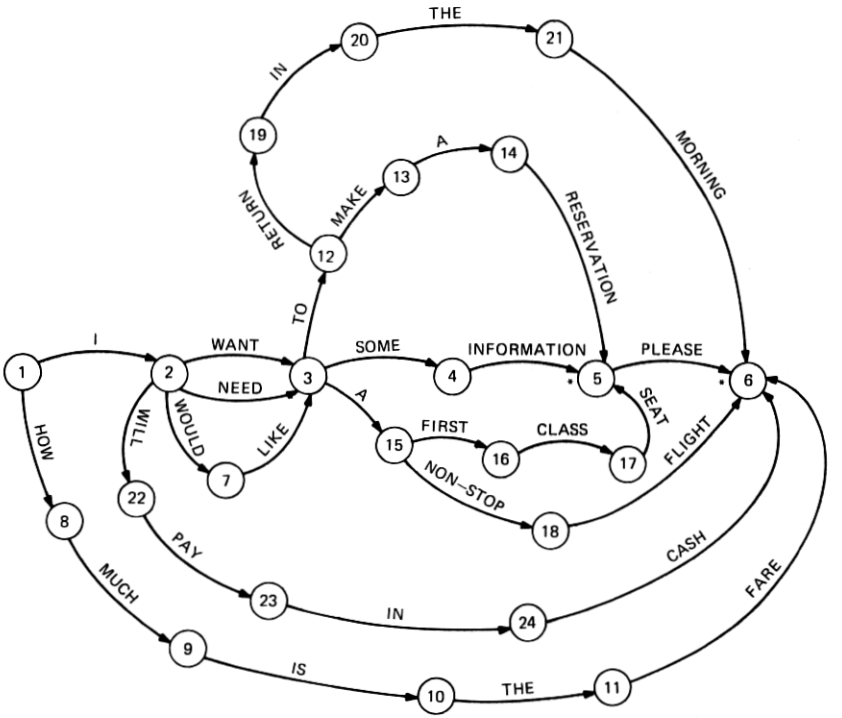


Fig. 2—Example state transition diagram.

The syntax analyzer then produces the string

$$\hat{W} = \hat{w}_1 \hat{w}_2 \cdots \hat{w}_k \tag{3}$$

for which the total distance, $D(\hat{W})$, given by

$$D(\hat{W}) = \sum_{i=1}^{k} d_{ij_i} \; 1 \leqslant j_i \leqslant M \tag{4}$$

is minimized subject to the constraint that $\hat{W} \in L$. Thus the syntax analysis is optimal in the sense of minimum distance.

Since, in general, $\hat{W} \notin L$, whereas W was assumed to be grammatically well-formed (i.e. $W \in L$), the process should correct word recognition errors.

In principle one could minimize the objective function of eq. (4) by computing $D(W)$ $\forall$ $W \in L$ and choosing the smallest value. In practice, when $|L|$ is large, this is impossible. One must perform the optimization efficiently. It has been shown by Lipton and Synder[4] that for a particular class of languages one can minimize $D(W)$ in time proportional to $|W|$. In fact one can optimize any reasonable objective function in time linear in the length of the input.

The particular class of languages for which the efficiency can be attained is called the class of Regular languages. For the purposes of this discussion we shall define the class of Regular languages as that class for which each member language can be represented by an abstract graph called a state transition diagram.

A state transition diagram consists of a finite set of vertices or states, Q, and a set of edges or transitions connecting the states. Each such edge is labeled with some $v_i \in V$. The exact manner of the interconnection of states is specified symbolically by a transition function, δ, where

$$\delta:(Q \times V) \rightarrow Q \tag{5}$$

That is, if a state $q_i \in Q$ is connected to another state $q_j \in Q$ by an edge labeled $v_m \in V$ then

$$\delta(q_i,v_m) = q_j \tag{6}$$

We also define a set of accepting states, $Z \subset Q$, which has the significance that a string $W = w_1 w_2 \cdots w_k$, where $w_i \in V$ for $1 \leqslant i \leqslant k$, is a well-formed sentence in the language, L, represented by the state transition diagram if and only if there is a path starting at q_1 and terminating in some $q_j \in Z$ whose edges are labeled, in order, $w_1, w_2, \cdots w_k$.

Alternatively we may write $W \in L$ iff

$$\begin{cases} \delta_1(q_1,w_1) = q_{j_1} \\ \delta_2(q_j,w_2) = q_{j_2} \\ \quad \cdot \\ \quad \cdot \\ \quad \cdot \\ \delta_k(q_{j_{k-1}},w_k) = q_{j_k} \in Z \end{cases} \tag{7}$$

We may then define the language, L, as the set of all W satisfying eq. (7). An example of these concepts is shown in Fig. 2. The accepting states are marked by asterisks.

While the definition given above of a Regular language is mathematically rigorous, it is not the standard one used in the literature on formal language theory but rather has been specifically tailored to the notational requirements of this paper. The interested reader is urged to refer to Hopcroft and Ullman[10] for a standard and complete introduction to formal language theory.

In the following discussion we shall restrict ourselves to finite Regular languages, i.e., those for which $|L|$ is finite. This restriction in no way alters the theory but its practical importance will become obvious in what follows. The finiteness of the language implies that its state transition diagram has no circuits, i.e., no paths of any length starting and ending at the same state. Thus there is some maximum sentence length which we shall denote, $l_{\max}$.

We now turn to the problem of efficiently solving the minimization problem of eq. (4). To do this we shall define two data structures Φ and Ψ which will be used to store the estimates of $D(\hat{W})$ and $\hat{W}$, respectively.

The first stage of the algorithm is the initialization procedure in which we set

$$\Phi_i(q) = \begin{cases} 0 & \text{for } q = q_0 \text{ and } i = 0 \\ \infty & \text{otherwise} \end{cases}$$

$$\Psi_i(q) = 0 \qquad 1 \leqslant i \leqslant |W| = k;\ \forall q \in Q \tag{8}$$

The data structures have two indices. The subscript is the position of the word in the sentence and the argument in parentheses refers to the state so that the storage required for each array is, at most, the product of $l_{\max} + 1$ and $|Q|$, the number of states in the set Q.

After initialization we utilize a dynamic programming technique defined by the following recursion relations:

$$\Phi_i(q) = \min_{\Delta} \{\Phi_{i-1}(q_p) + d_{ij}\} \tag{9}$$

where the set Δ is given by:

$$\Delta = \{\delta(q_p, v_j) = q\} \tag{10}$$

Then

$$\Psi_i(q) = \Psi_{i-1}(q_p)\hat{w}_i \tag{11}$$

where $\hat{w}_i$ is just the v_j which minimizes $\Phi_i(q)$. Equation (11) is understood to mean that the word $\hat{w}_i$ is simply appended to to the string $\Psi_{i-1}(q_p)$.

Unfortunately the concatenation operation is not easily implemented on general purpose computers so we change the recursion of eq. (11) by making Ψ into a linked list structure of the form:

$$\Psi_{1i}(q) = q_p$$

$$\Psi_{2i}(q) = \hat{w}_i \tag{12}$$

Then when $i = k$ we can trace back through the linked list of eq. (12) and construct the sentence $\hat{W}$ as follows: First find $q_f \in Z$ such that

$$\Phi_k(q_f) = \min_{q \in Z} \{\Phi_k(q)\} \tag{13}$$

set $q = q_f$ and then for $i = k, k-1, k-2, \ldots, 1$

$$\hat{w}_i = \Psi_{2i}(q)$$

$$q = \Psi_{1i}(q) \tag{14}$$

Table I — Example $[d_{ij}]$ matrix

Code	Vocabulary word	$I = 1$	$I = 2$	$I = 3$	$I = 4$	$I = 5$
1	Is	9	9	1	8	4
2	Fare	2	2	5	3	1
3	I	7	3	2	3	2
4	Want	2	9	4	7	3
5	Would	1	5	4	8	2
6	Like	2	5	2	6	5
7	Some	2	1	9	8	3
8	Information	7	7	4	7	8
9	Please	2	3	2	4	9
10	To	4	5	8	1	7
11	Make	6	3	9	8	5
12	A	4	7	6	9	8
13	Reservation	3	6	7	8	9
14	Return	9	7	6	4	8
15	The	8	6	5	2	3
16	Morning	3	4	5	6	7
17	First	8	6	8	7	5
18	Class	5	5	4	3	9
19	Seat	9	9	8	7	3
20	Non-stop	3	3	4	5	8
21	Flight	9	8	3	5	6
22	Will	6	7	7	6	5
23	Pay	4	4	4	4	3
24	In	3	3	3	6	9
25	Cash	5	4	3	7	6
26	How	2	9	8	7	5
27	Much	6	2	8	4	9
28	Need	7	6	5	4	3

Thus the sentence $\hat{W}$ is computed from right to left.

The operation of the above algorithm is illustrated in Tables I, II, and III. Table I shows the vocabulary words of the language diagrammed in Fig. 2 along with numerical codes and a sample $[d_{ij}]$ matrix. Table II shows the details of the operation of the algorithm for $i = 0,1,2$. Table III shows the results after the sentence has been completely analyzed.

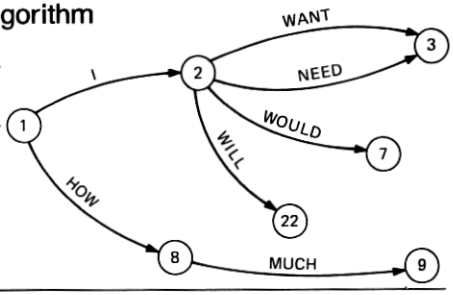
By locating the smallest entry in each column of the sample $[d_{ij}]$ matrix of Table I, it can be seen that the acoustic transcription of the sentence from which this matrix was produced is: WOULD SOME IS TO FARE. Clearly this is not a valid English sentence nor is there any path through the state transition diagram of Fig. 2 whose edges are so labeled.

Following Table II the reader can trace the operation of the algorithm as it computes the valid sentence having the smallest total distance. First the Φ and Ψ arrays are initialized according to eq. (8). To make the figure easier to read, this has been shown only for $i = 0$.

Note that there are two transitions from state 1; one to state 2 labeled I and the other to state 8 labeled HOW. Accordingly $\Phi_1(2)$ is set to 7, the metric for I; $\Psi_{11}(2)$ is set to 1, the state at the beginning of the transition and Ψ_{21} is set to 3, the code for the transition label I. Similarly $\Phi_1(8)$ is

Table II — Detailed operation of the first three stages of the algorithm

Code	Word	Position 1	Position 2
3	I	7	3
4	Want	2	9
28	Need	7	6
5	Would	1	5
22	Will	6	7
26	How	2	9
27	Much	6	2



$i\backslash q$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
ϕ_0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞		∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
ψ_{10}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_{20}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ϕ_1		7						2																
ψ_{11}		1						1																
ψ_{21}		3						26																
ϕ_2			13				12		4													14		
ψ_{12}			2				2		8													2		
ψ_{22}			28				5		27													22		

Table III — Final results of the algorithm

$i\backslash q$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	22	23	24
ϕ_0	0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
ψ_{10}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ψ_{20}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ϕ_1		7						2															
ψ_{11}		1						1															
ψ_{21}		3						26															
ϕ_2			13				12		4												14		
ψ_{12}			2				2		8												2		
ψ_{22}			28				5		27												22		
ϕ_3			14	22						5		21			19							18	
ψ_{13}			7	3						9		3			3							22	
ψ_{23}			6	7						1		10			12							23	
ϕ_4				22	29						7	15	29		23	26		24	25				24
ψ_{14}				3	4						10	3	12		3	15		15	12				23
ψ_{24}				7	8						15	10	11		12	17		20	14				24
ϕ_5					30	8							20	37		28	35	31	23	34			
ψ_{15}					4	11							12	13		15	16	15	12	19			
ψ_{25}					8	2							11	12		17	18	20	14	24			

set to 2, the metric for HOW; $\Psi_{11}(8)$ is set to 1 as before and $\Psi_{21}(8)$ is set to 26, the code for HOW. All other entries remain unchanged.

In the next stage more transitions become possible. Note in particular that there are two possible transitions from state 2 to state 3. In accordance with eq. (9), the one labeled NEED is chosen since it results in the smallest total distance, 13, which is entered in $\Phi_2(3)$; $\Psi_{12}(3)$ is set to 2, the previous state, and $\Psi_{22}(3)$ is set to 28, the code for NEED. Transitions to state 7, 9, and 22 are also permissible and thus these columns are filled in according to the same procedure.

The completion of this phase of the algorithm results in Φ and Ψ as shown in Table III. We can now trace back according to eqs. (13) and (14) to find $\hat{W}$. From Fig. 2 we see that there are two accepting states, 5 and 6. $\Phi_5(6)$ is 8 which is less than 30, the value of $\Phi_5(5)$, so we start tracing back from state 6. The optimal state sequence is, in reverse order, 6,11,10,9,8,1. The corresponding word codes which, when reversed, decode to the sentence: HOW MUCH IS THE FARE.

One final note: from the operation of the algorithm it should be clear that it is not necessary to retain $\Phi_i(q)$ for $0 \leqslant i \leqslant |W|$. At the ith stage one needs only $\Phi_{i-1}(q)$ to compute $\Phi_i(q)$. Thus the storage requirements are nearly halved in the actual implementation.

In closing we should note that this scheme is formally the same as (though conceptually different from) the Viterbi[5] algorithm and similar to methods used by Baker[6] and stochastic parsing techniques discussed in Fu[7] and Paz.[8] The crucial difference is that in the cited references, estimates of transitions probabilities are used whereas in this method the transitions are deterministic and the probabilities used are only those conditioned on the input $x(t)$.

III. THE SEMANTIC AND GRAMMATICAL STRUCTURE OF THE RECOGNITION TASK

In this section we shall discuss the application of the abstractions of the previous section to a particular speech recognition problem domain. The task which was finally selected was that of an airline information and reservation system. The choice was made for three reasons. First, the problem is difficult enough so that even under some artificial constraints, it is a significant test of the above described techniques. Second, previous work by Rosenberg and Itakura[9] which used single words rather than sentences composed of isolated words as input was available for purposes of comparison. Third, it affords the opportunity to add modes of human/machine communication such as speaker verification and voice response.

The semantics of the language we designed limits a user to the following types of messages. First, one may state the desired kind of transaction (i.e., requesting flight information or making a reservation). Then, one may make a reservation either by providing all necessary information in one sentence or by giving answers to questions as required. The user may select arrival and/or departure dates, times and cities, number of stops, number and class of seats, specific flight numbers and aircraft types. Alternatively, the user may ask questions about arrival and/or departure dates, times and locations of specific flights, the type of aircraft, the number of stops, the fare, the number of meals served and the flight time. Finally, he may request a repeat of any information or supply telephone numbers and method of payment.

For most of the above messages there are several acceptable grammatical structures of sentences conveying the same semantic information.

In order to limit the complexity of the syntactic analysis, certain arbitrary constraints were imposed:

(*i*) Dates consist of two digits followed by the name of the month.
(*ii*) Flight numbers are limited to one or two digits.
(*iii*) The vocabulary includes the names of only ten cities.
(*iv*) The length of the longest sentences is 22 words.

It was felt that these constraints could all be relaxed if so desired without making major system modifications.

Next we give an informal specification of the syntactic structure. In this description phrases enclosed in curly brackets are alternatives. Those enclosed in square brackets are optional, while those within angle brackets represent a class of words of the indicated type.

$$\text{I} \left\{ \begin{matrix} \text{WANT} \\ \text{WOULD LIKE} \end{matrix} \right\} \left\{ \begin{matrix} \text{SOME INFORMATION} \\ \text{TO MAKE A RESERVATION} \end{matrix} \right\} [\text{please}]$$

$$\text{I} \left\{ \begin{matrix} \text{WANT} \\ \text{WOULD LIKE} \end{matrix} \right\} \text{TO} \left\{ \begin{matrix} \text{GO} \\ \text{LEAVE} \\ \text{RETURN} \\ \text{DEPART} \end{matrix} \right\} [\text{FROM } \langle \text{city} \rangle][\text{TO } \langle \text{city} \rangle]$$

$$[\text{ON}][\langle \text{day} \rangle] \left[\left\{ \begin{matrix} \text{MORNING} \\ \text{AFTERNOON} \\ \text{EVENING} \\ \text{NIGHT} \end{matrix} \right\} \right] [\text{THE } \langle \text{date} \rangle]$$

$$\left\{ \begin{matrix} \text{AT WHAT TIME} \\ \text{WHEN} \end{matrix} \right\} \text{DO FLIGHTS LEAVE } [\langle \text{city} \rangle] \text{ FOR } \langle \text{city} \rangle$$

$$\text{HOW MANY FLIGHTS} \left\{ \begin{matrix} \text{ARE THERE} \\ \text{GO} \end{matrix} \right\} [\text{FROM } \langle \text{city} \rangle]$$

$$\text{TO } \langle \text{city} \rangle \text{ ON [day]} \left[\left\{ \begin{matrix} \text{MORNING} \\ \text{AFTERNOON} \\ \text{EVENING} \\ \text{NIGHT} \end{matrix} \right\} \right] [\text{THE } \langle \text{date} \rangle]$$

$$\text{WHAT PLANE IS ON FLIGHT } \langle \text{flightnumber} \rangle \text{ [TO } \langle \text{city} \rangle]$$

$$\left\{ \begin{matrix} \text{WHAT} \\ \text{HOW MUCH} \end{matrix} \right\} \text{IS THE FARE [FROM } \langle \text{city} \rangle] \text{ [TO } \langle \text{city} \rangle]$$

$$\text{IS A MEAL SERVED ON [THE] FLIGHT } [\langle \text{flightnumber} \rangle][\text{TO } \langle \text{city} \rangle]$$

I { WANT / WOULD LIKE / WILL TAKE } FLIGHT [NUMBER] ⟨flightnumber⟩

[TO ⟨city⟩] [ON ⟨day⟩ { MORNING / AFTERNOON / NIGHT / EVENING }] [THE ⟨date⟩]

I { WANT / NEED / WOULD LIKE } ⟨digit⟩ [{ FIRST CLASS / COACH }] SEAT [s].

I PREFER THE [⟨manufacturer⟩] ⟨aircraft type⟩.

I { WANT / WOULD LIKE } TO GO AT ⟨hour⟩ { a.m. / p.m. / O'CLOCK }

PLEASE REPEAT THE { ⟨flightnumber⟩ / FARE / ARRIVAL TIME [s] / DEPARTURE / PLANE / NUMBER OF MEALS / FLIGHTS }

{ AT WHAT TIME / WHEN } DOES FLIGHT ⟨flightnumber⟩

[{ FROM / TO } ⟨city⟩] { ARRIVE / DEPART }

I WILL PAY BY { CASH / AMERICAN EXPRESS / DINERS CLUB / MASTER CHARGE }

MY { HOME / OFFICE } PHONE [NUMBER] IS [AREA CODE]
[⟨area code number⟩] ⟨phone number⟩

I { WANT / WOULD LIKE } A NON-STOP FLIGHT

HOW MANY STOPS ARE THERE ON FLIGHT ⟨flight number⟩

$$\left[\left\{\begin{matrix}\text{FROM}\\ \text{TO}\end{matrix}\right\}\ \langle\text{city}\rangle\right]\left[\text{ON}\ \langle\text{day}\rangle\ \left\{\begin{matrix}\text{MORNING}\\ \text{AFTERNOON}\\ \text{EVENING}\\ \text{NIGHT}\end{matrix}\right\}\right]\ [\text{THE}\ \langle\text{DATE}\rangle]$$

WHAT IS THE FLIGHT TIME [FROM ⟨city⟩] [TO ⟨city⟩]

The above description is too informal to define every detail of the Flight Information Language. It should, however, give the reader a feeling for the basic syntax and semantics. This specification of the language is quite useless for the purpose of the syntax analysis algorithm. For that purpose we have produced a formal specification of the language, the state transition diagram for which is shown in Fig. 3. From this graph it may be seen that $|V| = 127$; $|\delta| = 450$; $|Q| = 144$ and $|Z| = 21$ with the accepting states being designated by asterisk.

IV. DETAILS OF THE SIMULATION

The simulation of the speech recognition system based upon the analysis described above may be treated as three separate problems. They are: (*i*) Random generation of many well-formed sentences, (*ii*) computing a $[d_{ij}]$ matrix for each in such a way that the number of acoustic errors resulting from a nearest-neighbor decision rule is controllable, and (*iii*) syntactically analyzing the sentences and tabulating the appropriate statistics automatically. We shall now discuss these in order.

The method for generating random sentences in the language is just the following algorithm:

(*i*) $W \leftarrow O; q_i \leftarrow q_1$ (W gets the null string) at the ith stage,

(*ii*) Chose a word, $w_{k_i} \in V$

(*iii*) If $\delta(q_i, w_{k_i}) \neq q_j$ for some j go to (2) else $W \leftarrow W\, w_{k_i}$ (W gets itself concatenated with W_{k_i}) $q_i \leftarrow q_j$

(*iv*) If $q \in Z$ and $\rho < \theta$ where ρ is a pseudorandom number and θ is some threshold, STOP; else go to (2)

When the procedure terminates, $W = w_{k_1} w_{k_2} \ldots w_{k_l}$; $l \leqslant l\max$. Obviously by changing the threshold, θ, one can vary the average length of the sentences produced. In the actual simulation, ρ was uniformly distributed on $(\frac{1}{2}, -\frac{1}{2})$ and θ was set to -0.25 producing an average sentence length of 10.3 words.

In all, 42,000 such sentences were generated. In addition we made up

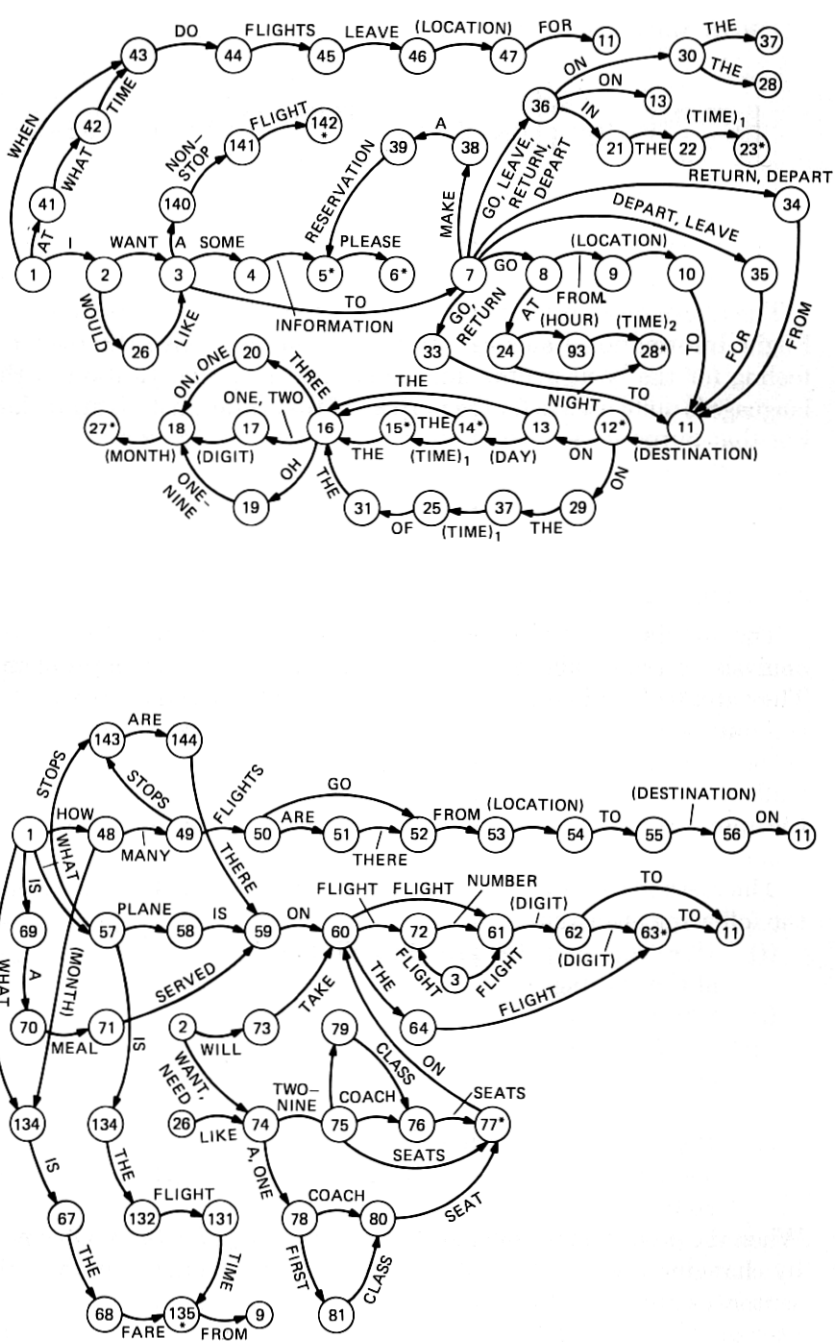
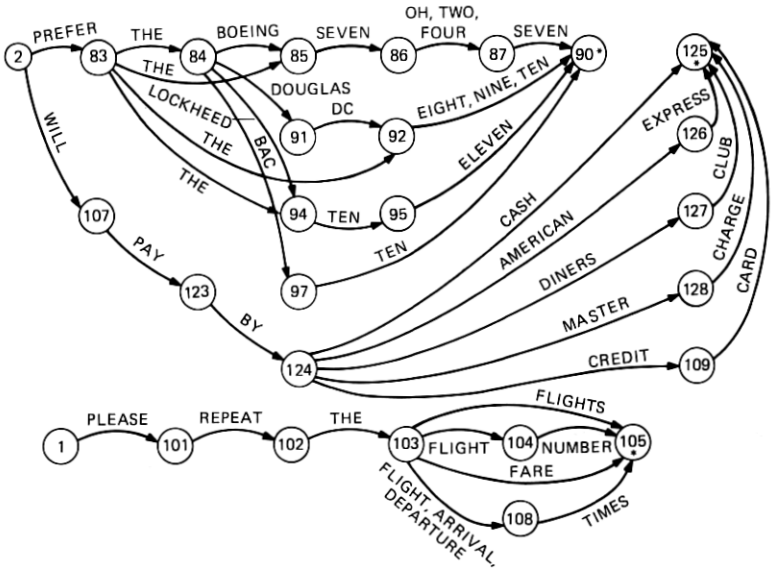
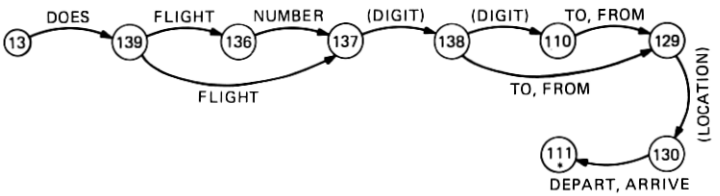


Fig. 3—State transition diagram of the Flight Information and Reservation Language.





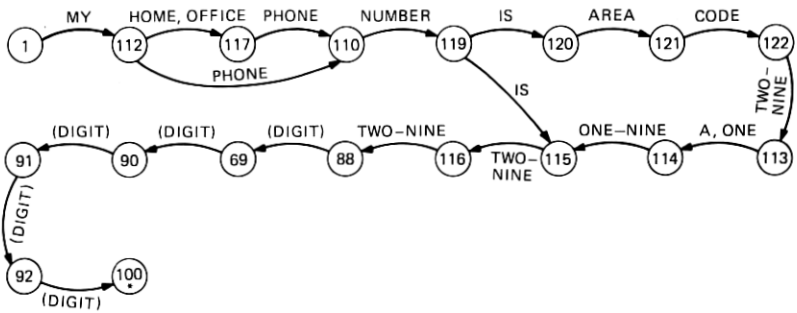


Fig. 3 (*continued*)

an additional 171 sentences averaging 9.7 words in length to use as a check against the randomly generated sentences. The additional sentences comprised several realistic transactions between airline customers and an automated flight information and reservation system.

The procedure for generating a $[d_{ij}]$ matrix simulating one which might be produced by acoustic recognition for the randomly selected sentence $W = w_1 w_2 \cdots w_k$ is as follows. Corresponding to each $v_j \in V$ we assign the five-dimensional Gaussian density function

$$p_j(\vec{x}) = (2\pi)^{-5/2}|U|^{-1/2} e^{-1/2(\vec{x}-\vec{m}_j)U^{-1}(\vec{x}-\vec{m}_j)^T} \tag{15}$$

where the covariance matrix, U, was chosen, for convenience, to be

$$U = \begin{bmatrix} \sigma^2 & & & & \\ & \sigma^2 & & 0 & \\ & 0 & \sigma^2 & & \\ & & & \sigma^2 & \\ & & & & \sigma^2 \end{bmatrix} \tag{16}$$

for selected values of σ^2. The mean vectors

$$\vec{m}_j = (m_{1j}, m_{2j}, m_{3j}, m_{4j}, m_{5j})$$

were fixed by selecting the m_{ij} at random from

$$m_{ij} = \begin{cases} 2 \\ 1 \\ 0 \end{cases} \quad \text{for } 1 \leq l \leq 5;\ 1 \leq j \leq |V| \tag{17}$$

until the 127 mean vectors were defined. Then for $w_i = v_j$, a random vector $\vec{y}$ was drawn from $p_j(\vec{x})$ and distances were computed according to:

$$d_{ij} = ||\vec{y} - \vec{m}_j|| \text{ for } 1 \leq j \leq |V| \tag{18}$$

where the norm is the simple Euclidean distance. Equation (18) was evaluated for $1 \leq i \leq k$ thus all entries in the $[d_{ij}]$ matrix were computed for each sentence W.

The acoustic recognition was simulated by a nearest neighbor rule so that $\tilde{w}_i = v_j$ if

$$d_{ij} \leq d_{in} \text{ for } 1 \leq n \leq |V| \tag{19}$$

Again, eq. (19) was applied for $1 \leq i \leq k$ and ties were arbitrarily broken. Clearly by changing the value of σ^2 in eqs. (15) and (16) the simulated acoustic error rate can be varied with small values of σ^2 producing low error rates.

Given the foregoing discussion, description of the simulation is quite simple. A set of 1000 random sentences was generated and its distance matrices computed. The sentences were syntactically analyzed and errors counted. This was done for $0.05 \leq \sigma \leq 2.1$ with σ being incremented by 0.05 for each set of 1000 sentences.

In addition, the specially formulated 171 sentences were typed in and processed. In this case σ was fixed at a value of 0.245 which resulted in

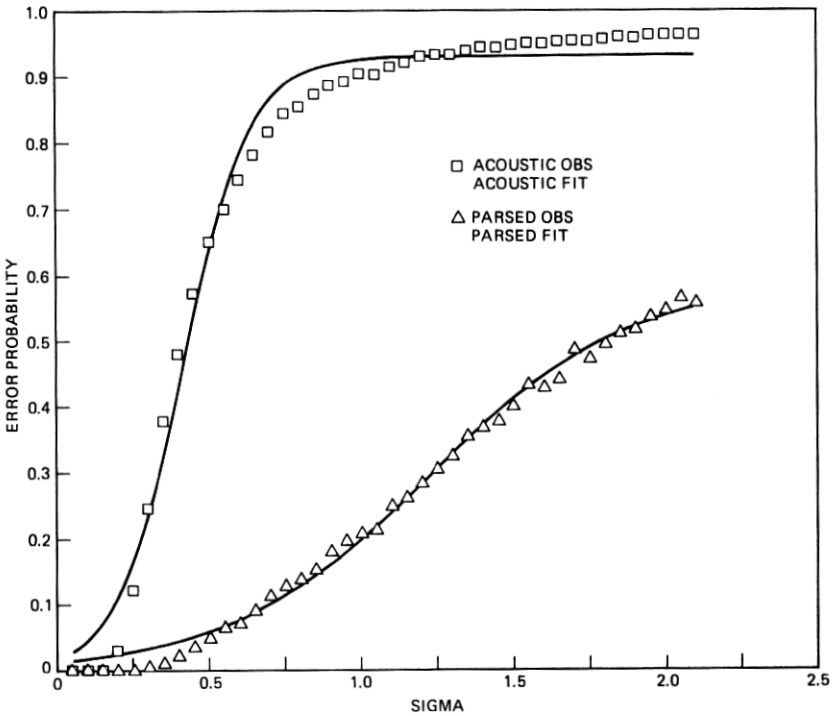


Fig. 4—Error rates as a function of σ.

an acoustic error rate of 11 percent which is close to the value observed by Rosenberg and Itakura[9] for a similar word recognition task. Error rates were measured for these sentences as well.

V. SIMULATION RESULTS

The overall results of the simulation are encouraging, showing considerable improvement in word and sentence recognition accuracy. Given an acoustic error rate of 10 percent on 1000 randomly generated sentences averaging 10.3 words per sentence, syntactic analysis reduces the word error rate to 0.2 percent resulting in a sentence error rate of 1 percent. A sentence is in error if even one word is improperly classified. For the test set of 171 sentences containing 1662 words, an 11 percent acoustic word error rate was lowered to 0.2 percent after syntactic analysis resulting in a 1.2 percent sentence error rate. The actual time required to analyze a 22-word sentence on the Data General Nova 840 is a small fraction of a second.

Details of the results are best given in the accompanying figures. Figure 4 is a plot of acoustic and syntactic word error probabilities as a

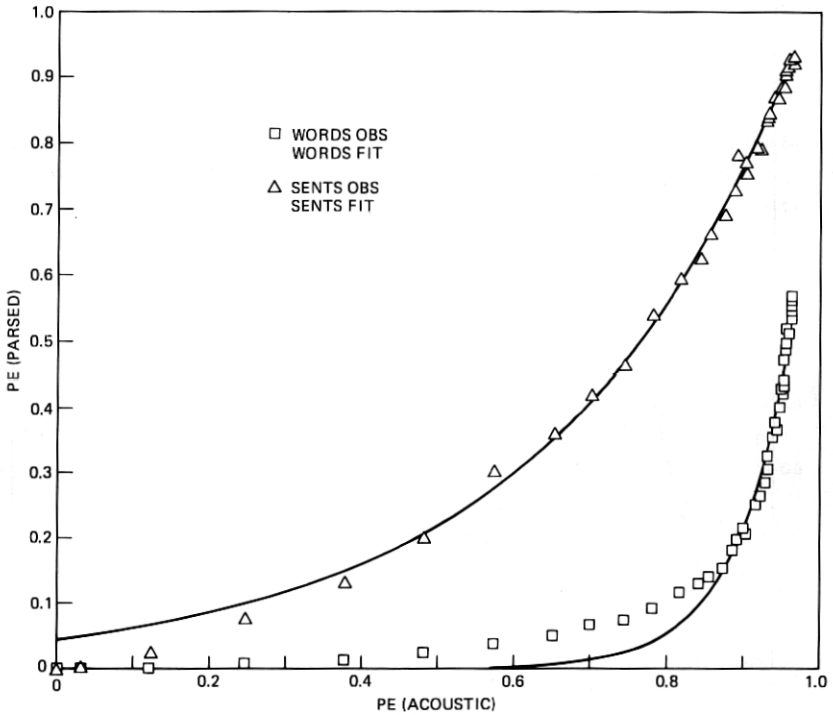


Fig. 5—Word and sentence error rates as a function of acoustic error rate.

function of σ of eqs. (15) and (16). Each point marked with a symbol is an observed data point based on the results obtained from 1000 sentences. It should be noted that the results for each point are based upon different sentences. The solid lines are obtained from a nonlinear least-squares fit of the data to the equation

$$P_e = \frac{\alpha_1\alpha_2}{\alpha_3\alpha_2 + (\alpha_1 - \alpha_3\alpha_2)e^{-\alpha_1\sigma}}. \tag{20}$$

The method used in fitting the data is described by McCalla.[11] The curve of eq. (20) is called a logistic curve; its significance is discussed in detail by Braun.[12] For the simulated data the standard deviation of the actual data from the fitted curve was <0.005.

Figure 5 shows the word and sentence error probabilities as a function of the acoustic word error probability. Once again the marked points are derived from sets of 1000 randomly generated sentences while the solid curves are obtained by fitting the data to an exponential curve of the form

$$P_{ep} = \alpha_1 e^{\alpha_2 P_e} \tag{21}$$

Once again the resulting fits were good having a standard deviation of <0.004.

VI. CONCLUSIONS

It is clear from the results of the simulation that the syntax analysis algorithm is very fast and effective in eliminating word recognition errors which occur at the acoustic level. It is expected that the performance of the algorithm for real speech input will depend on the characteristics of the acoustic recognizer and the task language. However, we believe that our results are indicative of the performance which can be attained in real speech recognition systems.

There are several areas for further research which are immediately suggested by this work. Although this entire presentation has been oriented toward the grammatical structure of sentences, the method described is certainly not restricted to that area. For example, the phonemic structure of words can be specified by a formal language as can the composition of acoustic features into phenomes. We therefore feel that optimal syntax analysis methods will be useful in more difficult speech recognition tasks than the one described here.

Another useful extension of the technique would be achieved by retaining the same optimality criterion while relaxing the restriction that $|\tilde{W}| = |\hat{W}|$. In other words, the algorithm would be allowed to insert and delete words. This could be an important aid in the solution of the segmentation problem in continuous speech.

On the theoretical side, it would be enlightening to derive analytical expressions for the average probability of error for the syntax analyzer, given the properties of the language and a characterization of the acoustic recognizer. Perhaps for this purpose the entropy or redundancy of the language might be sufficient, while the acoustic recognizer might be viewed as a noisy channel and characterized by its equivocation or capacity. In any event, it seems obvious that an information theoretic analysis would provide insights into the behavior of speech recognition systems.

Finally we note that Regular languages are the most simple syntactic structures. One naturally wonders whether efficient, optimal methods exist for formal languages of much greater complexity which would be better models of Natural Language.

In summary, we may say that optimal syntactic analysis techniques are useful and powerful tools to be used in tractable speech recognition tasks as well as being interesting mathematical objects.

REFERENCES

1. S. E. Levinson, "An Artificial Intelligence Approach to Automatic Speech Recognition," Proc. IEEE Conf. on Systems, Man and Cybernetics, Boston, November, 1973.

2. S. E. Levinson, "The VOCAL Speech Understanding System," Proc. 4th IJCAI, Tbilisi, U.S.S.R., September 1975.
3. S. E. Levinson, "Cybernetics and Automatic Speech Understanding," Proc. IEEE ICISS, Patras, Greece, August 1976.
4. R. J. Lipton and L. Snyder, "On the Optimal Parsing of Speech," Yale Univ. Dept. of Comp. Sci. Res. Report No. 37, New Haven, Conn., October 1974.
5. A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimal Algorithm," IEEE Trans. on Inf. Theor., *IT-13*, March 1967.
6. J. K. Baker, *The DRAGON System: An Overview," IEEE Trans. on Acoust., Speech, and Signal Processing, ASSP-23*, February 1975.
7. K. S. Fu, *Syntactic Pattern Recognition*, New York: Academic Press, 1974.
8. A. Paz, *Introduction to Probabilastic Automata*, New York: Academic Press, 1971.
9. A. E. Rosenberg and F. Itakura, "Evaluation of an Automatic Word Recognition System over Dialed-up Telephone Lines," J.A.S.A., *60*, Supp. 1, Fall 1976.
10. J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relations to Automata*, Reading, Mass.: Addison-Wesley, 1969.
11. T. R. McCalla, *Introduction to Numerical Methods and FORTRAN Programming*, New York: Wiley, 1967.
12. M. Braun, *Differential Equations and their Applications as an Introduction to Applied Mathematics*, New York: Springer Verlag, 1975.