AD-A049 680 LOGICON INC SAN DIEGO CALIF F/G 5/8
USE OF COMPUTER SPEECH UNDERSTANDING IN TRAINING: A PRELIMINARY--ETC(U)
JUN 77 J E PORTER, M W GRADY, M B HICKLIN N61339-74-C-0048
NAVTRAEQUIPC-74-C-0048-2 NL

UNCLASSIFIED

2 OF 2
AD
A049 680

END
DATE
FILMED
3-78
DDC

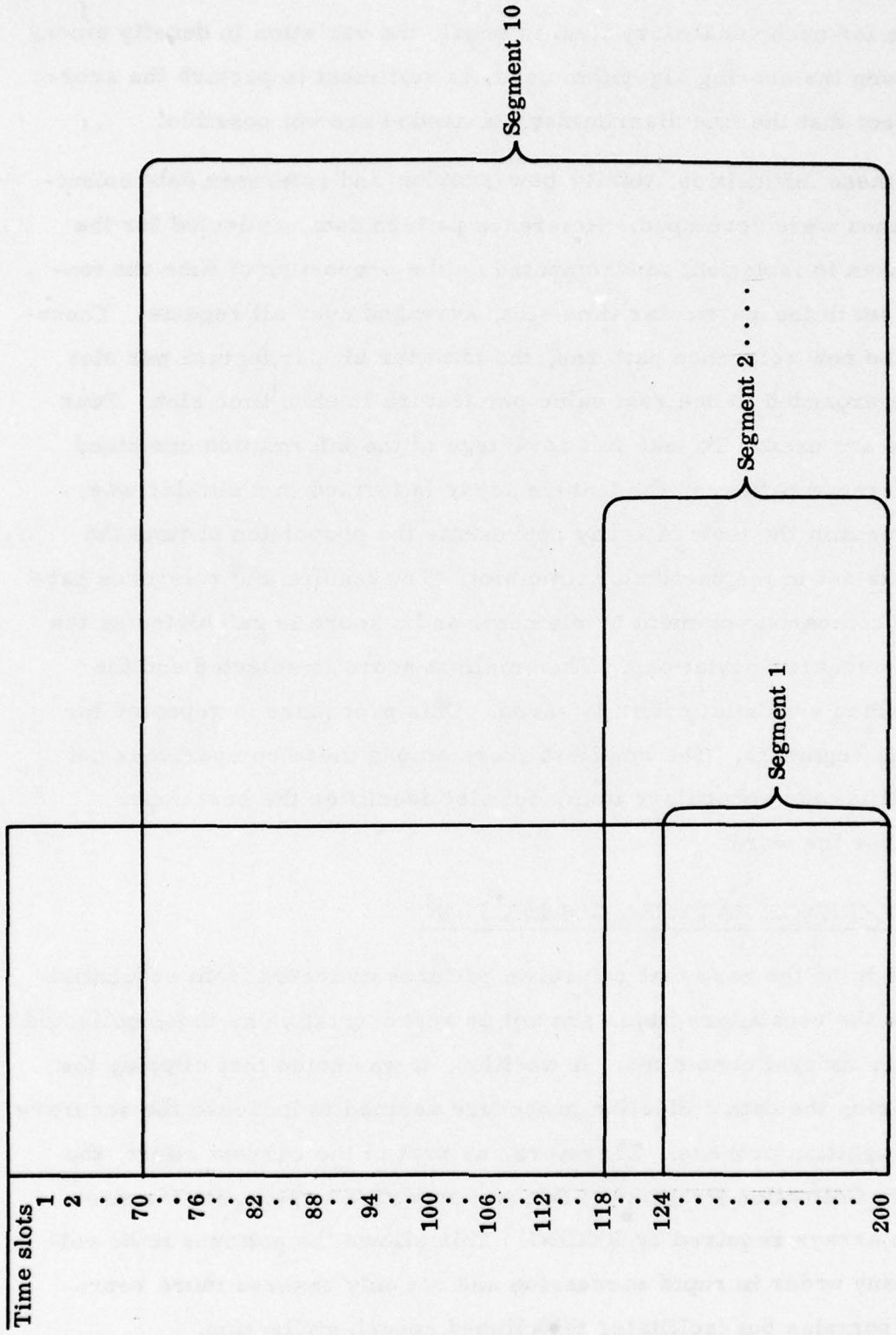


Figure 14. Segmentation of the Raw Data.

of repeats for each vocabulary item is equal, the variation in density among items, given the scoring algorithm used, is sufficient to perturb the scores to the extent that the fine discriminations needed are not possible.

To solve these difficulties, totally new scoring and reference data collection schemes were developed. Reference pattern data, collected for the digits spoken in isolation, are computed as the proportion of time the feature was set in the particular time slot, averaged over all repeats. Therefore, in the new reference patterns, the familiar bit per feature per slot pattern is expanded to one real value per feature in each time slot. Four time slots are used. To take full advantage of the information contained in the reference patterns, the feature array is formed in a similar way. Each element in the feature array represents the proportion of time the feature was set in the particular time slot. The feature and reference patterns are compared, element by element, and a score is calculated as the sum of the squared deviations. The smallest score is selected and the corresponding vocabulary item is saved. This procedure is repeated for each of the segments. The smallest score among these comparisons not only identifies the vocabulary item, but also identifies the best upper boundary for the word.

B.4 REFERENCE PATTERN COLLECTION

It appears to be the case that reference patterns collected from sequential repeats of the vocabulary items are not as representative as those collected under more natural conditions. In addition, it was noted that clipping the speech during the data collection procedure seemed to increase the accuracy of the recognition process. Therefore, as part of the current effort, the Voice Data Collection Program (VDC) was modified to generate the new reference arrays required by RECOG. This allows the patterns to be collected in any order in rapid succession and not only ensures more representative samples but facilitates the clipped speech stylization.

B. 5 RESULTS

Recognition accuracy data were collected and summarized for every talker. Data for one talker are shown in table 5. The processing time required to obtain the results shown for one talker using the Nova 1200 was about 2-1/2 hours. Table 6 summarizes the data obtained for all four talkers.

B. 6 CONCLUSIONS

This technique requires so much processing time that one might at first be tempted to omit it from consideration for a real-time application. The scheme was tested on an Eclipse and processing time was found to be reduced by 58 percent. Therefore, it is conceivable that, with optimized code available with FORTRAN V, running in a fast processor with floating point hardware, and perhaps some microcoded routines, that near real-time performance could be achieved for strings of three digits. However, we do not believe that the accuracies achieved with the technique warrants this sort of implementation yet. Further development is needed.

Many avenues of investigation are possible. For example, the initial segmentation might be improved based on a statistical study of the relative length of words in different positions in the utterance. In addition, the many constants including number of time slots, number of trial boundaries, etc. were chosen without the benefit of the standard data base reference and could probably be improved. Furthermore, if the leading digit were restricted to the 0 – 3 range, a better scheme for detecting the middle digit could be developed. The data base collected for this project provides the necessary reference against which further modifications can be tested.

In general, this approach is not without promise, but any further development should definitely have to await the results of the more sophisticated mathematical machine approach.

TABLE 5. DATA SUMMARY.

NUMBER OF VOCABULARY ITEMS: 11 NUMBER OF TIME SLOTS 4
STEP SIZE: 4 NUMBER OF STEPS AT BOUNDARIES: 10
DATA FILE NAME: MNSETB

	NUMBER OF DIGITS IN STRING									
	1	2	3	4	5	6	7	8	9	10
NUMBER OF STRINGS SPOKEN	11	0	12	32	0	0	0	0	0	0
% CORRECT	100.0	0.0	83.3	56.3	0.0	0.0	0.0	0.0	0.0	0.0

NUMBER OF DIGITS IN STRING	% CORRECT IN THIS POSITION									
	1	2	3	4	5	6	7	8	9	10
1	100.0									
2	0.0	0.0								
3	100.0	83.3	100.0							
4	93.8	87.5	75.0	96.9						
5	0.0	0.0	0.0	0.0	0.0					
6	0.0	0.0	0.0	0.0	0.0	0.0				
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0			
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0		
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

WORD SPOKEN	NUMBER OF REPEATS	% CORRECT
ONE	16	81.3
TWO	16	75.0
THREE	16	87.5
FOUR	16	93.8
FIVE	16	100.0
SIX	16	93.8
SEVEN	16	93.8
EIGHT	16	100.0
NINE	16	93.8
ZERO	16	81.3
POINT	15	93.3

TABLE 6. SUMMARY OF RECOG RECOGNITION ACCURACY IN FOUR TALKERS.*

Number of Digits in String	Number of Strings per Talker	Average % Correct in This Position (Range in Parentheses)			
		Talker			
		1	2	3	4
1	11	100	–	–	–
3	12	81.2 (58 – 100)	75.0 (42 – 92)	95.8 (92 – 100)	–
4	32	89.1 (84 – 94)	89.9 (84 – 94)	73.4 (66 – 78)	96.9 (94 – 100)

* This is the rejected approach.

APPENDIX B

LCSR TECHNICAL MEMO 003

LOGICON INTEROFFICE CORRESPONDENCE

TO: LCSR TM-003 Revision 3

FROM: J. E. Porter

SUBJECT: Locating Words Within Utterances

DATE: 3 February 1977

NO:

DISTRIBUTION: LCSR B

Locating Words Within Utterances

This note describes a method for determining a portion of an utterance in the data base which, with high confidence, contains the letters associated with a specific word which occurs within the utterance. The method is based on a simple statistical model of the time duration of spoken words within utterances, and has two parts; a mathematical procedure for extracting some statistics from observed time duration data, and formulas for finding the portion of the utterance where a word occurs, using the extracted statistics.

The Model

This procedure is based on the idea that the length of the portion of an utterance corresponding to a given word can be approximated as a product of three factors: the length of the utterance , a characteristic length of the word, and a "stretch factor" which depends on the number of words in the utterance and the ordinal position of the word within the utterance. Denote the characteristic lengths as

$$\ell(w) \text{ for } w \in \{".", "0", "1", \ldots "9"\}$$

and the stretch factor associated with the i^{th} position in an utterance of n words as $s(n, i)$.

Only the relative lengths of words within an utterance are needed to isolate the word, and therefore we are free to consider only the relative characteristic word lengths and relative stretch factors. We therefore may set

$$\ell(".") = 1$$

and

$$s(n, 1) = 1 \text{ for all } n,$$

indicating that ℓ is the characteristic length of a word relative to "point", and $s(n, i)$ is the stretch factor associated with the i^{th} word, relative to the first word, in an utterance of n words.

Let an utterance u consist of the words w_1, $w_2 \ldots w_n$. By counting the letters received from the VIP-100, and noting which letters belong to which words, the time duration of each word in the utterance can be established, as letters are received at a rate of one every 2 milliseconds. Let N_1, N_2, $N_3, \ldots N_n$ be the last letter corresponding to each word in the utterance. Then, if our model of relative word lengths within u were precisely correct, we would have for each word w_i in the utterance

$$\frac{N_i - N_{i-1}}{N_n} = \frac{\ell(w_i)\, s\,(n, i)}{\sum_{j=1}^{n} \ell(w_j)\, s\,(n, j)}$$

where $N_o = 0$. From this relation, one could develop formulae for the first and last letter of each word, and the problem would be solved (assuming the characteristic lengths and stretch factors were known). But speech is a very variable thing, and some mechanism must be introduced to accommodate the variation.

As the model already accounts for the major sources of variability (specifically: speech speed as reflected in the utterance length, relative word length, and word positional effects), the remaining variation may as well be treated in a conceptually simple way.

One such simple way, adopted here, is to introduce an 'adjustment' (fudge) factor on the length of the word being located. In this way, if a word begins sooner than would be predicted by the simple model, it is treated as being caused by an unusually long duration of the word itself, while the other words in the utterance are "normal". Late terminations of a word can similarly be treated as arising from long renditions of the word. Short word lengths explain late beginnings and early terminations.

If the characteristic lengths and stretch factors are known, and the beginning and end points of a particular word are observed in a given utterance, both the beginning and the end points can be used to compute the length adjustment factor. Doing this for a reasonably large sample, the statistics of the length adjustment factor should become clear, and a value unlikely to be exceeded can be chosen. This safe value can then be used to compute safe limits within which a given word will usually appear within an utterance.

Deriving the Statistics

There are two statistical problems to be addressed: first to find the characteristic lengths and stretch factors, and second to find a safe upper value for the length adjustment factor.

The characteristic lengths and stretch factors are found by least squares fitting of those variables to match some hand-marked data. The hand-marked data consists of a set, E, of utterances, each utterance containing from two to four known words $(w_1, \ldots w_n)$ and the numbers N_i of the last letter of each word. The procedure for least-squares fitting of the lengths and stretch factors is described below.

For each word in each example in E, we can define an error quantity, equal to the difference between the observed fractional length of the word and the fractional length predicted by our simple model. The sum of the squares of all these errors is then a suitable criterion to minimize. The criterion is then expressed

$$C = \sum_{e \in E} \sum_{i=1}^{n} \left(R_i - \frac{\ell(w_i)\, s(n, i)}{T} \right)^2$$

where $R_i = (N_i - N_{i-1}) / N_n$ (where $N_o = 0$)

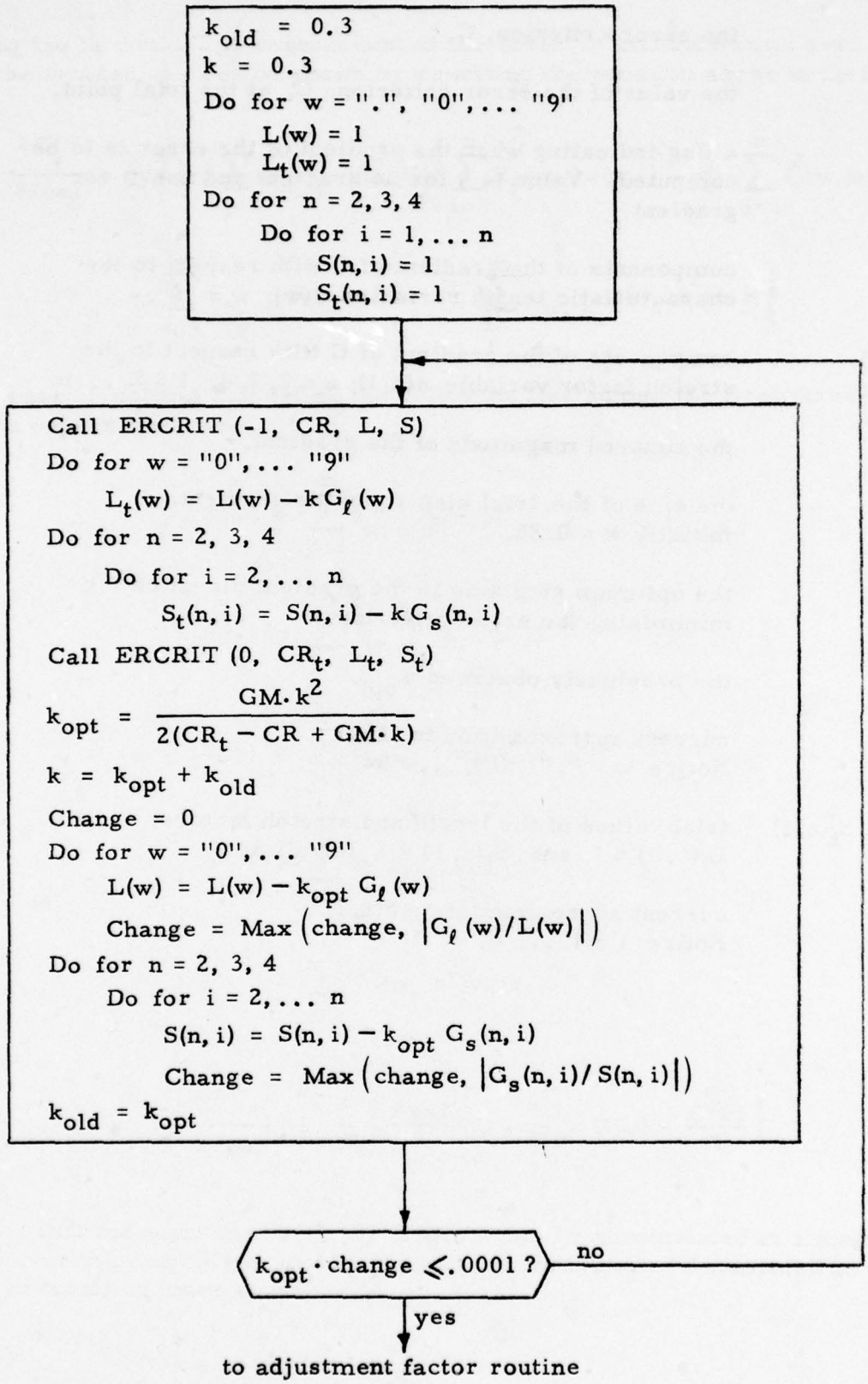
and $$T = \sum_{j=1}^{n} \ell(w_j)\, s(n, j)$$

The minimization is accomplished by successive approximations, using the method of steepest descent. In this procedure, the error criterion, C, is treated as a scalar point function in the space of undetermined length and stretch variables. The error value and its gradient are calculated at the present approximation, and a step is taken in the negative gradient direction, i.e., in the direction of steepest descent of C. The value of C is computed at the new point and, using the two values of C and the gradient calculated at the first point, an approximation to the minimizing position along the gradient direction is computed. The process is repeated, starting from the new approximation point, until the length and stretch factors remain essentially constant. The size of the "testing" step along the gradient is also updated.

Here is a flowchart for this part of the process. It uses a subroutine, ERCRIT, for computing the error criterion and, when required, the gradient and its squared magnitude. The data structures are described first.

Change	the maximum absolute value of the relative change in any control variable in a given step.

CR	the error criterion, C.
CR_t	the value of the error criterion, C, at the trial point.
GFLG	a flag indicating when the gradient of the error is to be computed. Value is 0 for no gradient and non-0 for gradient.
$G_{\ell}(w)$	components of the gradient of C with respect to the characteristic length variables $\ell(w)$; w = "0", ... "9".
$G_s(n, i)$	components of the gradient of C with respect to the stretch factor variable $s(n, i)$; n = 2, 3, 4, i = 2, ...n.
GM	the squared magnitude of the gradient.
k	the size of the trial step along the gradient. Initially k = 0.25.
k_{opt}	the optimum step size in the gradient direction for minimizing the error criterion.
k_{old}	the previously observed k_{opt}.
L(w)	current approximation to $\ell(w)$. Notice w = ".", "0", ... "9".
$L_t(w)$, $S_t(n, i)$	trial values of the length and stretch factors; $L_t(".") = 1$ and $S_t(n, 1) = 1$ for all n.
S(n, i)	current approximation to S(n, i). Notice i = 1, ... n.


$k_{old} = 0.3$
$k = 0.3$
Do for w = ".", "0", ... "9"
L(w) = 1
$L_t(w) = 1$
Do for n = 2, 3, 4
Do for i = 1, ... n
S(n, i) = 1
$S_t(n, i) = 1$
Call ERCRIT (-1, CR, L, S)
Do for w = "0", ... "9"
$L_t(w) = L(w) - k G_\ell(w)$
Do for n = 2, 3, 4
Do for i = 2, ... n
$S_t(n, i) = S(n, i) - k G_s(n, i)$
Call ERCRIT (0, CR_t, L_t, S_t)
$k_{opt} = \frac{GM \cdot k^2}{2(CR_t - CR + GM \cdot k)}$
$k = k_{opt} + k_{old}$
Change = 0
Do for w = "0", ... "9"
$L(w) = L(w) - k_{opt} G_\ell(w)$
Change = Max (change, $|G_\ell(w)/L(w)|$)
Do for n = 2, 3, 4
Do for i = 2, ... n
$S(n, i) = S(n, i) - k_{opt} G_s(n, i)$
Change = Max (change, $|G_s(n, i)/S(n, i)|$)
$k_{old} = k_{opt}$
$k_{opt} \cdot change \leq .0001$?
no
yes
to adjustment factor routine

The difficult part of this calculation is in computing the error criterion, C, and its gradient efficiently. In particular, this means avoiding nested loops over the data, control variables and words within examples. The necessary equations are developed below.

If we introduce an error associated with each example, e, consisting of words $w_1, w_2, \ldots w_n$, with data $N_1, \ldots N_n$, defined by

$$C_e = \sum_{i=1}^{n} \left[R_i - \frac{\ell(w_i)\, s(n,i)}{T} \right]^2$$

Then we have $C = \sum_{e \in E} C_e$

and the components of the gradient of C with respect to the control variables are

$$\frac{\partial C}{\partial \ell(w)} = \sum_{e \in E} \frac{\partial C_e}{\partial \ell(w)} \qquad w = \text{"0"}, \ldots \text{"9"}$$

$$\frac{\partial C}{\partial s(n,i)} = \sum_{e \in E} \frac{\partial C_e}{\partial s(n,i)} \qquad n = 2,\ 3,\ 4, \quad i = 2, \ldots n.$$

If the partial derivatives of the individual word error terms can be computed, then the components of the gradient can be computed in a single pass over the data, as these formulae show.

Let w be a word from "0" to "9", which may or may not occur in the example e, which contains words $w_1 \ldots w_n$. Then we have

$$\frac{\partial C_e}{\partial \ell(w)} = 2 \sum_{i=1}^{n} \left[R_i - \frac{\ell(w_i)\, s(n,i)}{T} \right] \frac{\partial}{\partial \ell(w)} \left(- \frac{\ell(w_i)\, s(n,i)}{T} \right)$$

$$= 2 \sum_{i=1}^{n} \left[R_i - \frac{\ell(w_i)\, s(n,i)}{T} \right] \left[\frac{\ell(w_i)\, s(n,i)}{T^2} \sum_{j=1}^{n} \delta_{w, w_j}\, s(n,j) - \delta_{w, w_j} \frac{s(n,i)}{T} \right]$$

where $\delta_{w, w_j} = \begin{cases} 1 & \text{if } w = w_j \\ 0 & \text{otherwise} \end{cases}$

Using the fact that T is independent of the index i, multiple sums over i can be avoided, as can be shown by rewriting the equation above in the form:

$$\frac{\partial C_e}{\partial \ell(w)} = \frac{2}{T}\left\{\frac{1}{T}\left(\sum_{j=1}^{n}\delta_{w,w_j}\, s(n,j)\right)\left[\sum_{i=1}^{n}R_i\,\ell(w_i)\,s(n,i) - \frac{1}{T}\sum_{i=1}^{n}\ell^2(w_i)s^2(n,i)\right]\right.$$

$$\left. - \sum_{i=1}^{n}\delta_{w,w_i}\,R_i\,s(n,k) + \frac{1}{T}\sum_{i=1}^{n}\delta_{w,w_i}\,\ell(w_i)\,s^2(n,i)\right\}$$

For ease of calculation define the following five sums, the first three of which depend upon w:

$$A(w) = \sum_{j=1}^{n}\delta_{w,w_j}\,s(n,j)$$

$$B(w) = \sum_{j=1}^{n}\delta_{w,w_j}\,R_j\,s(n,j)$$

$$C(w) = \sum_{j=1}^{n}\delta_{w,w_j}\,\ell(w_j)\,s^2(n,j)$$

$$D = \sum_{j=1}^{n}R_j\,\ell(w_j)\,s(n,j)$$

$$E = \sum_{j=1}^{n}\ell^2(w_j)\,s^2(n,j)$$

Then

$$\frac{\partial C_e}{\partial \ell(w)} = \frac{2}{T}\left\{\frac{A(w)}{T}\left[D - \frac{E}{T}\right] - B(w) + \frac{C(w)}{T}\right\}$$

Notice that the sums A, B, C, D, E and T can be accumulated in a single pass over the example e, and the partial derivative can be computed in one step in terms of those sums.

Proceeding similarly for the gradient components corresponding to the stretch factors,

$$\frac{\partial C_e}{\partial s(n,k)} = 2\delta_{m,n} \sum_{i=1}^{n} \left[R_i - \frac{\ell(w_i)\, s(n,i)}{T} \right] \frac{\partial}{\partial s(n,k)} \left(- \frac{\ell(w_i)\, s(n,i)}{T} \right)$$

$$= 2\delta_{m,n} \frac{\ell(w_k)}{T} \left[\frac{1}{T} \sum_{i=1}^{n} R_i\, \ell(w_i) s(n,i) - \frac{1}{T^2} \sum_{i=1}^{n} \ell^2(w_i) s^2(n,i) - R_k + \frac{\ell(w_k)\, s(n,k)}{T} \right]$$

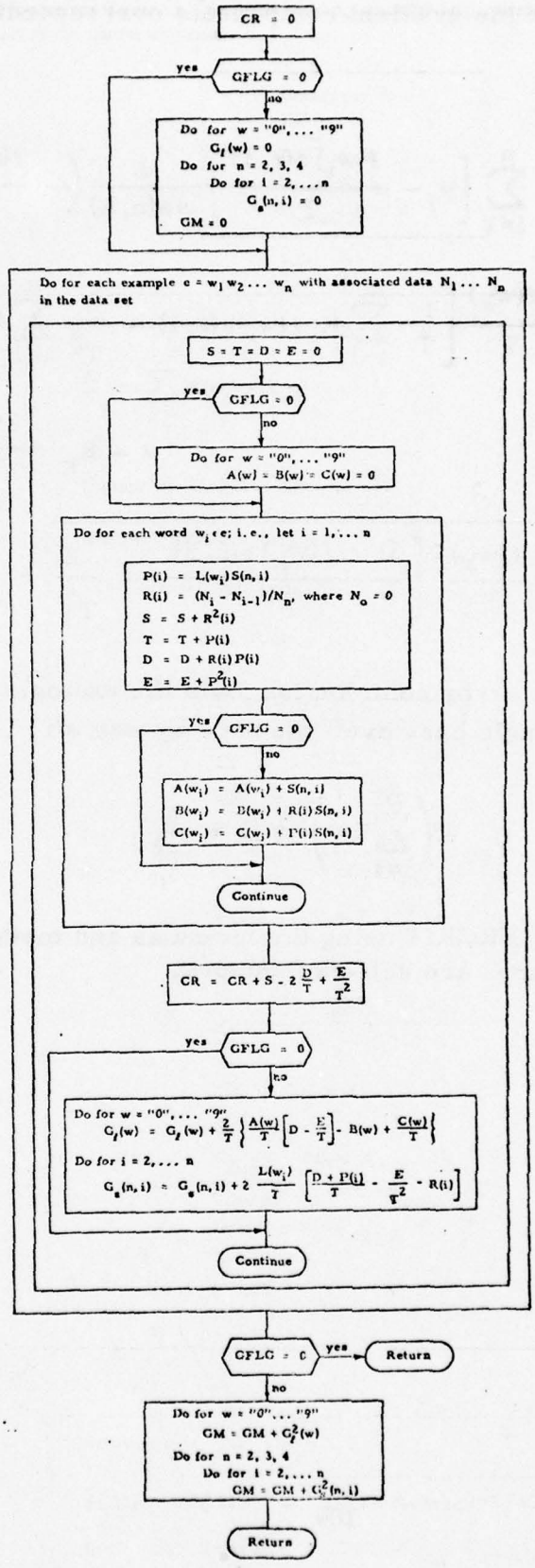
$$= 2\delta_{m,n} \frac{\ell(w_k)}{T} \left[\frac{D + \ell(w_k)\, s(n,k)}{T} - \frac{E}{T^2} - R_k \right]$$

Finally, we note that the error contribution from the example e, C_e, can also be computed in a single pass over the data by way of:

$$C_e = \left(\sum_{i=1}^{n} R_i^2 \right) - 2\frac{D}{T} + \frac{E}{T^2}$$

Below is a flowchart for ERCRIT using the formulas and method developed above. The data structures are self-explanatory.

ERGRIT: calling sequence (GFLG, CR, L, S)
returns CR, G_f, G_s, GM



To obtain the statistics of the adjustment factors, a set of adjustment factors is computed from the hand segmented data, and these values are printed out for examination by hand. The maximum, minimum, average and standard deviation are also calculated and printed.

As indicated in the discussion of the model, adjustment factors are computed from formulae based on the assumption that observed word beginning and end points are explained by expansion or contraction of the word in question. If the adjustment factor for w_i, the i^{th} word in an n-word utterance, is denoted F, then the observed end point of the preceding word (N_{i-1}) is accommodated by F when

$$\frac{N_{i-1}}{N_n} = \frac{\sum_{j=1}^{i-1} \ell(w_j)\, s(n,j)}{\sum_{j=1}^{n} \ell(w_j)\, s(n,j) + (F-1)\, \ell(w_i)\, s(n,i)}$$

So the left end point of w_i (when $i \neq 1$) leads to an adjustment value estimate

$$F_{left} = 1 + \frac{1}{\ell(w_i)\, s(n,i)} \left[\frac{N_n}{N_{i-1}} \sum_{j=1}^{i-1} \ell(w_j)\, s(n,j) - \sum_{j=1}^{n} \ell(w_j)\, s(n,j) \right]$$

Similarly, the right hand end point of w_i is accommodated by the adjustment factor F if $i \neq n$ and

$$\frac{N_i}{N_n} = \frac{\sum_{j=1}^{i-1} \ell(w_j)\, s(n,j) + F\, \ell(w_i)\, s(n,i)}{\sum_{j=1}^{n} \ell(w_j)\, s(n,j) + (F-1)\, \ell(w_i)\, s(n,i)}$$

which leads to the adjustment value estimate

$$F_{right} = \frac{N_i \sum_{j \neq i} \ell(w_j)\, s(n,j) - N_n \sum_{j=1}^{i-1} \ell(w_j)\, s(n,j)}{(N_n - N_i)\, \ell(w_i)\, s(n,i)}$$

Here is a flowchart for computing and printing the right and left estimates of adjustment factors, and the mean, maximim, minimum and standard deviation of those estimates.

Count = 0
$\Sigma F = 0$
$\Sigma F^2 = 0$
$F_{big} = 1$
$F_{small} = 1$

Do for each utterance $u = w_1, w_2 \ldots w_n$ with data $N_1, N_2 \ldots N_n$

$$T = \sum_{j=1}^{n} \ell(w_j)\, s(n, j)$$

$P = 0$

Count = Count + 2(n - 1)

Do for each $i = 1, \ldots n$

$D = \ell(w_i)\, s(n, i)$

If $i \neq 1$, then

$$F = 1 + \frac{1}{D}\left(\frac{PN_n}{N_{i-1}} - T\right)$$

$\Sigma F = \Sigma F + F$
$\Sigma F^2 = \Sigma F^2 + F^2$
$F_{big} = \text{Max}(F_{big}, F)$
$F_{small} = \text{Min}(F_{small}, F)$
Print "F_{left} =", F

If $i \neq n$, then

$$F = \frac{(T - D)N_i - PN_n}{(N_n - N_i)D}$$

$\Sigma F = \Sigma F + F$
$\Sigma F^2 = \Sigma F^2 + F^2$
$F_{big} = \text{Max}(F_{big}, F)$
$F_{small} = \text{Min}(F_{small}, F)$
Print out "F_{right} =", F

$P = P + D$

Average = ΣF/Count

St Dev = $\sqrt{(\Sigma F^2 - \text{Count} \cdot \text{Average}^2)/(\text{Count} - 1)}$

Print out Average, St Dev, F_{big}, F_{small}

Application

Once a value of the adjustment factor is chosen which is unlikely to be exceeded (denoted $\overline{F}$), a word w_i which appears at the i^{th} position in an utterance $n = w_1, w_2 \ldots w_n$ can be expected to lie entirely within the range of letter counts N_{left} and N_{right}, where

$$N_{left} = \frac{N \sum_{j=1}^{i-1} \ell(w_j)\, s(n, j)}{\sum_{j=1}^{n} \ell(w_j)\, s(n, j) + (\overline{F} - 1)\, \ell(w_i)\, s(n, i)}$$

$$N_{right} = \frac{N \left(\sum_{j=1}^{i-1} \ell(w_j)\, s(n, j) + \overline{F}\, \ell(w_i)\, s(n, i) \right)}{\sum_{j=1}^{n} \ell(w_j)\, s(n, j) + (\overline{F} - 1)\, \ell(w_i)\, s(n, i)}$$

where N is the total number of letters in the utterance.

Results

The VIP-100 and associated computer equipment were used to collect feature data for 119 utterances spoken by the LCSR project voice (MG). The utterances consisted of 393 words in groups of 2, 3 and 4, approximately uniformly distributed over the LCSR vocabulary. By studying listings of the feature strings, it was possible to identify characteristic patterns associated with each vocabulary word, and eventually to mark the approximate boundaries between words. These hand-marked data were used to develop the input numbers for use in the procedures described above.

When the length and stretch factors were computed for this hand-marked data, and those factors were used to compute the adjustment factors, one of the input utterances led to a much larger adjustment factor value than all the other data. Upon examining the feature string for that utterance, the last word (a "7") was found to be anomalously long. This utterance was discarded, leaving 118 utterances and 389 words in the data set.

Re-running the procedures on the reduced set of data led to the length and stretch factors given in tables 7 and 8.

The minimum error criterion (C) found for these data was 0.5872. As there are 389 words in the data, this corresponds to a root mean square error of 3.9% in predicting the fractional lengths of those words. Apparently the simple model used here fits the data remarkably well!

TABLE 7. NOMINAL DURATIONS OF LCSR VOCABULARY WORDS, IRRESPECTIVE OF POSITION IN 2-, 3-, AND 4-WORD UTTERANCES, NORMALIZED WITH RESPECT TO THE WORD "POINT."

word	.	0	1	2	3	4	5	6	7	8	9
length	1.000	1.168	0.788	0.870	0.983	1.055	1.147	1.178	1.111	0.773	0.969

TABLE 8. NOMINAL DURATIONS OF WORDS AS A FUNCTION OF THEIR POSITION WITHIN 2-, 3-, AND 4-WORD UTTERANCES, IRRESPECTIVE OF THE WORD SPOKEN, NORMALIZED WITH RESPECT TO THE DURATION OF THE FIRST WORD IN THE UTTERANCE.

Number of Words in Utterance	Location of Word in Utterance			
	1st	2nd	3rd	4th
2	1.000	1.580	---	---
3	1.000	1.086	1.446	---
4	1.000	1.152	1.079	1.461

The length and stretch factors were used to predict placement of words within utterances, and the adjustment factors were computed for the right and left word boundaries. The adjustment factors were ordered, and their cumulative distribution was plotted. The result is shown in figure 15. As indicated in that figure, 271 values of the right and of the left adjustment factor were computed. The right and left values are quite smoothly and quite similarly distributed, giving grounds for confidence that a reliable upper bound can be chosen. As indicated in the figure, the maximum observed adjustment factor value was 2.51 and the upper limit chosen was 2.75.

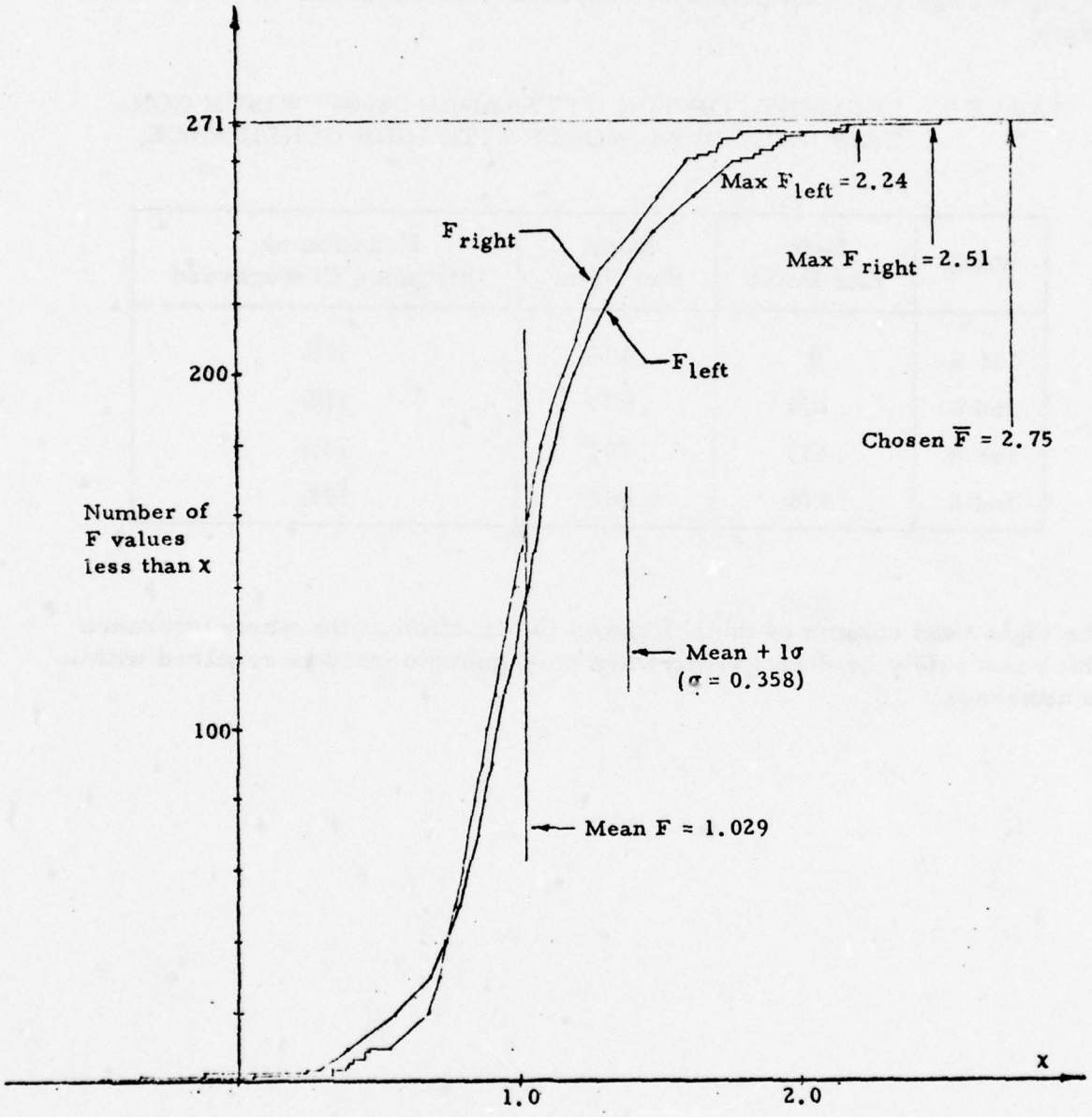


Figure 15. Cumulative Distribution of Adjustment Factors.

As an example of the use of these results, consider the utterance "6688", made up of the longest and shortest words in the LCSR vocabulary. Using the data presented above, right and left end points can be computed which will, with considerable confidence, contain each individual word. Table 9 gives the results of those calculations, using formulae similar to those at the top of page 113. End points are expressed as a fraction of the utterance length.

TABLE 9. SEGMENTS OF THE UTTERANCE "6688" WHICH CONTAIN INDIVIDUAL WORDS WITH HIGH CONFIDENCE.

Word	Left End Point	Right End Point	Fraction of Utterance Disregarded
1st 6	0	.494	51%
2nd 6	.180	.673	51%
1st 8	.433	.797	64%
2nd 8	.576	1.000	58%

The right hand column of table 9 shows the fraction of the whole utterance which can safely be disregarded when only a single word is required within an utterance.

APPENDIX C

LCSR TECHNICAL MEMO 017

LOGICON INTEROFFICE CORRESPONDENCE

TO: LCSR TM 017

FROM: J. E. Porter

SUBJECT: LOOPER

DATE: 1 May 1977
NO: ASD-77-212
DISTRIBUTION: LCSR "B"

Reference: This memo supersedes TM 011 (ASD-77-126) dated 14 March 1977

This memo is intended to describe in detail the LOOPER processing outlined in Notes on Automatic Generation of Mathematical Machines (SDR-203). Some corrections to the original are also included.

LOOPER operates mulitple copies of a machine simultaneously and constructs a whole array of loop letter sets, then purges it.

First an array of loop letter-sets is created,

$$\begin{array}{ccc} L_1^{(1)} & \cdots\cdots\cdots & L_{K-1}^{(1)} \\ \vdots & & \vdots \\ L_1^{(C)} & \cdots\cdots\cdots & L_{K-1}^{(C)} \end{array}$$

for each utterance in the example space. Each row of the array corresponds to a copy of the machine. The utterance is processed from left to right, keeping track of the progress of each active copy of the recognition machine. For each copy of the machine, and hence for each row r of the array, there is a state number, S_r, and a flag f_r which tells whether or not loop letters are being accummulated. (This will be explained later).

As new letters are processed, all the active copies of the machine are updated, and then a new copy of the machine is started if:

a) the new letter is in T_1, and
b) there is not a machine copy already in state 1 with $L_1 = \phi$ (the empty set).

Let C be the number of machine copies currently being used, hence the number of rows of the array which have to be processed. New copies are added at the bottom, so to determine whether a new copy of a machine needs to be started, it is only necessary to check that the letter being processed is in T_1, and copy number C of the machine is not in state 1 or $L_1^{(C)}$ is not the empty set.

When a copy of the machine reaches the recognition state, its state number is changed to K, and that copy need not be processed further. Thus active machines are those in States 1, 2, --K-1, while those in state K are inactive.

Updating a machine copy entails several considerations. If the machine copy is currently in state S, and the input letter is A, do the following when S<K:

a) If $a \in T_{S+1}$ do transition processing

b) If the f flag is TRUE, we are accummulating loop letters, set $L_S = L_S va$ and go the next copy

c) If $a \notin T_S$, then we must start accummulating loop letters, set f to TRUE, and $L_S = \{a\}$.

Transition processing handles deactivation and coalescing of machines, as well as simple transition. Deactivation occurs when the machine reaches recognition state and is easily done:

a) If $S_r = k-1$, set $S_r = k$ and exit.

When a machine copy moves into a state which is also occupied by another copy, it may be possible to eliminate the moving machine. The test is simply whether or not all the loop letter sets accummulated so far for the two machines are equal. Since new machines are added at the bottom of the array, and since machines can only move forward, it is sufficient to check the rows above the one being processed, until one is found which is in a state beyond the state being entered. For those which are in the state being entered, equality of the loop letter set is checked. Finally, if the moving machine can be eliminated, all the machine copies below it are moved up in the array, to keep the array small and dense. So the rest of transition processing is as follows, where r is the index of the row currently being processed:

b) $S_r = S_r + 1$

f_r = FALSE

$L_{S_r}^{(r)} = \phi$

c) If r = 1, exit

d) Work upward in the array, starting at row r-1, and check to see if the state of each machine (row) is S_r. As soon as one is found with state different from S_r, exit. For those found with state equal to S_r, do the following:

Check to see if $L_i = L_i^{(r)}$, for $i = 1, \ldots s_r$

If yes, eliminate this machine copy by moving all lower rows of the array, and states and flags, up one, and finally setting C=C-1. Then exit, terminating transition processing.

After the array of loop letter set is created, as described above, it must be purged of rows corresponding to machine copies which never quite make it to the recognition state, and any row whose letter sets all contain the sets of some one other row. A flow chart for the latter process, given on page 43 of SDR-203, is reproduced here as figure 16.

Finally we open the L_i^* to contain the intersection of all the rows of the array. If there is only one row, we are finished with this utterance. If there are more, we must ask if any one row is contained in the row of L_i^*. If yes, this utterance is already accommodated and can be discarded. If no, the array must be placed in a file for processing by OVERPASS.

OVERPASS processing, also described in SDR-203, opens the L_i^* to accommodate the most characteristic start point in each remaining utterance. The flow chart is reproduced here as figure 17.

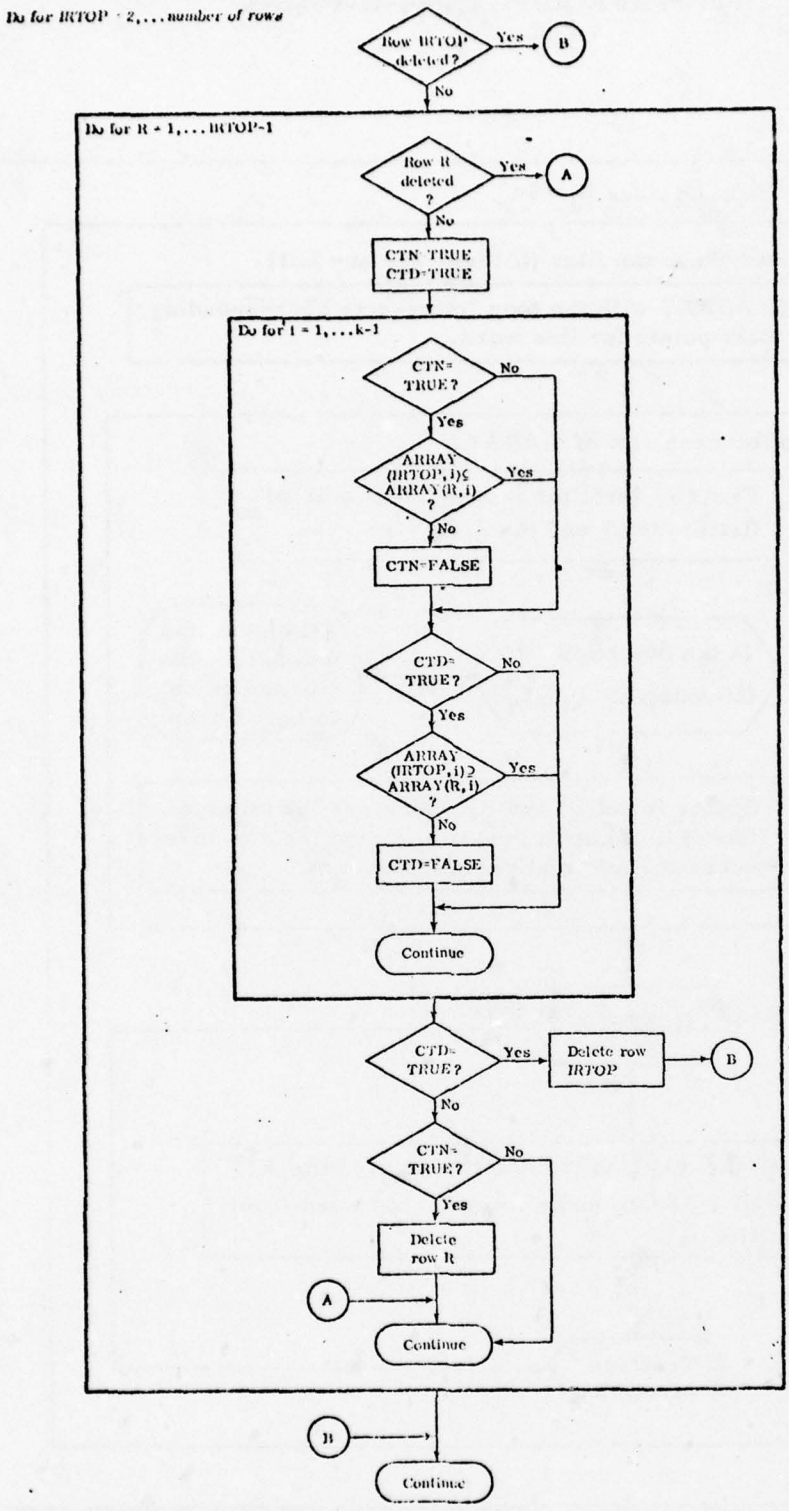


Figure 16. LOOPER Processing for Purging the Array of Loop Letter Sets of those Rows which Contain Other Rows.

OVERPASS

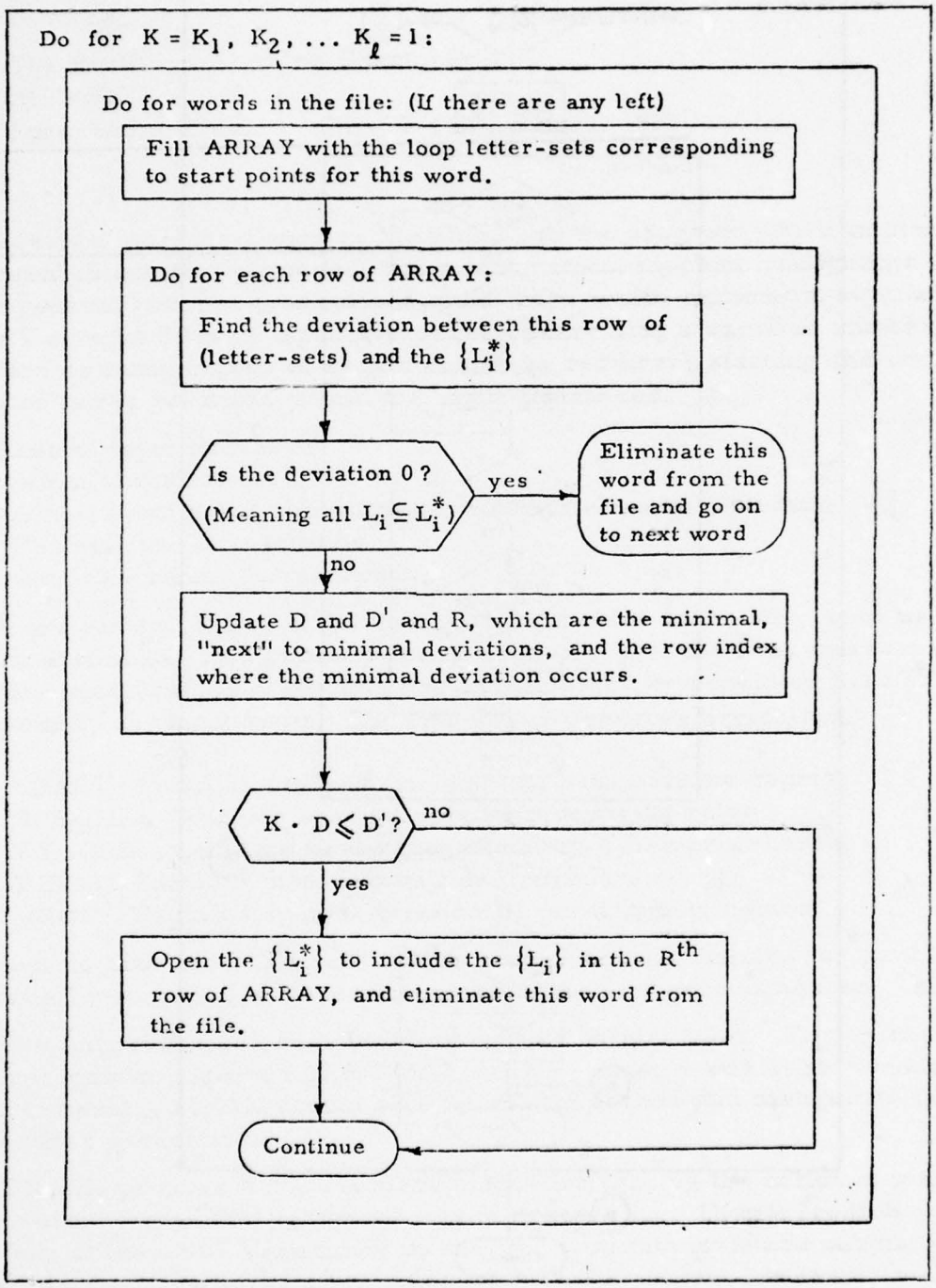


Figure 17. OVERPASS Processing.

APPENDIX D

LCSR TECHNICAL MEMO 018

LOGICON INTEROFFICE CORRESPONDENCE

TO: LCSR Notebook TM-018

DATE: 1 May 1977
NO: ASD-77-211
DISTRIBUTION: LCSR "B"

FROM: J. Porter

SUBJECT: The Research Machine Exerciser (REMEX)

Reference: Supersedes TM-013 (ASD-77-161) 6 April 1977

The Research Machine Exerciser (REMEX) is the program which implements, for research and test purposes, the mathematical machine recognition algorithm. It is a general purpose generalized multi-copy multi-automaton simulator. REMEX accepts files of input utterances, exercising a variable number of machines (and as many copies of each machine as required) starting and ending at specified letter positions within the input utterances.

The kinds of input data are:

1. A file of utterances
2. A file of start and end points in those utterances (optional)
3. The number of machines
4. Data describing each machine.

There are several things to be done with a machine exerciser, such as accumulate counter statistics, find primary letter sets, and test the final machines. These various special things to be done are implemented by providing REMEX with specialized sub-procedures. The five sub-procedures are called:

1. START: Contains criteria for starting new machine copies
2. SPECIAL: Contains several special processing rules
3. RECONIZ: Contains final recognition and other procedures
4. FINAL: Contains post-recognition procedures
5. DROP: Contains special procedures for dropping copies.

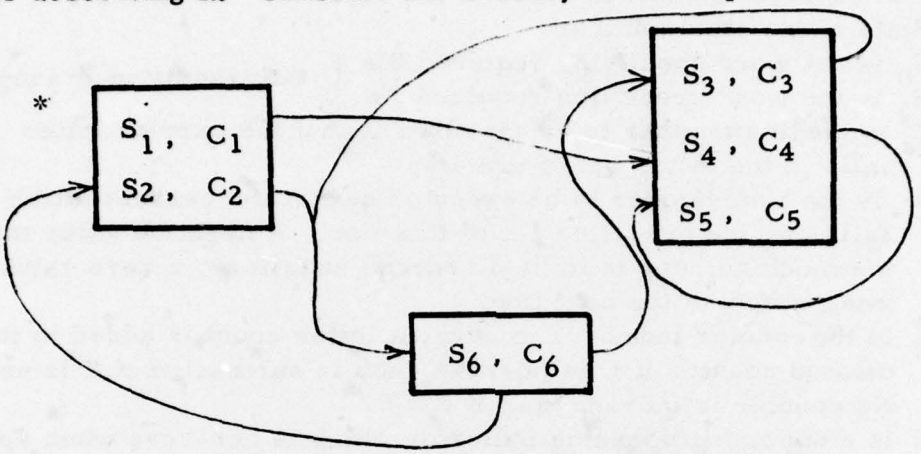
The hope is that REMEX itself will be static, and specialized for particular purposes by supplying different versions of the five subprocedures.

The five subprocedures may build or access special files. For example, in the final recognition algorithm RECONIZ will use counter statistics to compute recognition criteria. FINAL should also be able to access and manipulate files generated in the other procedures.

REMEX will produce some standard output for tracing the action of machines and copies of machines through the utterance processing. Output from the other procedures, if any, will sometimes be merged with this standard output, but more often delayed for printing after standard REMEX output, for example in FINAL.

Machine descriptive data

A graphic description of a machine is useful in understanding the data structure used for describing it. Consider the unlikely device depicted below:



Each box corresponds to a state, which will be identified by a number. The machine starts in the state marked by the "*". Within a state there are one or more pairs of letter sets and counter identifiers. Each pair is a line, and will correspond to a line, or row, of descriptive data. In operation, a machine copy starts at a particular line and checks the incoming letter to see if the letter is in the letter set of its current line. If yes, the counter identified in the current line is incremented by the incoming letter count, and the line at the end of the outgoing arrow becomes the current line. Additional processing may be performed after incrementing the counter and line indicator. If the incoming letter does not belong to the current line's letter set, the letter is checked against the next line's letter set, and so on, until it is found to fall in a letter set, or the last line of the state has been tried. If the last line of the state has been tried and the letter has not been a member of any letter set, the machine copy is dropped if there is no arrow from the bottom of the box; if there is such an arrow, it points to the next line to be executed.

Notice that the concept of "the current line" belongs to a copy of a machine, but the concept "line" itself is just descriptive of the machine automaton. A given machine is described by specifying a sequence of 7-tuples, one for each line in the machine. The data for each line given below are slightly redundant and actually allow simulating more general machines than those described above. Each line contains the following data:

State, S_0, S_1, l_s, l_f, C, SPI

where State is the state number

S_0 is the word specifying required 0's } the transition or loop letter set
S_1 is the word specifying required 1's }

l_s is the line number to be executed next if the current letter falls in the letter set of this line

l_f is the line number to be executed next if the current letter falls outside the letter set of this line. A negative value means the machine copy is to be discarded at failure, a zero value is interpreted as the next line.

$|C|$ is the counter index. The current letter count is added to the indicated counter if C is positive, and is subtracted if C is negative. No counter is incremented if C = 0.

SPI is a special processing indicator which is non-zero when special processing is required. One of the subprocedures is activated when the letter is in this line's letter set and SPI is positive, or when the letter test fails and SPI is negative. SPI's absolute value indicates which procedure is to be activated, and SPI is passed to that procedure as an indicator of what is required.

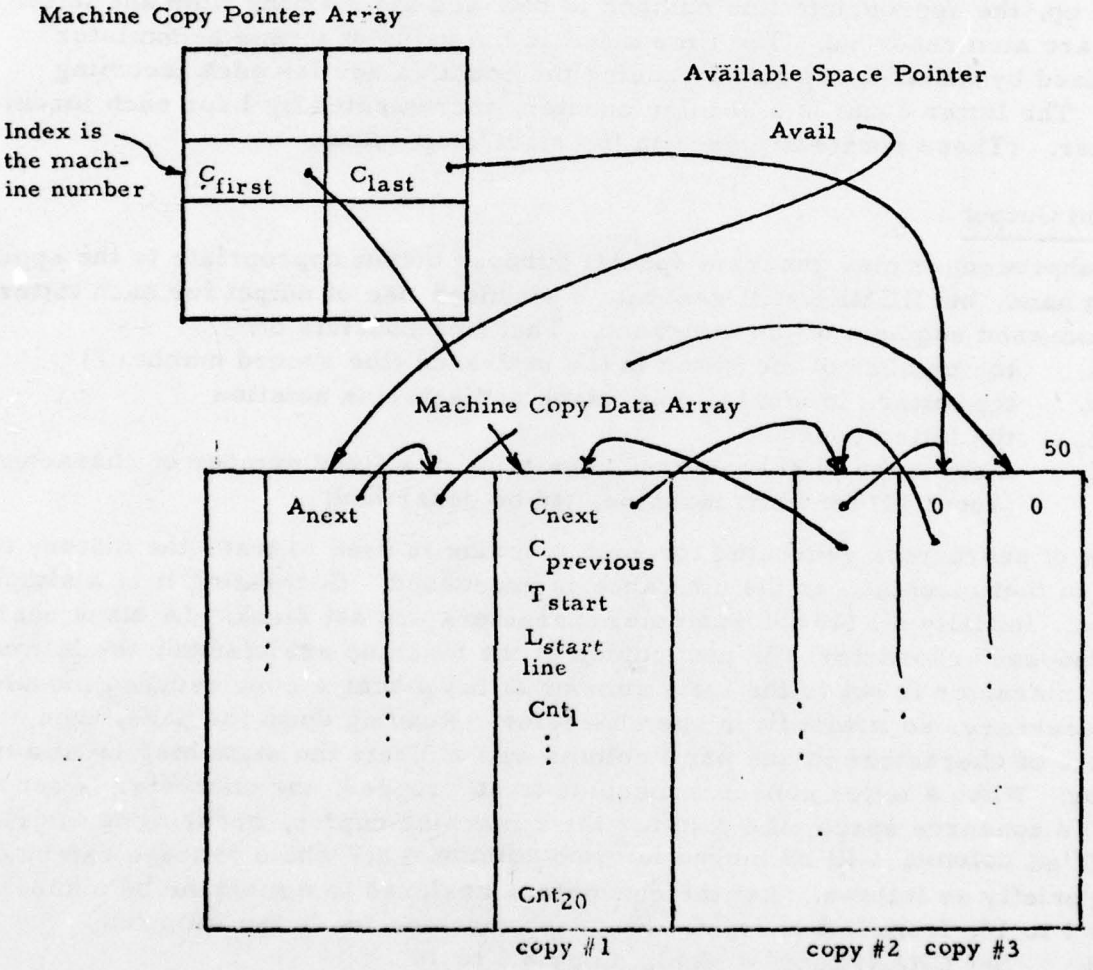
Each line in a machine is given a unique number, and the lines of a given state are given consecutive numbers. The example machine could be represented by the following data:

Line	State	S_0	S_1	l_s	l_f	C	SPI
1	1	S_{10}	S_{11}	4	0	0	0
2	1	S_{20}	S_{21}	6	-1	-1	-206
3	2	S_{30}	S_{31}	6	0	2	0
4	2	S_{40}	S_{41}	5	0	-3	0
5	2	S_{50}	S_{51}	0	-1	+1	102
6	3	S_{60}	S_{61}	5	2	2	0

In this example, three counters are used. The first counter (#1) will contain the difference between the number of letters ascribed to letter set S_5 and those ascribed to letter set S_2. The SPI value of 102 in line 5 might indicate that the procedure RECONIZ is called on receipt of a letter in S_5, and the SPI value of -206 might indicate that the procedure DROP is called when no letter set in state 1 is satisfied.

Machine Counter Data

REMEX must be capable of exercising at least 11 machines and perhaps 15 or 20 will be required eventually, if we develop special machines for special contexts. Several copies of each of these machines may be "active" at any one time, and (for research purposes at least) the counter values associated with any machine copy must be kept while that copy is active. Machine copies are also processed in a definite order: oldest one first, so it is necessary to be able to step through the copies in that order. However, logic for starting a new copy of a machine may entail data about the most recently started machine copy, so it is necessary to access that copy easily. Finally, copies may be dropped at any stage of the game (in any state) without disrupting the processing order. These requirements are all best satisfied by keeping machine copy data in a doubly linked list. The data structure recommended is illustrated below.



Space is provided for 50 copies of machines, with about twenty counters per copy. Processing is done one machine at a time, stepping through the Machine Copy Pointer Array. Copies of a given machine either first (oldest) or last (newest) may be accessed directly by way of pointers in that array. Data descriptive of each machine copy are kept in the Machine Copy Data Array, which is doubly linked. Zero values for the "next" or "previous" pointers indicate the ends of the list of machine copy data. Unused columns of the Machine Copy Data Array are singly linked and the top of the available list is pointed to by Avail.

When a new copy of a machine is activated, a column for the data is taken from the available list, all the counter values are set to zero, the double links are set up, the appropriate line number is set, and the starting time and letter counts are also recorded. The time count is the value of a time accumlator maintained by REMEX by simply adding the count values for each incoming letter. The letter count is a similar counter, incremented by 1 for each incoming letter. (These counts are used in the start/stop rules).

Standard Output

Each subprocedure may generate special purpose output appropriate to the application at hand, but REMEX will generate a standard line of output for each letter in (the processed segment of) an utterance. That line consists of:

a. the number of the letter in the utterance (the record number?)
b. the letter, in semi-colon, blank and asterisk notation
c. the letter count
d. a sequence of characters, consisting of a fixed number of characters (about 10) for each machine, (to be described)

The set of characters generated for each machine is used to trace the history of copies of that machine, as the utterance is processed. Generating it is a simple process. Initially all (10 for example) characters are set blank. (A blank character is an "unused" character.)As new copies of the machine are started, the leftmost unused character is set to the state number of the machine copy reduced module 10 if necessary, so it will fit in one character. Reading down the page, then, the sequence of characters in one print column will indicate the state history of a machine copy. When a letter causes a machine to be dropped, the character is set to "X". To conserve space, the data for later machine copies, occuring to the right of the X'ed column, will be moved left one column. This whole process can be described briefly as follows. Let the characters assigned to a machine be numbered Copy = 1 to 10, from left to right. For each machine we do the following:

1. Set CHAR(Copy) = blank, Copy = 1 to 10
2. Set Copy = 1
3. Proeess a machine copy. If the machine copy remains active, set CHAR (Copy) to the state of the machine copy, mod 10. If the machine copy is to be deleted, set CHAR (Copy) = "X"

4. If more machine copies are to be processed, set C = C+1 and go to 3; else quit.

We can add some bells and whistles to this. For example, when a machine copy makes it through all states and the RECONIZ procedure is activated to see if the potential recognition is a real one, the machine copy's output character may be set to "*" to represent a potential recognition.

What to do when there are more machine copies active than there are characters alloted? Let's be optimistic and believe this will happen only rarely, and probably not for more than one machine at a time. Then we should get away with temporarily allocating more characters to that machine. This will skew the output columns for following machines, but it should still be readable.

Major Processing Steps

1. Obtain the right versions of the subprocedures, machine descriptions, utterance files and start/stop points if available.

2. Do the following for each segment of each utterance to be processed.

   A. Initialize: Machine Copy Pointer Array
      Available Space Pointer
      Machine Copy Data Array (Put in single links)
      Time counter to zero
      Letter counter to zero

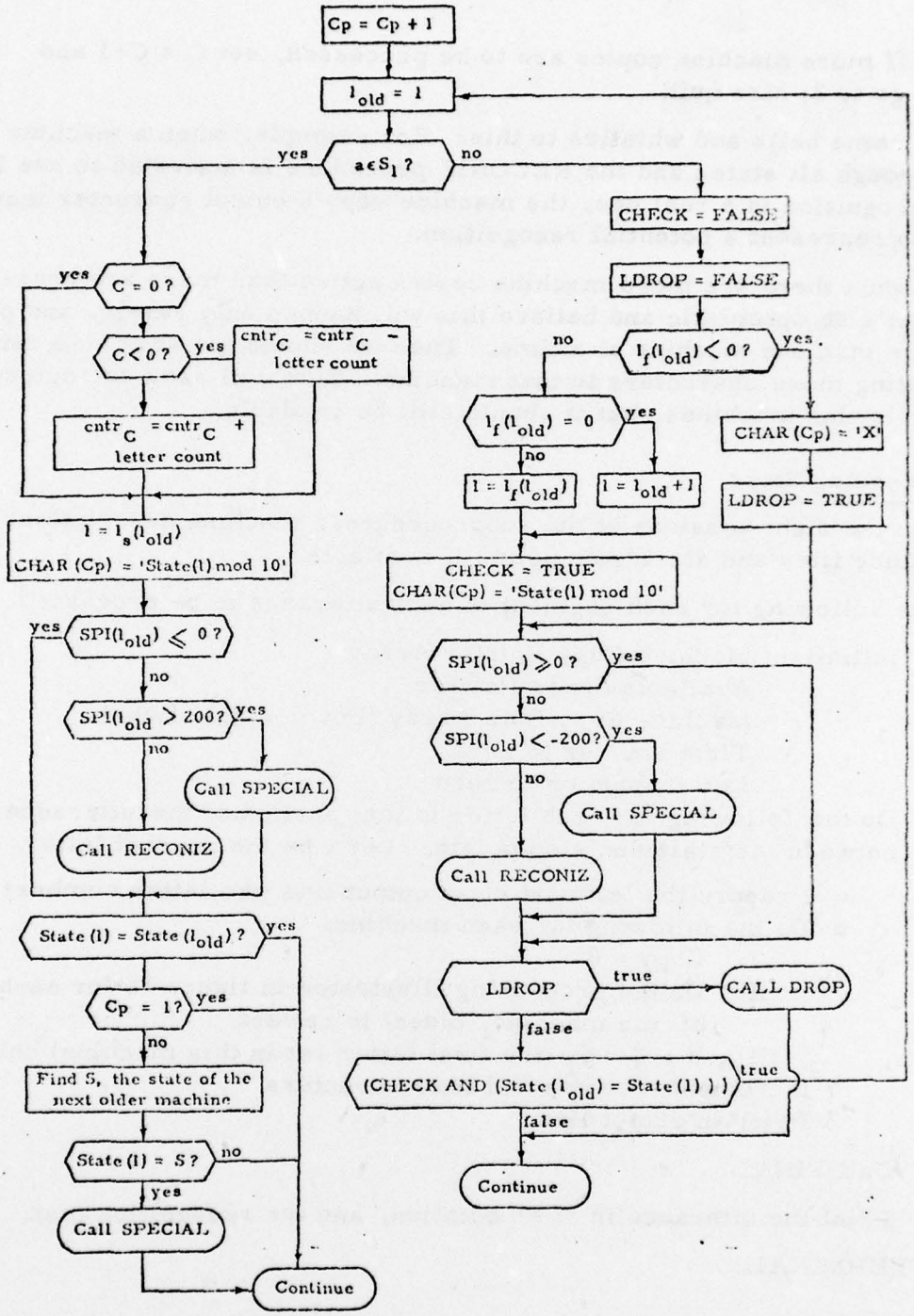
   B. Do the following for each letter in that portion of the utterance lying between the start and stop points. Let a be the current letter.

      α Prepare the left part of an output line (the letter number)
      β Do the following for each machine.
         i) Copy = 0
         ii) Do the processing illustrated in figure 18 for each copy of this machine, oldest to newest.
         iii) If a $\in S_1$ (the first letter set in this machine) call START
      γ Increment the time and letter counters
      δ Print an output line

   C. Call FINAL

   D. Print the utterance in "; *" notation, and the recognition map.

3. Call TERMINAL.



A is the letter, S_l is the letter set, l is the current line number, and Cp is the copy number.

Figure 18. Processing for Active Machine Copies.

The processing for each machine copy has some features not discussed earlier, viz:

1. When special processing is to be done, indicated by non-zero values of SPI, SPECIAL is called only if the absolute value of SPI is greater than 200, else RECONIZ is called.

2. When the machine copy transits to a new state, and that state is occupied by the next older machine SPECIAL is called. (SPECIAL will contain the processing to determine if a machine copy should be dropped when it overtakes an older copy.)

APPENDIX E

PROGRAM DESCRIPTIONS

The executable files produced for the LCSR project are described briefly in the following paragraphs. Figure 19 shows their use.

TMN ensures the number set file is formatted correctly. TMN checks to ensure that the data are in the appropriate columns, that the digit string and file name correspond, that the set descriptor is correct, and that "." follows the file name.

Extract is an end-user oriented program which:

a. Prompts the speaker to voice an utterance.

b. Saves the VIP features generated by that utterance on a disk file.

c. Compresses the features for LCSR processing.

d. Provides hardcopy printouts of both the raw feature data and compressed data.

WIZARD locates the words within utterances. The method is based on a simple statistical model of the time duration of the spoken words within utterances, and has two parts: a mathematical procedure for extracting some statistics from observed time duration data, and formula for finding the portion of the utterance where a word occurs, using the extracted statistics.

WIZARD writes the computed length and stretch factors into the file "WIZ. ST."

ESG builds an example space for a specified vocabulary item from specified compressed data files. Inputs to ESG include a list of the compressed

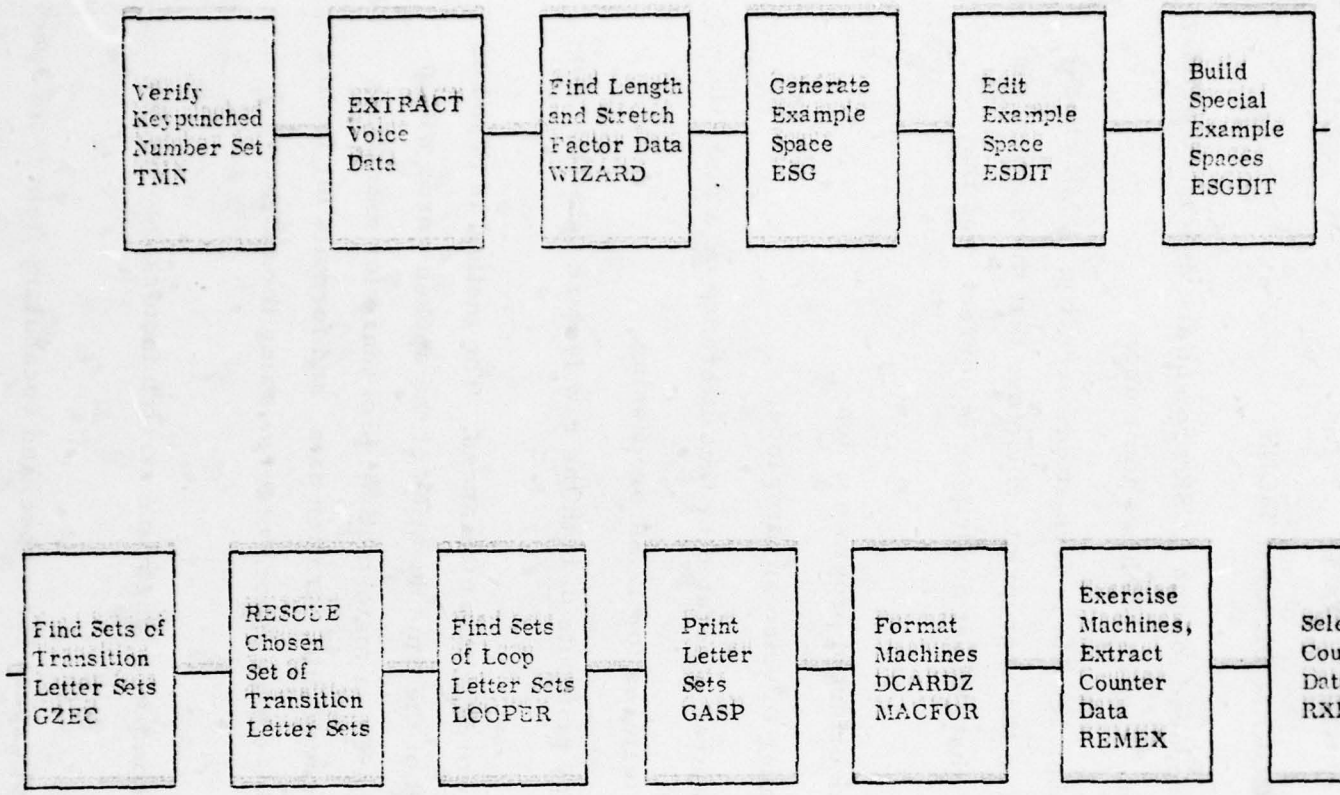


Figure 19. Use of LCSR Software.

data files which contain this item, and the set of length and stretch factors computed by WIZARD. ESG determines what portion of the utterance is most likely to contain that item, and writes the file name and starting and ending records to the example space file.

ESDIT is the example space file editor. It provides the capability to change individual start/stop values in the example space using an ESG or GENRLIZ printout.

For a GENRLIZ printout, the starting record number and the end record number of the compressed data entered by the user must be offset.

ESGDI operates on an existing example space file to produce a new example space file in which all utterances beginning with the vocabulary item specified are omitted.

GZEC finds the set of transition letter sets for a specified vocabulary item, using GENRLIZ.

RESCUE retrieves a desired set of transition letter sets from a temporary file and writes it into a machine file. Since the final set of transition letter sets for a vocabulary item may not be the best one due to the inclusion of bad examples, all the unique transition letter sets and the information to access them are saved in temporary files. RESCUE prints the desired set of transition letter sets if requested. RESCUE deletes the temporary files if requested.

LOOPER finds the loop letter sets for a particular vocabulary item. It provides a printout which shows the sets of possible loop letter sets for each example.

GASP prints transition letter sets or merged transition and loop letter sets for each specified machine.

DCARDZ automatically generates the CMAC** "card image" files used by MACFOR to produce the formatted machines. The CMAC** files can be edited using the text editor if any new special processing or special counter accumulation is called for.

MACFOR is the machine formatter. For each specified machine number, MACFOR reads the card images describing the machine, extracts the specified transition letter set or loop letter set from the appropriate disk files, and formats the machine to a disk file.

REMEX is the research machine exerciser. It was designed to serve one of many functions depending upon the particular subroutines loaded with it. The version implemented for this phase of the project is the counter data extraction version. It can operate using any number of machines. For each utterance, it finds all machines which go to recognition and saves the counter data collected; that is, the number of letters which occurred in each transition and loop letter set for every machine which went to recognition. In this version, two loop letter violations are allowable before a machine is considered to fail.

RXDIT is the counter data file editor. It creates from the counter data file for a speaker (<SUB>: CDAT, RX) counter data files for each of the machine numbers (<SUB>: CDAT** where ** is the machine number). The user can specify the counter data records to be kept in the files to be created with the file <SUB>:RXCARDS. The counter data records to be kept are assumed to be in ascending order.

NAVTRAEQUIPCEN 74-C-0048-2

DISTRIBUTION LIST

Naval Training Equipment Center
Orlando, FL 32813

Defense Documentation Center
Cameron Station
Alexandria, VA 22314

All other addressees receive one copy of report.

Seville Research Corp.
Suite 400 Plaza Bldg
Pace Blvd at Fairfield
Pensacola, FL 32505

USAHEL/AVSCOM
Dir, RD&E
ATTN: DRXHE-AV (Dr. Hofmann)
P. O. Box 209
St Louis, MO 63166

Commandant
USA Field Artillery School
Target Acquisition Dept
ATTN: Eugene C. Rogers
Ft Sill, OK 73503

Human Resources Research Organization
Division No. 1, Systems Operation
300 N Washington St.
Alexandria, VA 22314

Chief of Naval Research
Psychological Sciences
Code 450, Dept of Navy
Arlington, VA 22217

Chief of Naval Operations
OP-991B, Dept of Navy
ATTN: M. K. Malehorn
Washington, DC 20350

Chief of Naval Operations
OP-987P10, Dept of Navy
ATTN: Dr. R. G. Smith
Washington, DC 20350

Chief of Naval Operations
Dept of Navy (OP-506H1)
ATTN: CAPT H. J. Connery
Washington, DC 20350

38 Library
Navy Personnel Research and Development Center
12 San Diego, CA 92152

ERIC Clearinghouse on Educational Media (Technical)
Stanford University
Stanford, CA 94305

Grumman Aerospace Corp
Plant 36
ATTN: Mr. Sam Campbell
Bethpage, LI, NY 11714

American Psychology Association
Psychology Abstracts
Executive Editor
1200 17th St. NW
Washington, DC 20036

Director
Defense Research and Engineering
ATTN: LCOL H. Taylor, OAD E&LS
Washington, DC 20301

Commander
Naval Air Systems Command Headquarters
Code 340F
Washington, DC 20361

Commanding Officer
Naval Technical Training
ATTN: Code 016, NAS Memphis
Millington, TN 38054

US Air Force Human Resources Lab
AFHRL/OR Occupational Manpower Relations Div
Lackland AFB, TX 78235

US Air Force Human Resources Lab/DOJZ
Brooks AFB, TX 78235

AFHRL/FTO
ATTN: Mr. R. E. Coward
Luke AFB, AZ 85309

ASD SD24E
ATTN: Mr. Harold Kottmann
Wright-Patterson AFB, OH 45433

Commander
Navy Air Force, US Pacific Fleet
NAS North Island (Code 316)
San Diego, CA 92135

Chief
ARI Field Unit
P. O. Box 476
Fort Rucker, AL 36362

Chief
Naval Education & Training Liaison Office
AF Human Resources Laboratory
Flying Training Div
Williams AFB, AZ 85224

Naval Weapons Center
Code 3143
ATTN: Mr. George Healey
China Lake, CA 93555

TAWC/TN
Eglin AFB, FL 32542

Chief Naval Research
Code 437, Dept of Navy
Arlington, VA 22217

Scientific Advisor
HQ US Marine Corps
Washington, DC 20380

Calspan Corp.
Librarian
PO Box 235
Buffalo, NY 14221

Dr. Jesse Orlansky
Institute for Defense Analyses
Science & Technology Division
400 Army-Navy Drive
Arlington, VA 22202

National Defense College of Canadian Army
Staff College
Ft Frontena
Kingston, Ontario, Canada

Federal Aviation Agency
NAFEC Library, Bldg 3
Systems Research and Development Service
Atlantic City, NJ 08405

Commanding Officer
Rome Air Development Center
Technical Library
Griffiss AFB, NY 13440

Commander
Naval Air Test Center (CT 176)
Patuxent River, MD 20670

Commanding Officer
Naval Air Technical Training Center
NAS, Memphis 85
Millington, TN 38054

Commanding Officer
Human Resources Laboratory
Flying Training Division
Williams AFB, AZ 86046

Commanding Officer
FASOTRAGRULANT
Naval Air Station
Norfolk, VA 23511

AD-A049 680 LOGICON INC SAN DIEGO CALIF F/G 5/8
USE OF COMPUTER SPEECH UNDERSTANDING IN TRAINING: A PRELIMINARY--ETC(U)
JUN 77 J E PORTER, M W GRADY, M B HICKLIN N61339-74-C-0048
UNCLASSIFIED NAVTRAEQUIPC-74-C-0048-2 NL

1 OF 2
AD
A049 680

Technical Report NAVTRAEQUIPCEN 74-C-0048-2

USE OF COMPUTER SPEECH UNDERSTANDING IN TRAINING

Logicon, Inc.
P. O. Box 80158
San Diego, California 92138

FINAL REPORT MARCH 1977 - JUNE 1977

June 1977

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NAVTRAEQUIPCEN 74-C-0048-2	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): USE OF COMPUTER SPEECH UNDERSTANDING IN TRAINING: A Preliminary Investigation of a Limited Continuous Speech Recognition Capability.	5. TYPE OF REPORT & PERIOD COVERED: Final Report. March [illegible] - June 1977
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): J.E. Porter, M.W. Grady, M.B. Hicklin, L.F. Lowe	8. CONTRACT OR GRANT NUMBER(s): N61339-74-C-0048
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Logicon, Inc. P.O. Box 80158 San Diego, California 92138	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 3753-2P2
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Training Equipment Center Code N215 Orlando, Florida 32813	12. REPORT DATE: June 1977 13. NUMBER OF PAGES: 134
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*: 135p.	15. SECURITY CLASS. *(of this report)*: Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Speech Recognition
Speech Understanding
Continuous Speech Recognition
Controller Training System
Mathematical Machines
Real-time Speech Recognition

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes the requirements for connected word speech recognition in training systems design. A review of the published literature and commercially available systems demonstrates that no existing capability exists which fully satisfies the unique features of speech recognition in training devices. A new approach toward limited continuous speech recognition was investigated based on examining the speech data itself to find characteristic sound structures; the order in which these sounds occur; and the time duration of, and between, these characteristic sounds.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

next page

406 012

LB

20. ABSTRACT(Cont)

Algorithms were developed and exercised over speech data generated by a commercial preprocessor for a large number of utterances spoken by a single speaker. Characteristic structure was found for the 10 digits and the word "point." Borrowing from the mathematical theory of formal languages, these "sounds" are termed transition letter sets. The residual data were formed into loop letter sets which are used to reduce false recognitions. Nondeterministic finite transducers, defined by the transition and loop letter sets for each vocabulary item, were exercised over entire utterances to validate the basic concepts and extract time duration data.

While the complete implementation and testing of this mathematical machine approach toward limited continuous speech recognition remains incomplete, the preliminary results have verified the basic assumptions and provided the encouragement to proceed with the development of the technique.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont)

TABLE OF CONTENTS (Cont)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont)

LIST OF TABLES

LIST OF TABLES (Cont)

SECTION I

INTRODUCTION

This report documents the findings and work accomplished in the initial phase of a project whose ultimate goal is to obtain a capability for Limited Continuous Speech Recognition (LCSR) for the Naval Training Equipment Center (NAVTRAEQUIPCEN).

Background

LCSR is generally understood in the speech research community to mean the problem of automatically recognizing natural human speech consisting of isolated utterances which are sequences of words chosen from a small (less than 30-word) vocabulary spoken continuously; i.e., without pauses or breaks between words. Reliable automatic recognition of isolated words is now a fait accompli, but the techniques used in isolated word recognition have proven to be ineffective when applied to continuous speech. Since continuous speech is the more natural mode of human verbal communication, the scientific and industrial communities are vigorously pursuing extensions of earlier successes in isolated word recognition. However, the general problem of automatically recognizing continuous speech has shown itself to be so difficult as to warrant restriction to a limited problem, to sharpen the focus of the difficulties of reliability recognizing even relatively few words in relatively short utterances.

NAVTRAEQUIPCEN has demonstrated the gains to be made in training effectiveness by incorporating an automatic speech recognition capability in training systems. Isolated word recognition is often sufficient to meet the requirements of these systems. But it also often occurs that an automated

training system either requires, or can be made much more effective by, introducing an LCSR capability. NAVTRAEQUIPCEN therefore supported this project to investigate the possibilities for obtaining an LCSR capability in support of training system development.

The initial goals of the LCSR project were:

a. To identify the unique features of the LCSR problem as it arises in training applications, in order to have the sharpest possible definition of the problem of interest to NAVTRAEQUIPCEN, and to guide the remainder of the project.

b. To survey the relevant technical literature and industry to determine the state of the LCSR art, and to determine if there exist systems or techniques which satisfy the requirements of the training LCSR problem.

c. On the basis of an understanding of the training LCSR problem and the current state of the LCSR art, to assess the feasibility of obtaining an LCSR capability for support of training system development.

d. If obtaining a training LCSR capability is feasible, to develop a plan for doing so.

On the basis of early work in pursuit of these goals, it was determined that developing a training LCSR capability, while not without risk, was a worthwhile venture. The LCSR project goals were therefore extended and modified, as will be discussed in section V.

Contents of this Report

Section II addresses the unique features of the LCSR problem, as it arises in training applications. Section III discusses the results of a survey of the technical speech recognition literature. The results of that review revealed that there are no reported systems or techniques which are immediately applicable to NAVTRAEQUIPCEN's LCSR requirements. Section IV introduces a novel technical approach to the LCSR problem, called the

"mathematical machine approach." On the basis of the detailing of training LCSR requirements and the literature survey, this mathematical machine approach to LCSR was deemed worthy of further investigation, and section V presents an envisioned three-phase plan of attack. The remainder of this report, sections VI through XII, describes the word done and results obtained during the first phase of a study of this mathematical machine approach. These results are preliminary in that no speech recognition capability has been produced or tested (that work will be accomplished in the next phase). But the approach is investigative, and thus even the preliminary results are interesting in their own right.

The appendices contain supporting material. References precede the appendices. Appendices A, B, and C contain LCSR project technical memoranda which expand upon certain topics discussed in this report. A large proportion of the present effort has been expended in producing and exercising a large array of computer programs for manipulating and analyzing speech data. Appendix D lists and describes very briefly the programs which were developed. Appendix E contains a brief description of an earlier approach to LCSR which has been abandoned.

SECTION II

THE REQUIREMENT FOR RECOGNITION OF CONTINUOUS SPEECH IN SUPPORT OF TRAINING

The Source of the Requirement

Automatic speech recognition has been shown to offer opportunities for significantly improving the efficiency and effectiveness of training systems. Systems developed in cooperation with NAVTRAEQUIPCEN which demonstrate this practical benefit of speech recognition in training systems include the Ground Controlled Approach Controller Training System (GCA -- CTS) and the Automated Adaptive Flight Training System (AFTS). On the basis of experience gained in these systems, it is clearly desirable and appropriate to expand the use of automatic speech recognition in training systems.

Many training applications can be supported adequately by a capability to recognize isolated words or word groups automatically. The applications mentioned above are of this type. However, in some applications isolated word recognition is not adequate. An automated training system for training air intercept controllers, for example, requires recognition of numerical data, naturally spoken as an unbroken sequence of digits. In this and similar applications, the number of digit sequences of interest precludes the use of isolated word recognition algorithms via the artifice of treating each possible sequence as a potential explanation of an utterance.

Spoken strings of digits are a special, but frequently occurring, case of a more general phenomenon. It often occurs that a set of phrases to be recognized in a training application consists of sequences of words or word groups taken from a much smaller lexicon, but spoken continuously. For

example, the GCA–CTS lexicon associated with glidepath commands can be regarded as various combinations of only 10 words, spoken continuously. If recognition could be accomplished in terms of the smaller, underlying lexicon, the total recognition potential associated with a lexicon of a fixed size would be greatly increased. Viewed in this way, one requirement to recognize continuous speech is revealed to be the need to recognize a large number of phrases constructed from a relatively small lexicon, but spoken without pauses.

Another, closely related, requirement for treating continuous speech arises in training situations which allow the trainee considerable latitude in his verbal behavior. It often occurs that the allowed vocabulary is very large, but the task situation is such that a reasonably small number of key words or phrases can be identified. If these key words or phrases can be recognized when embedded in extraneous verbal material, the training objectives can be met. This "word spotting" problem, while quite different from the problem of recognizing utterances taken exclusively from a fixed and relevant lexicon, shares many basic technical difficulties with the latter problem. The LCSR project is specifically directed toward recognizing continuous speech taken from a small, fixed lexicon, but it is worthwhile to note that many techniques developed in this context can be expected to be applicable to the word-spotting problem also.

Unique Features of the Training LCSR Problem

Experience in applying automatic speech recognition to practical training systems has revealed several special characteristics of the LCSR problems which arise in this class of systems. These special characteristics make the training LCSR problem much more specific than what is generally referred to as the "limited continuous recognition problem" in the technical literature. Several features which localize the training LCSR problem

within the larger domain reported in the literature are discussed below. While not all of these characteristics are universally shared by all LCSR problems arising in training applications, it is true that any solution to the LCSR problem compatible with these characteristics would meet the requirements in most training applications.

a. A small vocabulary is involved. Many training problems entail vocabularies of 20 words or less, and often recognizing fewer words would be a useful capability. The 10 digits in combination with a few control words is a fairly representative and common case. Using a mixed strategy of isolated and continuous speech recognition techniques can sometimes reduce the required vocabulary size of the continuous part of the problem even further.

b. The vocabulary is fixed. Within a given training application, the vocabulary changes with a half-life measured in months or years. As a result, rapid accommodation of vocabulary changes, while attractive, is not an important requirement. Techniques which entail detailed (and perhaps time-consuming) off-line analysis of the vocabulary items are therefore at no particular disadvantage.

c. Semantic, syntactic, and other higher knowledge sources are often nearly or completely irrelevant. This observation is typified by the numerical data entry problem, where strings of digits must be recognized and there are essentially no hard data available in the remainder of the system which can be used to predict what the spoken digit string might be. In many cases, a priori probabilities can be assigned to gross features of the utterance, such as to the number of digits in the utterance, or to the identity of the first digit. But it often occurs that within the utterance (i.e., for non-initial words) the branching factor is essentially equal to the size of the vocabulary. The fact that a training system has specifically to deal with errors committed by the trainee exacerbates the problem, as deviations from proper syntax (for example) may be both more likely to occur and more interesting in themselves in the training environment than in the operational environment.

d. Real-time operation is necessary. Effective training often requires very quick response to trainee vocalization, either to preserve realism of a simulated environment or to minimize the latency between response and reinforcement. A time lag of less than two seconds between completion of an utterance and recognition is required.

e. Recognition accuracy must be high. Trainee motivation (and thus training effectiveness) drops precipitously with any decrease in a training system's reliability, and recognition failures are perceived as just another variety of system failure by the system user. The supposition that low recognition accuracy can be tolerated in training systems is often supported by the argument that the purpose of the system is to teach correct verbal behavior; and hence, the careful enunciation required for good recognition can be demanded of the trainee. This argument is fallacious for two reasons: (1) few training systems have precise enunciation as an important training objective and (2), at the present state of the art, recognition accuracy in the high 90-percent region is only attainable with audio input which is very understandable to the human ear – sloppy enunciation significantly degrades the already less-than-perfect recognition accuracy currently attainable.

f. Speaker independence, while convenient, is not a necessity. Training systems which warrant a dedicated speech recognition capability tend to be associated with tasks which require several hours or more of training. A small amount of time spent adapting the system to the trainee's voice is rarely a significant drawback, particularly since this adaptation period can sometimes be treated as part of the training experience wherein the trainee learns the vocabulary or how to operate the training system.

g. The computational requirements should be compatible with central processors on the scale of minicomputers or even smaller systems. This is simply an empirical observation on the economics of training systems. The computers used in training systems tend to be dedicated, and the training systems tend to be of such a scale and have development budgets which can accommodate the cost of mini- or micro-computers, but often not the cost of a large main frame. Counterexamples can undoubtedly be found, but experience indicates they are the exception rather than the rule. This same observation applies to special-purpose hardware which supports the front-end analysis of the analog speech signal. Sophisticated, special-purpose preprocessing hardware can become very expensive and hence it is desirable to utilize established, commercially available components if possible.

SECTION III

LITERATURE SURVEY RESULTS

Purpose

A survey of the technical literature regarding recognition of continuous speech was undertaken as a preliminary task of the LCSR effort. The motivating reasons for this survey were:

a. To determine if there exist in the literature techniques for performing LCSR which are immediately usable, with minimal or no alteration, in applications of interest to NAVTRAEQUIPCEN.

b. (Assuming that such techniques do not exist) to obtain insight into approaches to the LCSR problem and thereby derive guidance to promising areas of investigation.

While searching the literature for recognition techniques which might be immediately applicable to NAVTRAEQUIPCEN training problems, it was necessary to carefully consider the unique features of the training LCSR problem discussed in the previous section. For example, many of the reported techniques cannot be used in real time partly because of the heavy computational burden imposed by incorporating syntactic and semantic information. In evaluating these techniques, it is necessary to disregard the irrelevant parts of the techniques and consider only those portions which are necessary to solve the more limited problem.

Deriving guidance for the current effort from the literature reduced to discerning general characteristics of those techniques which have reported greatest success, and relating those general characteristics to an approach being concurrently developed (the mathematical machine approach described later). The latter part inevitably takes on the character of rationalization,

but it is nevertheless useful to understand the relationship between the newly conceived approach and others currently under investigation.

Material Surveyed

The list of material surveyed (included in references) includes only those articles, books, conference records, etc., which were thoroughly read and understood. Many others were scanned to determine their usefulness for the purpose of this survey, and rejected as inappropriate for further study.

Several collections of individual articles and survey articles were particularly useful in gaining an overview of the state of continuous speech recognition and its advancement during the present decade. Hyde's survey of 1967, Hill's article of 1971, papers from the 1974 Institute of Electrical and Electronic Engineers (IEEE) Symposium (edited by Reddy), the special issue of the IEEE Proceedings for April 1976, and the Conference Record of the 1976 IEEE International Conference on Acoustics, Speech and Signal Processing contain about enough material to justify the conclusions reached by surveying the larger body of material. The most important information missing from that collection is contained in Martin's dissertation (1970).

Results

The primary conclusions which result from the literature survey are:

a. No technique for performing LCSR which is readily adaptable to applications of interest at NAVTRAEQUIPCEN has been reported in the reasonably accessible technical literature. This conclusion merits high confidence with regard to publications prior to April 1976 and moderate confidence with regard to publications prior to April 1977.

b. A comparison of the general characteristics of the more promising techniques suggests that the adopted approach to LCSR (described in section IV) is a good match to NAVTRAEQUIPCEN's requirements and has reasonable potential for success.

These conclusions are discussed separately in the following paragraphs.

Existing Techniques – The resources of this project have permitted surveying only a small fraction of the technical literature pertinent to the LCSR problem. Fortunately, there is enough active speech research going on to guarantee a vigorous interchange of information among researchers around the world. As a result, it can safely be inferred that, since no wholly adequate technique for LCSR was reported or referenced at the 1976 IEEE International Conference on Acoustics, Speech and Signal Processing, no such technique had been reported before that time.

Thoroughly reported techniques for recognition of continuous speech have failed to meet the requirements of the LCSR problem either by not giving adequate recognition accuracy or by not being feasible for real-time implementation. Unfortunately, there is so little interest or success in real-time operations of recognition systems that no mention is made of the speed of operation for many otherwise thoroughly reported techniques. However, it is possible to infer something about speed of operation for most of these systems by comparison with other, more or less similar, systems whose speed of operation has been reported.

There are two sources of the large computational requirements which cause most recognition techniques to be inappropriate for real-time operation. One is performing preprocessing in the central processor, and the other is use of a hypothesize-and-test recognition algorithm. The types of preprocessing encountered in the literature include extraction of zero-crossing statistics, weighted filtering of the entire spectrum, spectral analysis (e.g., by fast Fourier transform), autocorrelation analysis, and linear prediction. All require processing data representing the acoustic signal sampled at a rate of 10 to 20 kiloHertz. Calculations such as these can only be done in real time by specialized computational hardware such as array processors.

Hypothesize-and-test recognition algorithms have one element which repeatedly proposes hypothetical explanations of the spoken utterance, and a second element which compares the hypothesized explanation with the observed speech features. These techniques run into trouble in real-time operation because the number of hypotheses which have to be tested is enormous, even when only the most plausible hypotheses are considered. Recognition algorithms of the hypothesize-and-test type are currently the most popular, with sequential decoding a close and gaining second.

Although it is difficult to compare different recognition techniques objectively in the absence of test data taken under identical test conditions, it is interesting to attempt to identify those reported techniques which "come closest" to meeting the NAVTRAEQUIPCEN requirement for an LCSR system. Two systems are mentioned below. One has demonstrated recognition accuracy equal or near the best obtained to date but is not promising for real-time application, and the other operates in real time but gives less than adequate recognition accuracy.

The HARPY System – The HARPY system was developed by Lowerre at Carnegie-Mellon University. It is a developmental outgrowth of two earlier systems produced under the direction of Reddy. It has demonstrated a recognition accuracy (on words) of 93 to 98 percent with an effective vocabulary of about 11 words. (The actual vocabulary contained 39 words but was constrained by a rigid syntax with a branching factor of 10.8). This performance was based on training and testing with the same set of four speakers. Recognition accuracy was 98 percent when training and test data were taken on the same occasion, and 93 percent when the training and testing data were separated in time by five months. The average number of words per utterance in the test data was 5.5, giving error-free recognition of complete utterances 71 percent and 89 percent of the time for the two test conditions. The HARPY system shows both higher recognition accuracy than its predecessor, the HEARSAY 1 system, and about an order of magnitude faster operation than

HEARSAY 1. Unfortunately, this is still inadequate for most training applications, as about 4.5 million computer instructions must be executed for each second of speech to be recognized. On the PDP KA-10 computer, for which HARPY was designed, operation takes about 13 times real time.

The source of HARPY's heavy computational burden is both preprocessing in the central procesor and a carefully optimized hypothesize-and-test recognition algorithm. About 64 percent of the computation time required is for preprocessing (generating autocorrelation and linear prediction coefficients). Relegating the preprocessing to a special-purpose processor might therefore result in operation at about five times real time.

LOCUST is an evolution of the HEARSAY (1972)–DRAGON (1974)– HARPY (1976) systems developed at Carnegie-Mellon University. Whereas the earlier systems ran on a PDP-10 at many times real time, LOCUST will run on a PDP-11 at four to five times real time. According to Lowerre, this system is currently operational only for "very small grammars," and they are continuing its development. No details are yet available (in the public domain) on LOCUST.

In his dissertation published April 1976, Lowerre claimed HARPY was both faster and more accurate than any other connected speech recognition system in operation at that time.

Martin's System – Martin, in his dissertation of 1970, describes a system for recognizing connected strings of digits using the preprocessing technique which is now implemented (with improvements) in Threshold Technology's commercially available speech recognition hardware (e.g., the VIP-100) used for isolated word recognition. Although the system he describes used an analog recognition technique following the (essentially) analog preprocessor, it could easily be duplicated in a digital processor, such as Data General Corporation's Nova series of minicomputers, in real time. After very thorough testing of his system (on 32,500 connected digits spoken

by 105 talkers chosen to represent all dialects of American English, recognition accuracy was found to be 88 percent on a word basis.

Martin's system is interesting in several different respects, including:

a. The recognition accuracy achieved is remarkable, especially for the time period in which it was produced.

b. It is a speaker-independent system.

c. No other system has been more thoroughly tested and reported in the technical literature.

It is interesting to note that, when specialized for 10 talkers of a single dialect (Boston), Martin's system achieved recognition accuracy of more than 94 percent, approaching the accuracy of the HARPY system but with more than twice as many talkers. However, this result is not considered representative of the accuracy which could be obtained in military training applications, as many different dialects will have to be accommodated in that environment, and it is not clear how Martin's system could be trained for individual talkers.

In view of Martin's early success, a question arises as to why his approach was not developed further. The approach described in his dissertation was limited (in its analog implementation and for continuous speech) by its sensitivity to misrecognition of individual features. One can hypothesize that other researchers who were working in the more flexible environment of digital computers did not adopt Martin's approach because it was not clear how to incorporate syntactic or semantic knowledge sources. The consensus at that time (at least within the ARPA program) was that the acoustic signal did not have enough information in it for successful recognition, and the real problem was adding the higher knowledge sources.

Commercial Endeavors – The principal commercial speech recognition companies were also contacted to determine their plans vis-a-vis a continuous word recognition capability. Companies contacted were:

a. Threshold Technology, Inc.

b. Scope Electronics, Inc.

c. Dialog Systems, Inc.

This review confirmed that no commercial (off-the-shelf) continuous recognition system is currently available, nor do the major companies plan to announce such a system in the near future. However, those contacted did consider the limited continuous speech recognition problem solvable.

Relevance to the LCSR Project Effort

The literature reveals some trends in continuous speech recognition which can be interpreted as augering well for the line of inquiry described in later sections. Some of these trends are discussed below.

There is a trend toward de-emphasis of segmentation into classical phonemes and specific phoneme recognition. Earlier efforts focused on recognizing speech phoneme-by-phoneme, with articles appearing on the difficulties of recognizing particular phonemes. The tendency now is to treat the preprocessor more nearly as a sound classifier, and to ignore preconceived notions of what the speech data received from the preprocessor are like. The reason for the tendency is that reliable segmentation into phonemes turned out to be impossible, dashing the early hopes that the internal reference representations of words could be some simple variation of familiar phonetic spellings, modified by phonological rules.

It follows from the failure of ridgidly phoneme-oriented recognition that there is a tendency to go to the speech data (that is, develop algorithms for

processing real speech data) to determine its recognizable chara teristics. This is in contrast to the early reliance on the "obvious" phonemic content of words to be recognized. The recognition techniques being developed now therefore tend to have two parts: the recognition technique per se, and techniques for deriving relevant parameters (such as Markov transition probabilities or likelihood-measure thresholds) from large samples of speech. This trend marks the demise of the early influence of linguists and phoneticians on speech recognition research.

There is also a recent trend toward sequential decoding of the speech signal instead of exhaustive hypothesize-and-test recognition methods. The distinction between these two approaches becomes blurred as the methods for optimizing the search of the test space become more and more efficient. Interestingly, both HARPY and Martin's approach are essentially sequential in nature. Both use a transition state model to determine a limited set of next-possible features. In the case of HARPY, this was a considerable simplification over its predecessor's models, which entailed probabilities of transitions to each of a large set of possible next states.

The approach adopted for the LCSR effort reported herein and described in the following sections conforms to each of the trends mentioned above; namely, toward:

a. Treating the preprocessor as a sound classifier.

b. Emphasizing the derivation of the recognizable speech characteristics from real speech data.

c. Sequential decoding.

SECTION IV

THE MATHEMATICAL MACHINE APPROACH TO LCSR

Rationale

Among the more interesting dimensions along which human speech displays its remarkable variability is the distribution of information among the acoustic speech signal, syntactical structure, nonverbal cues, and other more or less subtle communications channels connecting a speaker and his listeners. It is generally agreed that most natural verbal communication places a significant amount of its information content in the nonacoustical channels, and that the acoustic signal often does not even carry enough information to identify reliably the component words of an utterance constructed over a small and known vocabulary. Verbal communication of strings of digits must not be of that character, however, as completely unpredictable strings can be reliably communicated over a controlled acoustic channel, such as the telephone. In fact, most verbal communication of numerical data has a very high degree of uncertainty associated with it. It is very reasonable to infer, therefore, that the acoustic signal corresponding to a spoken digit string contains the information needed to recognize its component digits.

Simple audio tape splicing experiments quickly verify two further facts about the acoustic signal conveying digit strings. First, the information conveying an individual digit of a string is localized (one can easily construct a tape for "67" out of tapes for "463" and "978"), and, second, human perception of the individual digits is based on the sequential occurrence of individually identifiable sounds, the timing of those sounds, and exclusion of some intervening sounds. (One can make a recognizable "7" by combining the necessary sounds snipped from other, unrelated, words.)

It is reasonable to assume, on the basis of these intuitively obvious facts, that individual words which occur in digit strings can be characterized and recognized on the basis of:

a. sounds, or small groups of sounds, which reliably occur

b. in a fixed order

c. with characteristic time durations and time intervals between these groups.

A Proposed Technique

If these assumptions are correct, and if the characterizing sounds or sound groups occur with sufficient reliability, it should be possible to recognize words within digit strings in a simple sequential manner, noting the occurrence of a sound of the first group, then listening for a sound of the second group, etc. until all of the required types of sounds have been detected. During this process, the time durations of the component sounds can be noted. Also, the occurrence of any spurious sounds can be used to reject a potential but false detection. When all the required sounds have been detected, without rejection due to spurious sounds, the time durations of the component sounds and intervals can be compared statistically to reference data derived from observations of previously recorded speech. A potential recognition can then be accepted or rejected on the basis of the similarity of the observed time durations to the reference data. The primary appeal of such a simple recognition procedure is that it could easily be performed in real time. The approach investigated and described in this report is an extension and modification, without jeopardizing the potential for real-time implementation, of this simple recognition method.

Many variations and refinements to this simple sequential procedure can be formulated. As formulated above, the recognition procedure can be modeled as a simple finite-state mathematical machine (a finite automaton). Adding features to accommodate the less-than-perfectly reliable occurrence of the characteristic sound groups and problems with starting the recognition

process at the right time result in a more complex process, but still one conveniently modeled as a mathematical machine. The more complicated version is a nondeterministic finite transducer. The utility of concepts from the theory of mathematical machines for describing the class of recognition algorithms considered in this project has led to calling it "the mathematical machine approach."

An Apparent Paradox

Linguists and phoneticians noted long ago that words are made up of special sequences of sounds with characteristic durations. The characteristic sounds have been studied in detail, leading to the concepts of phone, morpheme, allophone, phoneme, etc. Armed with this understanding, the earliest researchers in speech understanding knew that a mathematical machine approach to automatic speech recognition would work, and that the characteristic sound groups needed to make it work were simply phonemes or something very close to phonemes. However, after several decades of research, this approach has not led to a reliable automatic LCSR capability.

An apparent paradox therefore exists between the fact that phoneme-directed speech recognition has failed to solve the problem of LCSR and the assumed fact that words are characterized by sound groups, order, and time. (A more practical point of view might be that the real paradox is why anyone should elect to pursue an approach which has not produced results in several decades, but the resolution of either version of the paradox turns out to be the same.)

The Paradox Resolved

Failure of the phoneme-oriented recognition approaches to produce a viable LCSR capability may be ascribed to the difficulties of producing a preprocessor or analog transducer for extracting speech features corresponding to phonemes. This may be due to an inadequate understanding of

the human aural perception mechanism; but, whatever the reason the fact is that individual phonemes cannot easily and reliably be detected even in clearly spoken strings of digits (wherein we have reason to believe the acoustic signal carries sufficient information to support word recognition). Apparently the preprocessors used in speech recognition are sensing the acoustic signal differently than humans, resulting in an inability to perceive (in the psychological sense).

The Deterministic Preprocessor

Raising the question of the correspondence between sounds and the output available from a given preprocessor points up the need to refine slightly an argument given earlier. Let us define a deterministic preprocessor as one which reliably and synchronously produces a given output for a given input acoustic signal, but allows the possibility that different input signals may produce the same output and that different input signals which are perceptually indistinguishable to a human may produce different output signals. Then the tape-splicing experiments referred to earlier can be used as a basis to argue that, if the output of a deterministic preprocessor carries enough information to support reliable recognition of continuous speech, that information must be present in the preprocessor output in such a way that individual words are represented by:

a. characteristic classes of output, which occur

b. in a fixed order

c. with characteristic time durations and time intervals between characteristic output samples.

The failure of phoneme-oriented sequential recognition techniques suggests that if there were such characteristic output classes present in the data produced by the preprocessors used in those systems they did not correspond to phonemes. Is it possible that the preprocessors available today do have a rich enough output structure to support LCSR, even though the output classes do not correspond exactly to phonemes? The fact that isolated

word recognition techniques (which typically do not entail explicit recognition of any subword structures) work well suggests that the answer is "yes." Until the data from a particular preprocessor are examined to determine the absence or presence of characteristic output classes, without preconceived notions as to what those classes are (or correspond to in terms of a human percept), the potential of that preprocessor for supporting LCSR remains unknown, and the usefulness of a relatively simple sequential recognition technique also remains unknown.

The apparent paradox is thus resolved: the difficulties of phoneme-oriented recognition are related to the difficulties of producing a phoneme-extracting preprocessor, and have little relevance to the potential success of sequential recognition procedures. Preprocessors successfully used in isolated word recognition probably exhibit some reliable output classes unrelated to phonemes but which can support LCSR. It is interesting and worthwhile to try to discover what the appropriate classes of output are. Interesting because if they can be shown not to be present, there is little hope that the preprocessor can support LCSR. And worthwhile because if the appropriate output classes can be discovered they can be used as the basis of a sequential, mathematical machine type of recognition procedure.

The Approach Characterized

The unique feature of the mathematical machine approach to LCSR is that it attempts to discover characteristic classes of preprocessor output which are associated with each vocabulary item, and explicitly uses those classes, together with their durations and intervals, in a relatively simple recognition procedure. The method of discovering the characteristic output classes is direct automated examination of speech data. It is a data-intensive approach, and has the peculiar advantage that, even if it doesn't lead to a successful LCSR technique, the nature of the chosen preprocessor (and the data it produces) would at least be quite well understood.

Preceding arguments have shown that if these characteristi output classes, order, and time durations are not present in the preprocessor output when digit strings are spoken, in reliable enough form to support reliable recognition of the component words, then the preprocessor is guilty of dropping too much information while transforming its acoustical signal input. One can almost imagine a critical experimental procedure whereby the labeled output from a given preprocessor is subjected to tests revealing the presence (or absence) of characterizing output classes and their properties. The maximum recognition performance obtainable on the basis of these data could be computed and the end result would be a statement, based on fact, about how well that preprocessor can support LCSR. It would be a significant breakthrough in speech recognition research to be able to separate the issues of preprocessor and recognition algorithm performance in this way. There is, unfortunately, little evidence (in the literature) of interest in devising or performing critical test procedures of this kind, probably owing to their difficulty. The mathematical machine approach is a modest effort to approximate such a critical test.

Note that the approach is preprocessor specific; that is, it attempts to answer questions about, and produce an LCSR procedure for, a specific preprocessor. And while the methods and techniques employed, for the most part, can be applied to any feature extraction-like preprocessor, it is not clear that results obtained for one preprocessor would be useful for any other preprocessor.

Critical Questions

As a scientific investigation, the mathematical machine approach to LCSR attempts to answer the following questions. If the answers can be found and are felicitous, an LCSR capability will result.

a. How can classes of preprocessor output suitable for characterizing each vocabulary item be found in and extracted from samples of speech data?

b. How reliably do these characteristic output classes occur in speech samples?

c. How unique are the characteristic classes for each vocabulary item? (How bad is the false recognition problem?)

d. How may the output observed between characteristic samples be described and used to eliminate false recognitions?

e. What are appropriate descriptions of the durations of, and intervals between, the characteristic samples?

f. How can these temporal data be used to reduce false recognitions?

g. In view of the observed characteristics of the preprocessor output for individual vocabulary items, what are appropriate mathematical machine structures for spotting each word?

h. How well do the individual word spotting procedures work (disregarding interaction among the words of an utterance)?

i. How can individual word recognizers be combined, and their observed interactions within an utterance be used, to produce a more effective composite recognition procedure?

j. How effective is the resulting LCSR technique?

As described in section V, the current phase of this project provides only preliminary answers to some of these questions. That section also shows the relation of the current phase to subsequent phases.

Applicability of the Mathematical Machine Approach to Training

The mathematical machine approach to LCSR has intentionally been formulated to match the unique features of the training LCSR problem discussed in section II. The rather simple sequential recognition procedures of the mathematical machine approach have unusual potential for implementation in real time and, as noted in section II, real-time operation is an important requirement for LCSR in support of training. Most of the

recognition techniques which are the subject of research today entail extensive searches and are not amenable to real-time operation in minicomputers. The mathematical machine approach can be considered to gain real-time operating potential at the cost of extensive analysis of speech data, performed off-line. The small, fixed vocabulary of the training LCSR problem permits this trade off. The computational requirements for the mathematical machine recognition procedures are modest in terms of memory requirements as well as speed, therefore meeting another special requirement of the training LCSR problem. They are suitable for mini- or micro-computer implementation, and, although the off-line (prerecognition) computational requirement in support of the necessary analysis is large, it too is compatible with a disk-equipped minicomputer.

The Essence: The Preprocessor

The mathematical machine approach emphasizes careful scrutiny of the speech preprocessor to find exactly what information is available there, and a means to use it effectively. This is consistent with the observation that semantic, syntactic, and other nonacoustic knowledge sources often have little to contribute in the training LCSR problem, and the conviction that a viable preprocessor must provide sufficient information in its output to support recognition of digit strings.

SECTION V

SCOPE OF THE PROJECT

The LCSR Project Phases

The information reported herein represents the results of only the first phase of an envisioned three-phase effort. The goals of this current effort will be discussed in detail in the following paragraphs but, briefly, the intent has been to formulate the mathematical machine concept as applied to the LCSR problem and test many of its underlying assumptions. Based on the results of that effort, the next phase will complete the development and analysis of the approach, resulting in a demonstration of a speaker-unique LCSR capability. A final phase would investigate removing the highly speaker-dependent constraints of the earlier phases, resulting in an LCSR capability that can then be integrated into training system design.

One Speaker First – The mathematical machine approach described herein is preeminently a search for structure and information in the output of the chosen preprocessor. One expects to find more structure and more word characterizing information in speech data produced by one subject than in data from several speakers. It is therefore appropriate to use a single subject while developing techniques for discovering the structure of the speech data. The overall plan of attack for the project then becomes:

a. Develop techniques for discovering the structure of a single speaker's speech (Phase 0).

b. Evaluate the LCSR potential of the mathematical machine approach to recognizing the initial subject's speech (Phase 1).

c. Apply the same techniques to several other speakers, and re-evaluate the LCSR potential (Phase 1).

d. Compare and contrast the structure and word characterizing information found in the several speakers (Phase 2).

e. Develop and apply techniques applicable to more than one speaker (Phase 2).

Speaker Independence and the Mathematical Machine Approach – The United States Navy conducts training programs all over the world; and even at home, will train students with a variety of accents and dialects. A speech recognition program must support these various accents. While in theory a "speaker independent" recognition capability means exactly that, namely capable of recognizing each and every speaker, it is difficult to imagine that such a system can be developed which will handle the broad range of accents encountered in the Navy's training programs.

Most isolated word recognition systems currently in use utilize a "training" mode in which the system "learns" the characteristics of the speaker's voice. And indeed, the eventual development of the mathematical machine approach to LCSR may require a similar function.

An additional approach toward making a system capable of recognizing all speakers is to provide a library of reference data, and then to adapt the data to the individual characteristics of the particular speaker. Presumably this approach would demand less time and effort than training the system from scratch.

The relationship of speaker independence, speaker training, and speaker adaptation to the mathematical machine approach to LCSR is not known at this time. Are there simple characterizing classes of preprocessor output and timing characteristics which can be used to recognize the speech of several or all people? This will not be determined until the necessary speech data are gathered and examined. The mathematical machine approach requires devising automatic procedures for extracting the characterizing information from large collections of speech samples. It is conceivable that

a multi-speaker LCSR capability could be produced by exercising these same information extracting procedures on a large speech data base representative of many speakers. If so, speaker independence, or at least the basis for an adaptive system, will have been achieved at remarkably little cost. If inter-speaker variations are too great to be accommodated by a single mathematical machine structure and temporal characterization, methods will have to be found to specialize a general recognition procedure to individuals.

Goals of this Phase

As discussed in the previous section, the LCSR project will attempt to answer a variety of scientific as well as practical questions. The following goals were established for the initial, Phase 0, effort.

a. Examine the unique character of the continuous speech recognition problem vis-a-vis training system design (section II).

b. Review the state of the art to assess the applicability of current research to this more limited problem (section III) and determine the efficacy of an early segmentation approach to LCSR developed by Logicon as an in-house effort during 1975. (This approach is briefly described in appendix D. It is no longer being pursued.)

c. Investigate the mathematical machine approach to LCSR as described in an internal Logicon report, "Notes on Automatic Generation of Mathematical Machines for Recognition of Continuous Speech" published in 1976.

1. Develop procedures for extracting the characterizing structure of sound classes output from the preprocessor.

2. Determine the uniqueness of these characteristic classes across vocabulary items, and the reliability of the structure within speech data for a single speaker.

3. Examine the data output from the preprocessor between the characteristic sound classes described above, and determine how this information can be used to eliminate false recognitions.

4. Define an appropriate description of the durations of and intervals between, the characteristic samples.

These general goals have been achieved through the development of a variety of software programs and analyses of the resulting data. More specifically, then, this initial phase of the LCSR project has been concerned with:

a. Collecting a large amount of speech data for a particular speaker (section VI).

b. Generating "example spaces"; that is, dividing the speech data into collections for each vocabulary item (section VII).

c. Implementating a scheme for finding the characteristic sound groups within each example space (section VIII).

d. Implementing a scheme for finding the residual sound groups (sounds between the characterizing groups) (section IX).

e. Exercising the mathematical machines over speech data, while extracting the temporal structure within the speech data (sections X and XI).

The LCSR System Environment

Before proceeding to the discussions of these areas, the following paragraphs describe the hardware/software environment in which this work is being conducted.

The Preprocessor – An important key to the potential success of the mathematical machine approach is that the segmentation problem is entirely avoided. The acoustic transducer or preprocessor is considered to be a sound classifier, and its output at any moment is the classification of the received speech signal at that time. Any similarity of the sound classification system to a set of phonemes is largely coincidental. State transitions within the recognizing machines are conditional upon the current class of sound falling in specific sets of sound classes. There is, therefore, no

requirement to force the received sound into a limited set of classification possibilities, such as phonemes. In fact, the richer the set of sound classifications is, the more discrimination can be exercised in the recognition machines.

The necessary contribution of the system's acoustic transducer is therefore the classification of the received speech signal. The primary requirements are that the classification should be repeatable and the set of classifications should resolve the significant voice features as well as possible while keeping the information volume within workable bounds. The VIP-100 developed by Threshold Technology, Inc. satisfies these requirements. In fact, the combination of spectral and phonemic features detected by that device constitutes a very rich sound classification system. Furthermore, the considerable success obtained by NAVTRAEQUIPCEN, Logicon, and others using this device as the front end for isolated word recognition shows that it classifies voice sounds in an effective way. Logicon's success in some experiments with speaker-independent recognition is particularly indicative of the power of its nonspectral features for sound classification.

The output of the Threshold Technology preprocessor (VIP) consists of 32 binary features produced every 2 milliseconds. These features are of two types. Seventeen of them are related to the relative energy content of specific spectral bands, and the remainder result from logical and analog operations on the short-term power spectrum. Most of the latter type of features are partially successful attempts to automatically detect phonemic categories, or phoneme groups (such as unvoiced, noise-like consonants).

For this study, a combination of 15 spectral and nonspectral features listed in table 1 have been selected from the output of the preprocessor to classify the received sound. The features are binary; therefore, there are 2^{15} or 32,768 possible sound classifications. The duration of each type of sound is noted by counting the number of 2-millisecond samples during which the classification does not change.

TABLE 1. VIP-100 FEATURES USED FOR LCSR PROGRAM.

No.	Function	Description
17	MAX_D 17	Digital representation of maximum energy in filter 17 (passband centered at 5.263 KHz).
18	Λ_3	Simple vowel (but – speaker w)
19	$\Sigma 2$	Simple vowel (let)
20	$EG_1 + EG_2$	Energy gap – weak sounds
21	$N_1 + N_3$	Nasal allophones
22	0_x	Zero crossing – fricatives (3 KHz)
23	UVNLC	Unvoiced noise-like consonant
24	$w_1 + w_2$	Semivowel (we)
25	r_1	Semivowel (rate – speaker x)
26	r_2	Semivowel (rate – speaker y)
27	S	Fricative (sat)
28	3	Simple vowel (bird)
29	NSB_D 2	Broad negative slope – nasals (front vowels, low back vowels, semivowels)
30	NSB_D 3	
31	$\Lambda 1$	Simple vowel (but – speaker z)

The Computer – A distinction must be made between the computational requirements of the off-line analysis of the speech data and the requirements for the actual recognition algorithm. The initial phase of the LCSR project has concerned itself exclusively with data analysis and, as a result, has placed moderate demands on both the computer and mass memory. The project has utilized a Data General Corporation

Nova 1200 minicomputer and a Diablo dual cartridge disk system for a combined mass storage of 3-million words. A system console (CRT) and medium speed line printer also supported the program. (See figure 1.)

Because the recognition algorithm itself is deterministic, sequential, and very easily implemented in a computer, real-time operation in even smaller computers should be easily and naturally attainable with small vocabularies. The method also lends itself very naturally to parallel processing, as the individual word recognition machines are relatively autonomous. Larger vocabularies can therefore be accommodated by parallel processing, for example in microprocessors.

The Software Environment – Almost without exception, all the software developed during this project has been written in FORTRAN IV and runs under Data General Corporation's Real-Time Disk Operating System.

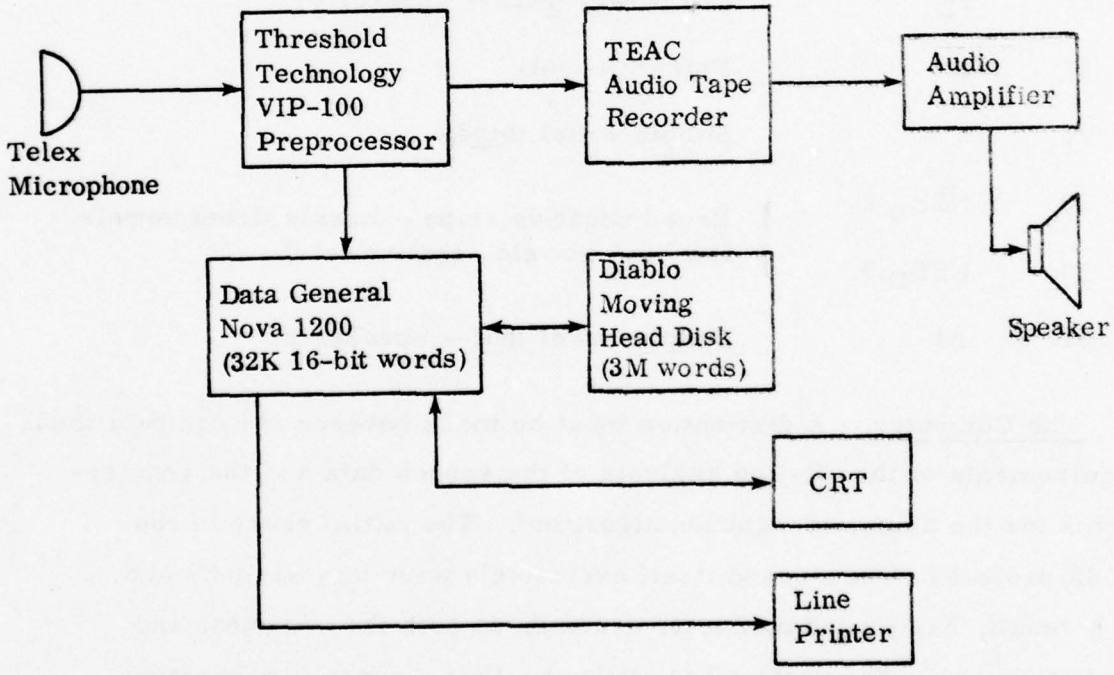


Figure 1. LCSR Equipment Configuration.

SECTION VI

THE SPEECH DATA

Data Collection Procedures

A large sampling of speech data was collected by having the chosen speaker (MWG) read prepared lists of number strings into a Telex close-speaking microphone connected to the Threshold Technology model VIP-100 speech preprocessor. Both audio and digital output were taken from the VIP-100. The audio signal was recorded by an audio tape recorder for later verification of the intelligibility of the speech signal. The digital data, produced at the rate of two 16-bit words every 2 milliseconds for the duration of each utterance, were recorded on a magnetic disk. As described in section V, only 15 of the 32 available speech features recognized by the VIP-100 are used in this phase of the LCSR project. A compressed form of the speech data was also recorded on disk by discarding the unused bits corresponding to the unused features. About 8 percent of the 15-bit data elements in the resulting data stream are single occurrences; i.e., differ from both the preceding and following 15-bit data elements. These single occurrences are randomly distributed throughout the data and are also dropped. The data stream is then further reduced by replacing each sequence of identical 15-bit data elements with a single copy of the data element, together with an integer indicating the number of times that data element occurred. On the average, each sequence contains about 6-1/2 identical 15-bit data elements. (See figures 2 and 3.)

Since it will be desirable eventually to test the LCSR techniques with live input, live input was also used in the data gathering process. The speech data were recorded in the slightly noisy environment of a small computer center. Ambient noise sources included an impact printer, fans, an air

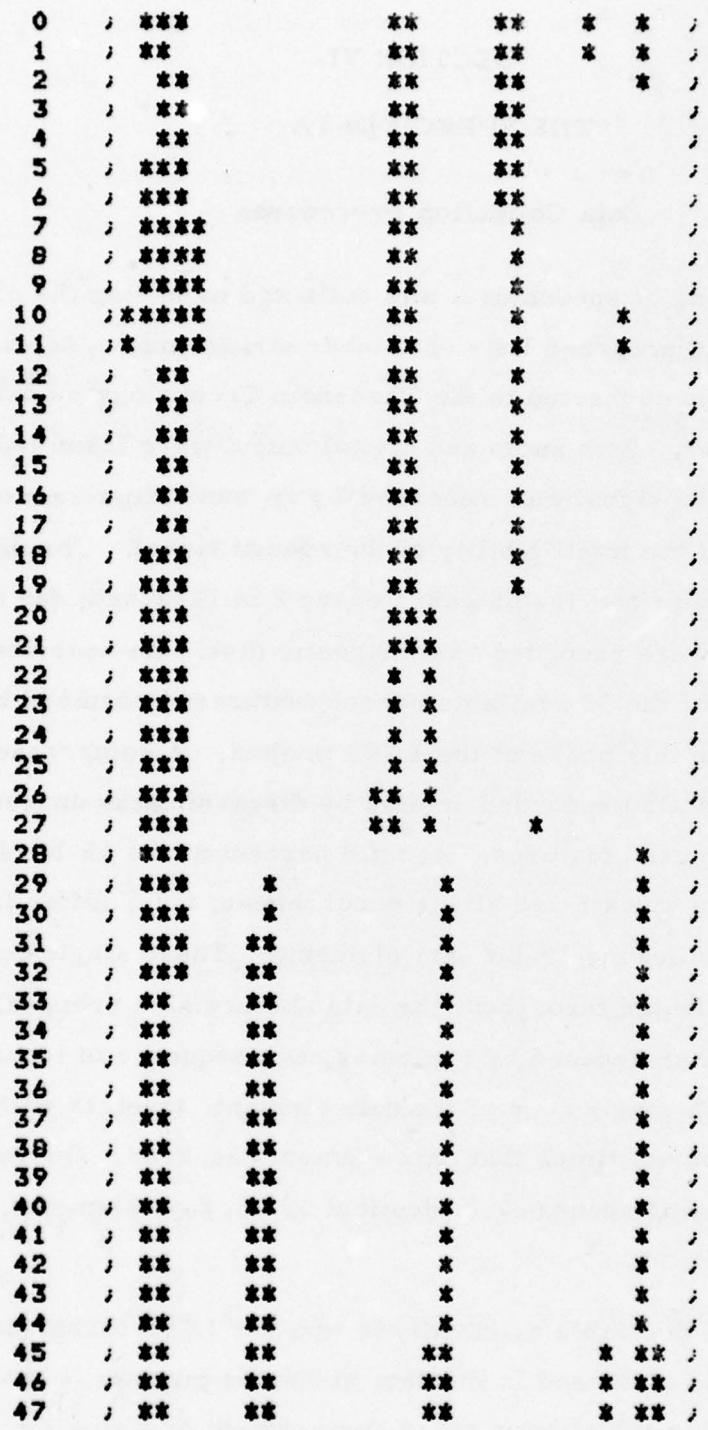


Figure 2. Raw Data Printout.

```
 1 ;*    **   *  *  ;   2  (  0,  2)
 2 ;*    **          ;   4  (  3,  6)
 3 ;*     *          ;   3  (  7,  9)
 4 ;*     *     *    ;   2  ( 10, 11)
 5 ;*     *          ;   8  ( 12, 19)
 6 ;**               ;   2  ( 20, 21)
 7 ; *               ;   5  ( 22, 28)
 8 ;  *           *  ;  16  ( 29, 44)
 9 ; **          * ** ;  17  ( 45, 62)
10 ;             *  ;   3  ( 63, 65)
11 ;        *    *  ;   2  ( 66, 67)
12 ;        *    ** ;   2  ( 68, 69)
13 ;        *    *  ;   5  ( 70, 74)
```

Figure 3. Compressed Data Printout.

conditioner, and background conversation. Although informal experiments indicated that ambient noise had little effect on the features present (bits set) in the VIP-100 output stream, speech data were either not collected or were discarded when any of these noise sources were noticed to be particularly loud. Reasonable care was taken to adjust the Telex microphone to rest about two centimeters from the speaker's lips, and to speak so that the peaks registered on the VIP-100 loudness meter were of relatively constant level.

The chosen speaker has had no specialized vocal training of any kind, but has considerable experience in using the VIP-100 in isolated word recognition systems. He has a slightly atypical American male voice with an anomalous slight accent, sometimes incorrectly identified as English or Australian. He tried, while recording the speech data, to speak very naturally and to avoid unusually careful enunciation. The speech data were collected over a three-month period on two separate days. The data are divided into sets identified by letter, each containing 55 utterances. On most days, a single set was recorded but, to observe the effects of vocal strain (and probably boredom), six sets were recorded on one day. Each set took approximately 20 minutes to record. Eighteen sets of data were recorded in total.

The 18 sets of recorded speech data have been divided into three groups of six sets each, and are designated the training data base, the interim test data base, and the test data base. Table 2 shows the dates and uses of each data set. All initial investigations of the speech data have been performed using the training data base. The interim test data base will be used to observe how well information derived from one set of data generalizes to other data, and to adjust, modify, or "optimize" recognition algorithms. The test data base will not be listed or accessed in any way before it is used to test the final version of any recognition algorithms produced.

There is considerable evidence that when a motivated speaker is given immediate feedback from a recognition system he is quickly and effectively conditioned to speak so as to achieve the best possible recognition results. This is a very useful and fortunate phenomenon, but it does tend to obscure the significance of some recognition system tests. The large test data base was recorded before any recognition algorithms were produced in order to allow separate testing of the system with and without speaker conditioning.

Vocabulary and Utterances Selected

The vocabulary was limited to a set of words which would be immediately useful in training systems, namely the digits and the word "point." This vocabulary provides an ideal test case for a strictly speech data-based recognition algorithm because it has the property that syntactical information is of limited use in aiding recognition. Many training systems could benefit from a continuous digit recognition capability including the laboratory version of the GCA – CTS currently under test at NAVTRAEQUIPCEN. A limitation of that system as it exists is the requirement to speak heading digits in turn commands in isolation; for example, "Turn right heading... zero... two... five."

It is assumed that an important cause of variability in successive voicings of a word is the influence of context. Another influence of context

TABLE 2. RECORDED SPEECH DATA.

Speech Data Set[a]	Date Recorded (1977)	Use
A	2/6	Training
B	2/7	Interim test
C	2/8	Training
D	2/9	Test
E	2/11	Training
F	2/11	Interim test
G	2/11	Training
H	2/11	Test
I	2/11	Training
J	2/11	Interim test
K	2/16	Training
L	2/17	Test
M	3/7	Interim test
O	3/10	Test
P	4/11	Interim test
Q	4/11	Test
R	4/11	Interim test
S	4/11	Test

[a] Each set of data contains 175 words from the LCSR vocabulary in 55 utterances containing from one to four words.

is to obscure the boundary between words with similar sounds at their juncture ("67"). In order to thoroughly observe and test in the presence of these contextual influences, it was desirable that the recorded speech data contain representative samples of all vocabulary items in many contexts. A simple

algorithm was devised and used to generate cycles of data, each cycle containing 11 sets of data, and each set of data having the following properties:

a. Each set contains each vocabulary item spoken once in isolation.

b. Each set also contains 44 utterances containing from two to four words.

c. Each digit occurs in each set 16 times. The word "point" occurs in each set 15 times.

d. Each digit occurs as the first word and as the last word of a multiword utterance exactly four times, and as an internal word exactly seven times. The word "point" occurs in the same positions the same number of times, except that one occurrence is missing.

e. Every possible transition from one digit to another, from a digit to the word "point," and from the word "point" to a digit occurs exactly once in a set.

The algorithm can be used to produce a large but unknown number of data cycles.

Each set of 55 utterances contains 11 words spoken in isolation, and approximately 11 utterances of three words and 33 utterances of four words. Eliminating the nonsense pair "point-point" changes one utterance from three words to two, or from four words to three.

Within a cycle of 11 sets of utterances, none of the 484 multiword utterances is repeated. Word sets A through K are one cycle of data, and sets L through S are part of one other. Some utterances may be common to more than one cycle.

SECTION VII

GENERATING EXAMPLE SPACES

Sounds, Words, and Data

The mathematical machine approach to developing an LCSR capability involves extensive manipulation of the output data obtained from the selected speech preprocessor, the Threshold Technology model VIP-100. The remainder of this report is concerned almost exclusively with those output data, and almost never directly with an acoustic speech signal. To simplify the discourse, the temptation to confound sound with its digital transformation as produced by the VIP-100 is yielded to. Although terms such as "word" and "utterance" refer to acoustical signals, expressions such as "the data base consists primarily of multiword utterances" are used. Since the VIP-100 is a deterministic preprocessor, the text goes even further and states such things as "it is difficult to precisely locate a word within an utterance" when truly discussing locating a segment of digital data within a larger segment of the same type of data. Confusion doesn't seem to arise, and the result is greater readability.

Purpose

The mathematical machine approach to developing an LCSR capability entails the detailed examination of many example voicings of each vocabulary item. The recorded speech data base consists primarily of multiword utterances. An example space (a collection of speech samples, each containing the vocabulary item of interest) can be formed by simply selecting all those utterances which contain the word of interest. An example space found in this way would also contain considerable extraneous material in the form of

words other than the one of interest. There are several reasons to cut away the extraneous material, leaving only data corresponding to the word of interest.

One reason is that, in searching an example space for information which characterizes the subject word, an important test of significance is that the information must be present in all (or at least a large fraction) of the examples. Removing the extraneous material from the example space sharpens that test of significance. Another reason for removing the extraneous material is simply to reduce the search time – a benefit whether the search is done by eye or by computer.

A Statistical Model for Utterance Segmentation

The difficulty in removing the extraneous material in an example space is of course that it is difficult to locate a word within an utterance. However, knowing the words, and their order, in an utterance is a great aid. Experience showed that, after looking at and comparing several dozen examples of VIP-100 data with known contents, the human capabilities in pattern recognition begin to break the code, and it becomes possible to estimate with some confidence the location of boundaries between words. [1]

Data segmented into words by hand were used to calibrate a statistical model of word location, which in turn was used to automatically excise words from utterances in the data base, thereby forming more efficient example spaces than would result from simple selection.

---

1. In fact, it was success in an experiment of this kind which provided the intuitive conviction that the mathematical machine approach to recognition was possible. The eye soon begins to pick out characteristic patterns in different parts of a word. The patterns are difficult to define precisely but the conviction grows stronger that they are there. Gathering the data by hand gives some tentative success, but the job soon gets out of hand and one longs for a computer's help.

TABLE 3. INTRINSIC WORD DURATION AND POSITION-DEPENDENT STRETCH FACTORS FOR HAND-MARKED VIP-100 DATA.

Intrinsic Word Duration:

Word	Point	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine
Duration (relative to "point")	1.000	1.168	0.788	0.870	0.983	1.055	1.147	1.178	1.111	0.773	0.969

Position Effect (duration relative to the first word of the utterance):

Number of Words in Utterances	Position of Word Within the Utterance			
	1	2	3	4
2	1.000	1.580	–	–
3	1.000	1.086	1.446	–
4	1.000	1.152	1.079	1.461

The statistical model and its use are described in detail in appendix A. It treats word location as a function of four influences:

a. An intrinsic word duration factor.

b. A positional influence, depending only on the ordinal position of a word within an utterance and the number of words in the utterance.

c. The speed of speech, reflected in the duration of the entire utterance.

d. Random variations.

This model predicts that, aside from random fluctuations, the duration of a word within an utterance is equal to the product of the duration of the entire utterance, an intrinsic duration factor depending only on the word, and a "stretch" factor which depends only on the number of words in the utterance and the order of the word within the utterance. The intrinsic length and position-dependent stretch factors are defined only in relation to the overall length, so these factors can be taken as unity for (for example) the word "point," and for the first word in an utterance.

The intrinsic word length and stretch factors were found by a nonlinear least squares fitting of the duration of words within utterances predicted by the model to the durations found in hand-marked data. The details of this calculation in the computer program WIZARD are described in appendix A. The length and stretch factors found are shown in table 3.

The results of this determination are interesting in that they show that the hand-marked data were remarkably consistent. The results are based on 189 hand-marked two-, three-, and four-word utterances totalling 389 words. In the process of finding the length and stretch factors which best fit the data, it was found that the minimum sum squared error in fractional duration was 0.587, indicating that the length and stretch factors in table 3 predict the observed fractional length of words within utterances, with a standard deviation of only 3.9 percent!

It is also interesting to note that the computed durations of words conforms to the intuitive observation that "eight" and "two" are quite short, with "zero" and "four" through "seven" quite long. The positional effect data show a general lengthening of successive words in an utterance, except in the second and third words of a four-word utterance. Apparently the selected speaker has a tendency to speak four-digit numbers in iambic dimeter. The magnitude of the reversal in the stretch factor between these two words is more than 8 percent (over two standard deviations), which strongly suggests that the observation has a real basis and is not a statistical artifact.

Automatic Generation of Example Spaces

The hand-marked data were further examined to determine the magnitude of the random fluctuations of word positions from the positions which are predicted by the best fit length and stretch factors. The boundaries between a given word and its neighbors in a hand-marked utterance can be "explained" as due to an additional lengthening or shortening of the word itself, beyond the intrinsic length and positional effects. Different values of this additional stretch factor, labeled F, are required to accommodate the right and the left boundaries. F values were computed for every boundary observed in the hand-marked data (271 values each for F_{right} and F_{left}), and the cumulative distribution of these F values was plotted. (Refer to appendix A.) On the basis of the observed distribution of F values, it appears that an F value of 2.75 will be exceeded very infrequently.

Safe limits can be computed for the left and the right boundaries of a given word in an utterance with known length and known constituent words by using the worst-case value of 2.75 for F to account for random variations of position. The Example Space Generator (ESG) program is used to extract portions of multiword utterances which contain desired words using this procedure. Approximately half of a four-word utterance is discarded by

this process, corresponding to about a 75-percent reduction in the extraneous material in those utterances. The amount of extraneous material discarded is less for utterances with fewer words but, as the data bases are predominently made up of four-word utterances, the potential savings in computer time is considerable.

Subsequent experience with the automatically generated example spaces has not revealed a single case where the portion extracted from a data base utterance failed to contain the example space word.

SECTION VIII

GROUPS OF SOUNDS WHICH RELIABLY OCCUR IN SPEECH DATA

The VIP-100 as a Sound Classifier

This section is concerned with the existence of, finding, describing, and testing classes of VIP-100 output which reliably occur in a fixed order when a given vocabulary item is spoken into the VIP-100 preprocessor system. The output classes which are sought are called (in this text) "groups of sounds," but one should remember that it may be impossible to characterize the set of acoustic inputs causing a reliable class of outputs as a sound (a human percept). Acoustic input examples which are indistinguishable as sounds may produce output both in and out of the output class, and it is certainly the case (with the VIP-100) that some acoustic signals distinguishable as sounds produce identical output. This study treats the VIP-100 as a device for classifying its acoustic input, and terminology used herein implies that the classification scheme implemented in the VIP-100 is the same as a human's classification; i.e., into sounds. In reality, the two schemes are similar in many ways but distinctly different.

Let a sound be defined as a unique sequence of 15 features, then the 15 binary features in the VIP-100 output selected for use in this project give that device an apparent capacity to distinguish among 2^{15} sounds; i.e., to classify sounds into 2^{15} different categories. Many of these apparently possible sound classes simply do not occur in speech data, and the VIP-100 may in fact be constructed in such a way that the set of possible output constructions is much smaller. But the number of different 15-bit combinations actually observed is quite large (at least 2^9).

It is relatively rare that any single sound (i.e., unique sequence of 15 features) occurs reliably in many voicings of a given word. Examining 96

examples of the LCSR vocabulary items revealed one sound common to every rendition of the word "zero" and another to every rendition of the word "eight." All nine other vocabulary items had no common single sounds. To characterize words one must therefore use a collection of sets of sounds rather than a collection of specific sounds.

There is a practical reason for not considering all possible subsets of the more than 2^9 possible sounds. It has to do with representation. Simply to identify one of the possible subsets (by numbering them) requires more than 256 bits of information, or 32 16-bit words. More efficient identification schemes can reduce the average number of bits in a label, but only at the cost of complexity and a significant effort to find the probability of occurrence of various subsets. Furthermore, the entropy of that distribution may not allow the order of magnitude reduction needed to get the addressing problem down to a manageable size.

The question then arises as to what class of subsets of sounds should be considered. Only a few properties of the allowed class of subsets of sounds seem desirable on an a priori basis. Three will be mentioned. First, the class should be rich in small sets, as smaller sets of sounds give a "tighter" characterization of the word. Second, the class of allowed sets of sounds should have the following property. If F is a subset of the 15 features, and V_F is a set of binary values (zeros or ones) for the features in F, the class of allowed sets should contain a set which contains only sounds with values V_F for the features in F. This requirement expresses the conviction that individual VIP-100 feature combinations are related (however imperfectly) to sounds as perceived by humans, and that the human sound classification system is one that works for spoken strings of digits. This says, for example, that a collection of sound categories based only on the number of features with value one, without regard to which features they are, is not adequate.

Finally, the class of allowed sets of sounds should be a lattice under the operation of set inclusion. This is the mathematical statement of the requirement that, for any two given sets of sounds, the class should contain a unique smallest member containing all of the sounds of both sets, and a unique largest member containing only sounds in both sets.

Once a class of sets of sounds has been selected, and the sets of sounds which reliably occur in individual vocabulary items have been found, it is possible to test, in a general way, the appropriateness of the chosen class. By examining a large number of examples of a word in which a set, S, of sounds reliably occurs, the individual sounds in S occurring in each individual example can be determined. If the collection of observed sounds nearly or completely covers S, S is an efficient characterization of that class of sounds. If relatively few of the sounds of S actually occur in words, but the class of sets does not include a set smaller than S but containing all the observed sounds, the chosen class does not allow efficient characterization at that point in the given word. If this situation prevails for many of the reliably occurring sound sets, then the class of allowed sound sets is inappropriate.[2]

The class of sets of sounds used in this project is the collection of those sets which can be specified by giving the binary value of any subset of the 15 chosen features, and which is indifferent to the value of all remaining features. Each such set corresponds to an assignment of a zero, one, or blank to each of the 15 features. A sound is a member of the set precisely when it has zeros and ones at the indicated features, regardless of the value of the other features. This class has all three of the desirable properties mentioned above, when the empty set is appended as a special object. In addition, the sets have a compact representation, and individual sounds are easily checked for membership in them.

---

[2] This test has not yet been made on the class chosen in this project.

Definition of Transition Letter Sets

Underlying the search for sets of sounds which reliably occur, in a fixed order, in words is the notion that a simple sequential recognition procedure might use those sets. The recognition procedure can be modeled as a finite automaton, which in turn can be depicted as in figure 4.

This figure represents a mathematical machine with k+1 states. It is initially in the left-hand state, indicated by an asterisk. As sounds are received from the preprocessor (after eliminating single occurrences and redundant occurrences of any given sound), they are checked for membership in the set of sounds denoted by T_1. When such a sound is received, the automaton transitions to the second state, and subsequently received sounds are checked for membership in the set of sounds T_2. When the incoming sound is not a member of the relevant set, the machine remains in its current state. Transitions are indicated by straight arrows, labeled with the set of sounds which cause transition. Loops indicate that the automaton remains in its current state when the received sound is a member of the complement of T in Σ. (Σ is the set of all sounds.)[3]

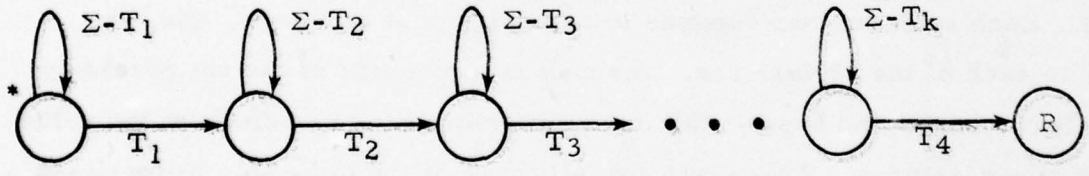


Figure 4. A Simple Finite Automaton for Word Recognition.

[3] Forego for now questions of eliminating false alarms and getting the automaton started at the beginning of a word. These problems are treated in sections to follow.

Borrowing some terminology from the theory of formal languages, of which the study of automata is a part, the input received from the preprocessor is called a letter. Σ is the set of all letters, and the automaton depicted in figure 4 is a recognizer for the formal language consisting of all "words" formed over Σ in which the first occurrence of a letter in T_1 is followed by a letter in T_2, etc. The last state is called a final, or recognition, state. Since the T_i are sets of letters which cause transition of the automaton from one state to another, they are called "transition letter sets."

Since letters and sounds are really the same thing, only transition letter sets are considered which are represented by a string of 15 symbols taken from the set zero, one, and blank. The class of sounds, or letters, is not closed under complementation, so in general the set of letters which causes the automaton to remain in its current state, denoted Σ-T in figure 4, has no such representation, but this causes no trouble.

Transition Letter Sets

An heuristic algorithm for finding transition letter sets, developed at Logicon in 1976, was used to search example spaces containing each of the LCSR vocabulary items. The algorithm was implemented in the computer program GENRLIZ (appendix D).

The example spaces from which the transition letter sets were extracted were derived from the training data base, and contained 96 examples of digits and 90 examples of the word "point." The example spaces used initially were generated automatically by program ESG, as described in section VII. The transition letter sets found in these example spaces are shown in figure 5. No transition letter sets are shown for the word "seven" because the example space used in that case was formed in a different manner, as will be described later.

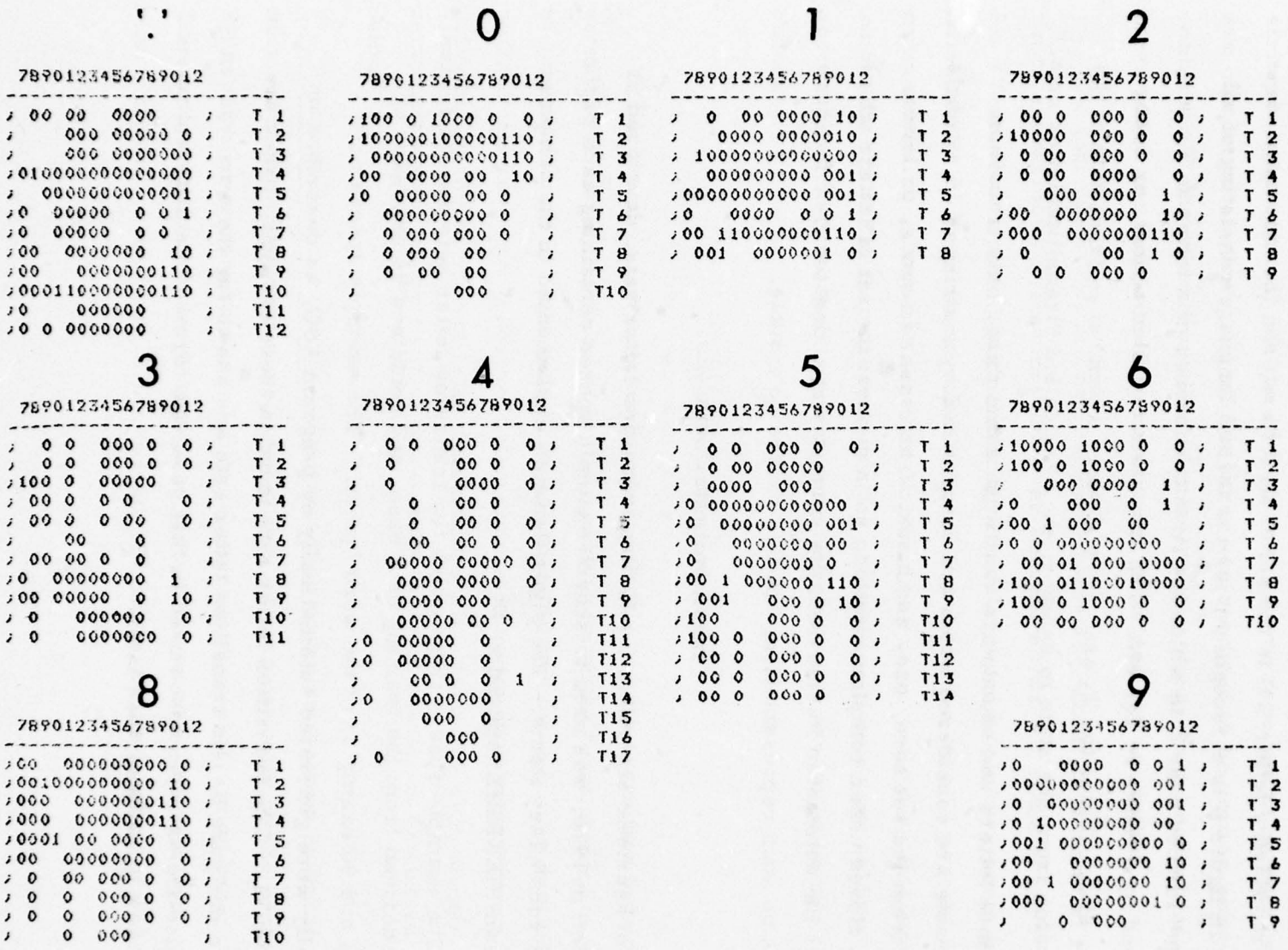


Figure 5. Sets of Transition Letter Sets from ESG-Generated Examples.

Each transition letter set is represented as a sequence of 15 symbols in a horizontal row. Ones indicate features which are on (have binary value 1), zeros indicate features which are off (binary value 0), and blanks indicate features which may be either on or off. Features used are numbered 17 through 31, and are indicated in order from left to right. The number at the top of each column helps to identify the feature. Feature number 32, on the far right, should be ignored.

None of the examples was discarded in forming the transition letter sets of figure 5. Thus every example of the word "five" in the training data base has a subsequence of 14 sounds (letters), each of which is a member of the corresponding transition letter set.

The large number of transition letter sets found for each vocabulary item (between 8 and 17) shows that the speech data for each vocabulary item have a large amount of common sequential structure. The variability of the transition letter sets within each word shows that the VIP-100 is responsive to, and reliably detects, a wide variety of different sounds. The variability of the transition letter sets from word to word indicates that the vocabulary items are quite distinct in terms of the sound classes detected by the VIP-100, a result which was expected on the basis of the VIP-100's effectiveness in isolated word recognition. Finally, that this much structure and differentiation was found to be common to many (90 to 96) examples of each word, collected over several days, suggests that there are invariant structural features in the speech data which are reliable enough for use as a basis for recognition. In general, the transition letter sets indicate that the VIP-100 is an excellent preprocessor for speech data.

To fully appreciate the remarkable amount of structure indicated by the transition letter sets, it should be noted that a feature must have a probability of being set less than 0.0072 to survive 96 random samplings as a zero, or more than 0.9928 to survive as a one. On the average, more

than 90 features have been found for each vocabulary item which meet this stringent requirement. Of the 15 features, all are required to be zero in one context or another, and all but three are required to be zero in some contexts and one in others. Furthermore, many words have several features which are required to be zero at one point and one at another.

The individual features are set (have value one) about 25 percent of the time, averaging over features and over several words. A value of zero for a feature at a particular point in a particular word would then seem to be more likely to occur than a value of one. But there are only 10 percent as many surviving as one as there are surviving as zero, indicating that even among the most reliably set features the probability of being set seldom approaches unity closely enough to survive 96 samplings without being observed as a zero.

Transition Letter Set Tests

Three preliminary tests have been performed on the transition letter sets, to get some indication of 1) their sensitivity to the training data used to generate them; 2) their generality as indicated by how well they describe nontraining data; and 3) how well they characterize the vocabulary items, as indicated by the number of false recognitions when used in a very simple recognition algorithm. Each of these tests, and the result, is discussed below.

Training Data Sensitivity – The transition letter sets shown in figure 5 were extracted from example spaces generated automatically by the program ESG. This program excises portions of utterances in the training base. The portion excised is large enough to contain the word of interest with high confidence. Because of the random variations of the location of words

within utterances, the excised portion must perforce contain extraneous material in addition to the word of interest.

A question therefore arises as to how the transition letter sets found are affected by the extraneous material. To answer this question, new example spaces were generated by hand for each of the vocabulary items. The procedure used was to examine the ESG-generated examples and remove any obviously extraneous material. The computer listing used was produced by GENRLIZ and included an indication of which letters in the example fell in each transition letter set. This information reinforced the observer's ability to discern the pattern of the word within the utterance fragment, and made the hand marking much easier (though still tedious).

New transition letter sets were then extracted from the hand-marked example spaces. These are shown in figure 6, in the same format used in figure 5. A special example space was generated for those cases of the word "two" which do not occur as the initial word of an utterance, simply by eliminating from the set for all "twos" those examples which were initial words. The transition letter sets for the noninitial "twos" are identified by the number two; those for all twos, both initial and noninitial, are identified by the letter T. The latter should be compared to the transition letter sets labeled "2" in figure 5.

The most informative comparison between the two sets of transition letter sets in figures 5 and 6 would be their relative effectiveness in recognition algorithms. Unfortunately these data have not yet been developed, and this report provides only a more qualitative comparison.

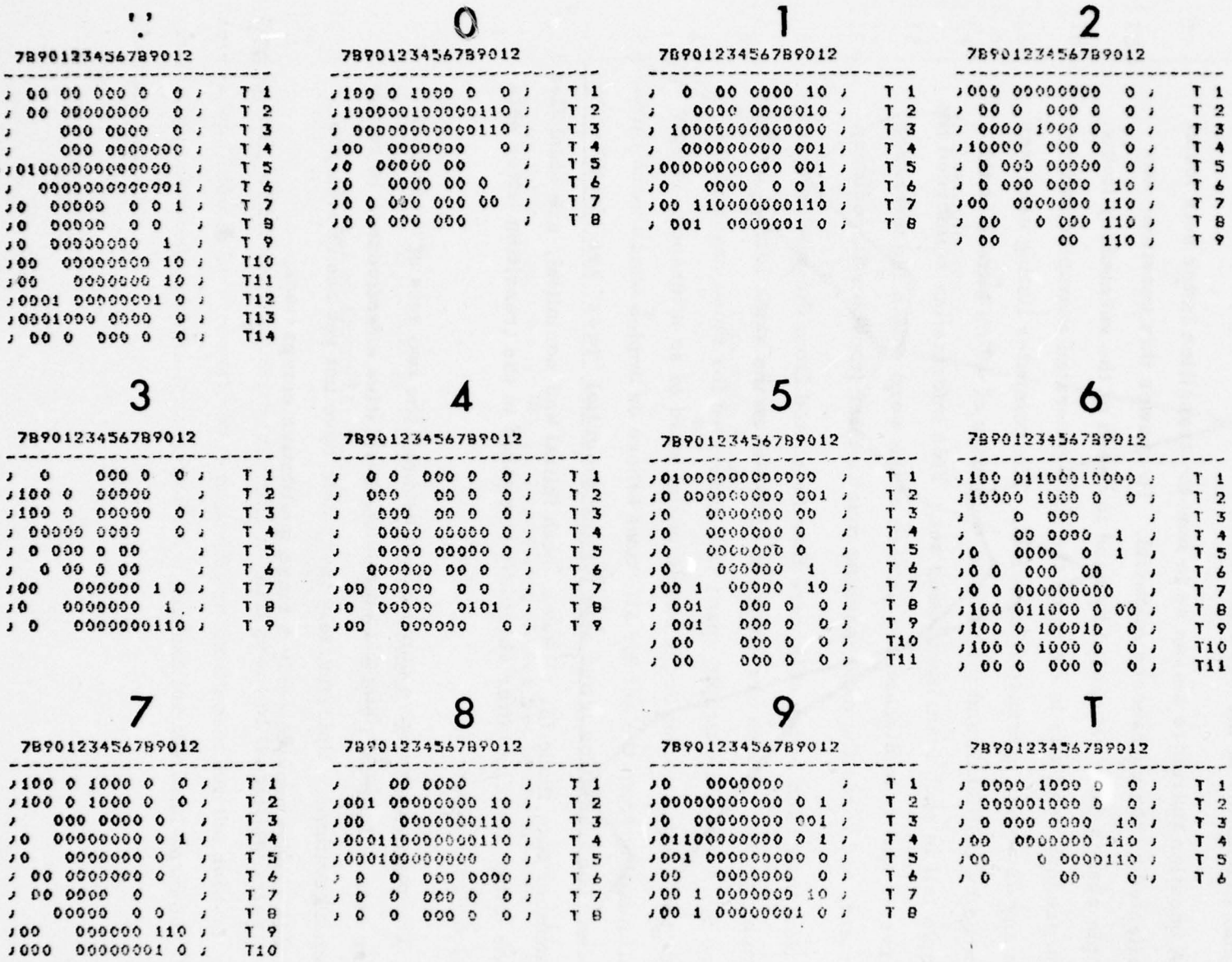


Figure 6. Sets of Transition Letter Sets from Hand-Marked Data.

There is a remarkable amount of similarity between the two groups of transition letter sets. When compared on a set-by-set basis, there is usually a clear correspondence between the transition letter sets developed from the two example spaces, including many cases of identity. The extreme case occurs for the word "one," in which there is no difference whatever between the two sets of transition letter sets. The morphological similarity between these two collections of transition letter sets suggests that GENRLIZ is very effective in extracting structure for individual words for sample sizes on the order of 100, and that the structure found is quite stable in the presence of extraneous material.

However, there are some interesting differences. The total number of transition letter sets found for comparable vocabulary items is reduced by 16 percent, from 110 to 92. This is not surprising, as removing the extraneous material reduces opportunities for false matches. Remarkably, there is some indication of a tendency for the transition letter sets to be more restrictive when derived from the hand-marked example space. An indication of this effect is shown in figure 7, which is a frequency graph of the number of features on which the transition letter sets are indifferent. The number of individual sound types (letters) in a transition letter set is equal to two raised to the power of the number of features on which it is indifferent. Therefore, the smaller the number of indifferent features in a transition letter set is, the more restricted is its characterization of the vocabulary item. The frequency graph shows that the hand-marked data have a tendency to produce transition letter sets which are indifferent on fewer features. The difference in the observed frequencies is statistically significant at the 0.05 level according to the Kolmogorov-Smirnov two-sample test, the null hypothesis being that the ESG-generated example space yields transition letter sets with fewer or the same number of indifferent features as the hand-marked example space. Another indication of the same

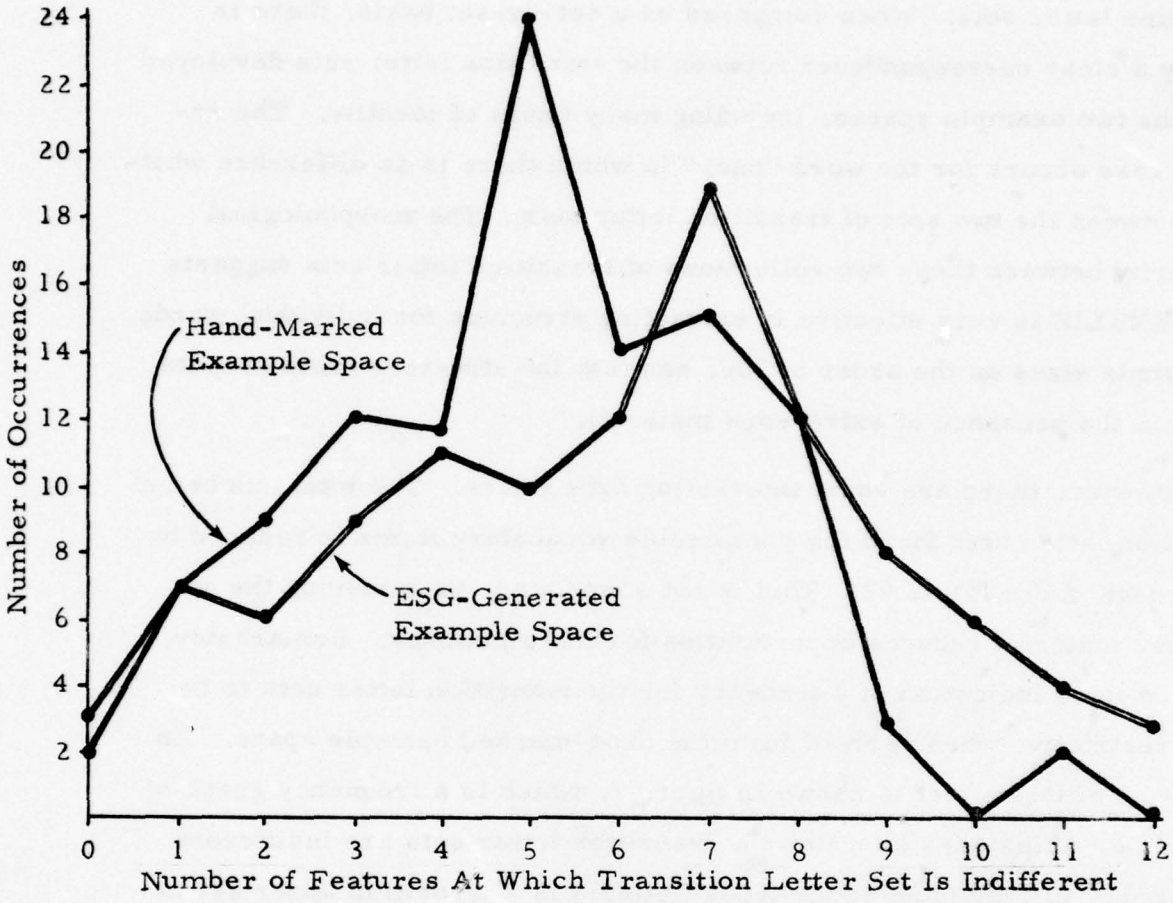


Figure 7. Histogram of Indifferent Feature Counts, for Transition Letter Sets Found from Automatically and Hand-Marked Example Spaces.

type is that, of all the features occurring in each group of transition letter sets (about 1000 in both cases), 39.5 percent are indifferent for the automatically generated example space, and 33.9 percent for the hand-marked example space.

The transition letter sets generated from the hand-marked example space were used in all subsequent testing and other investigations. Hereafter, only these are referred to.

Generality – To be useful for speech recognition purposes, the transition letter sets must be descriptive of speech samples other than those from which they were found. To test the generality of transition letter sets formed from the hand-marked example space, speech data recorded in October 1976 (over four months before the training data were recorded) were examined to determine if examples of the word "seven" contained sound sequences satisfying the transition letter sets for that word.[4]

The old speech data examined in this test consisted of three sets of utterances similar to those in the project's data base. These data included 48 examples of the word "seven" recorded by the same speaker and under circumstances similar to those used in producing the project data base. The test word occurred isolated and in the initial, medial, and final position of multiword utterances, contiguous to all words of the LCSR vocabulary.

Of the 48 examples examined, 47 had subsequences of sounds satisfying all 10 transition letter sets. The remaining example ("four six point seven") had a subsequence of sounds satisfying all but the last transition letter set. These data were gathered using a normally useful procedure which discards a segment of variable length at the end of an utterance.[5] This procedure

---

[4] The word "seven" was used in these tests because it was the first vocabulary item for which transition letter sets were found. That, in turn, was because the "seven" example space was the first to be formed. It was formed first (by hand) because the "seven" data were most familiar to the marker.

[5] Feature LP_4 (long pause) signals completion of a speech sample. The software backs up 50 VIP samples (a set of 32 features), and continues going back through the samples. When feature 26 (UVNLC), 28 ($n_1 + n_3$), or 29 ($EG_1 + EG_2$) is found, the search terminates. This signals the effective end of the feature data.

may have eliminated the missing letter. It is possible that even this single failure to satisfy all 10 transition letter sets would not have occurred while operating in real time, since the termination procedure will not be used during real-time operation.

The following chart presents an estimate of the probability that a random example of the word "seven" will have a subsequence of sounds satisfying the transition letter sets, assuming that the observed failure is in fact a failure. Symmetric 50- and 90-percent confidence intervals are also shown.

Description	Values (percent)
Maximum likelihood	97.9
50-percent confidence interval	94.4 – 99.4
90-percent confidence interval	90.5 – 99.9

These preliminary tests are an encouraging indication that transition letter sets generated by the heuristic algorithm in GENRLIZ do in fact capture a structural property of speech data which occurs with sufficient reliability to serve as a basis for automatic speech recognition procedures.

Characterizing Power – Another property of transition letters of great interest is their ability to characterize (that is, to identify uniquely) vocabulary items. The test described above suggests that almost all examples of the chosen speaker's word "seven" will pass through all the states of a simple serial recognition automaton, reaching the recognition state. Now one must consider the magnitude of the false recognition problem; how many words other than "seven" also have a subsequence of letters satisfying the transition letter set for "seven" and thus must be eliminated by some means yet to be discussed?

To investigate this question, one set of speech data used in the test described above was examined to find all subsequences of sounds which satisfy each of the 10 transition letter sets for the word "seven." The

collection of utterances was identical to those used in set A of the training data, but recorded in October 1976. These data consist of 175 words in 55 utterances and include 16 instances of the word "seven." Fifty-six sound sequences were found which satisfied the 10 transition letter sets. This included 16 sequences corresponding to the bona fide "sevens" and 40 false recognitions in 159 words other than "seven," giving an average of about 0.25 false recognitions per word.

Many of the false recognition sequences spanned several words and could obviously be eliminated by a very crude test of the time elapsed between the occurrence of the first and last sound of the required sequence. To evaluate this approach, the following procedure was used. Training data were examined to determine an upper limit to the elapsed time in the word "seven." Thirty-eight examples were timed, and their cumulative distribution plotted. The observed values ranged from 156 to 326 milliseconds, and the distribution was a smooth ogive without excessive tailing for the high values. On the basis of this distribution, 420 milliseconds was chosen as an upper limit on elapsed time which would have very low probability of being exceeded by a real "seven." Applying this criterion to the sequences satisfying the transition letter sets eliminated 24 of the 40 false recognitions. All of the 16 correct recognitions fell well within the acceptance criterion. There remained 16 false recognitions in the 159 words other than "seven," or about 0.10 false recognitions per word.

In view of the fact that several powerful techniques for eliminating false recognition are available, the preliminary result that (for one vocabulary item at least) potential false recognitions occur at the rate of about one per 10 words is very satisfying.

Where the Phonemes Aren't

Early approaches to continuous speech recognition centered upon detecting phonemes within utterances. Although an extensive literature exists, the approach was ultimately found to be unfruitful. The approach selected for the current study has been to abandon preconceived notions about the phonemic character of the data available from the preprocessor, and instead simply treat the preprocessor as a sound classifier. The fact that it is capable of chracterizing sounds is shown by its isolated word recognition capability. There is certainly no reason to hold to a phonemic representation if some other classification scheme is adequate.

That the classification scheme provided by the preprocessor is in fact different from a phonemic classification scheme can be shown empirically by examining the sets of transition letter sets. Each of these sets characterizes a portion of the particular word. It is apparent that, in some cases, the features selected for use from those set by the preprocessor fail to distinguish between sounds which are phonemically different and which can be distinguished by a human listener. As an example, the first transition letter set for the word "zero" is identical to that for the word "seven." Their phonemic spellings, however, are different: 'zɪro, 'sɛvən. Furthermore, the particular speaker's pronunciation is not unusual.

Conversely, the classification scheme does distinguish between some sounds for which the phonemic representations are the same. The phoneme /n/ appears in the characterizations of the words "one," "seven," twice in "nine," and in "point." While similarities exist between transition letter sets, T7 in "one," T9 – T10 in "seven," T6 – T8 in "nine," and T11 – T13 in "point," the letters are not identical. (See figure 6.) This indicates that the features selected provide a more precise characterization of these speech sounds than is possible phonemically. The fact that structure was found for each of the vocabulary items demonstrates that this scheme is consistent, and yet the sound classification scheme is not obviously phonemic in character.

SECTION IX

RESIDUAL SOUND GROUPS

Definition of Loop Letter Sets

Having found sound groups which reliably occur in samples of each vocabulary item, it is natural to expect that the residual portion of a word (after discarding the highly reliable parts) has limited variability and can be used to reject potential false recognitions. In perceptual terms, one does not expect vowel sounds between the stop and final fricative of the word "six," or a fricative between the initial and final /n/ in "nine." It is intuitively appealing to expect a similar construction in terms of the sound groups of the VIP-100. The transition letter sets give reliable points of reference within words, and the sounds which occur between these points can be observed in the training data. During recognition, if a sound is detected between two reference points which has never been observed there before, the word can be rejected.

A finite automaton for carrying out the simple recognition procedure just described is shown in figure 8. L_i contains the set of all sounds (letters) which are known to occur between the first occurrence of a letter in the transition letter set T_i, but before the first succeeding occurrence of a

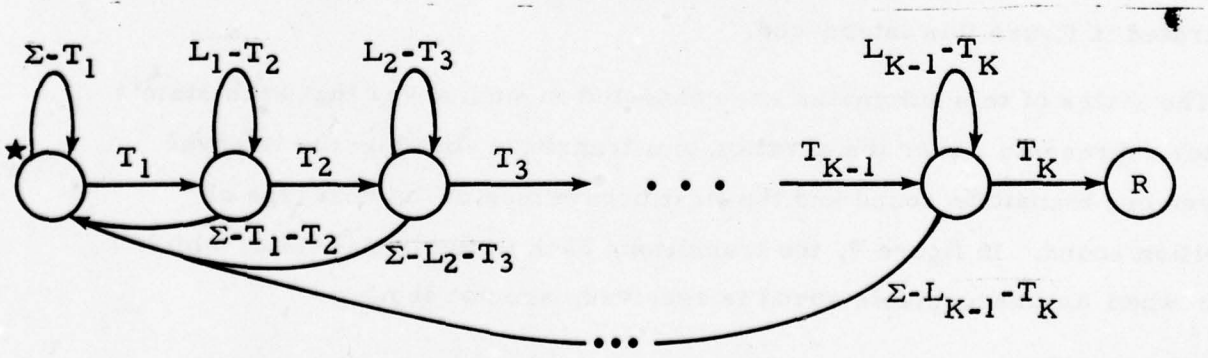


Figure 8. Finite Automaton for Recognition, Using Residual Sound Groups.

letter in T_{i+1}. When a sound is encountered which is not appropriate to its location within the potential word, the automaton reverts to its initial state, ready to recognize the word should it occur later.[6] The residual sound groups are seen to be associated with the loop transitions in this automaton.

The question again arises (see page 50) as to what class of sets of sounds is appropriate for describing the residual sounds. For uniformity within the recognition procedure, and for reasons similar to those given in connection with transition letter sets, the same mathematical representation of sounds is used to describe both the residual sounds and the transition sounds, i.e., 15 zero's, one's, or blank's.

The automaton of figure 8 must be further modified before the treatment and definition of residual sounds can be made precise. Notice that this automaton transits from one state to the next on the first occurrence of a letter in the state's associated transition letter sets. Eventually, the durations of, and intervals between the reliably occurring sounds of the transition letter sets must be timed. To do this, a counter is associated with each state and, whenever a transition (including loop transitions) leads to a state, that state's counter is incremented by the time duration of the sound causing the transition. Using this implementation, the automaton of figure 8 will allow timing the interval between first occurrences of letters in each transition letter set, without distinguishing between the duration of the transition type of sound and a residual type of sound. To correct this deficiency, the finite automaton illustrated in figure 9 is introduced.

The states of this automaton are connected in such a way that each state's counter represents either the duration of a transition sound or the interval between one transition sound and the first occurrence of the next type of transition sound. In figure 9, the transitions back to the initial state, which occur when an unacceptable sound is received, are not shown.

---

6. Forego again the question of how to get the machine started at the right time.

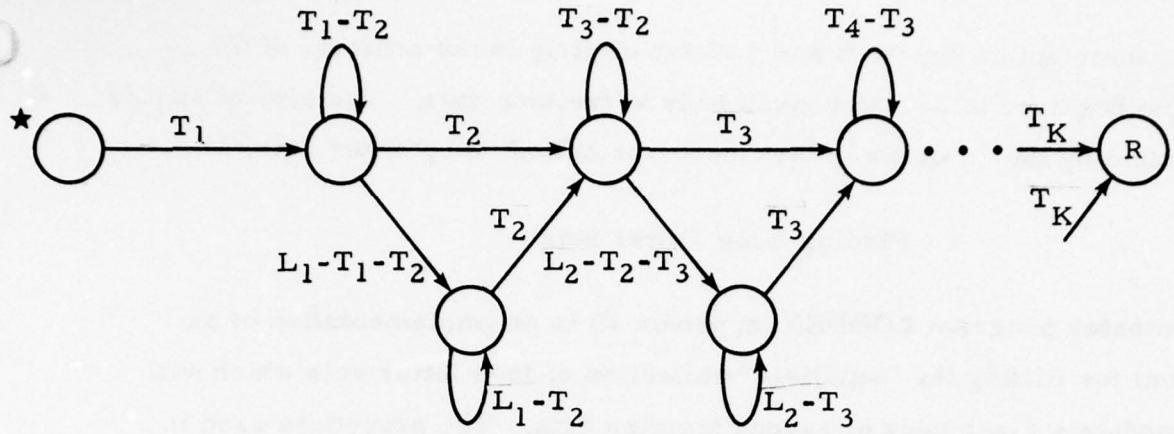


Figure 9. A Finite Automaton Which Facilitates Timing Both Durations of and Intervals Between Transition Sounds. (Returns to the initial state are not indicated.)

As indicated in section VII, the class of allowed letter sets is not closed under the operation of complementation. In other words, one cannot represent, using 15 zeros, ones, and blanks, the letter sets included in, for example, "; 00 00 0000 ;" but not those in ";100 0 1000 0 0 ;". The letter sets T_i-T_{i+1}, L_i-T_{i+1}, and L_i-T_i-T_{i+1} are therefore not of the allowed class. They cannot be described by specifying values on some features and remaining indifferent to the values on others, and they cannot be represented as a string of 15 zeros, ones, and blanks. This causes no difficulties, however, since the set differences are effectively implemented by ordering the test performed on incoming letters. In a typical state of the automaton, a newly received letter is first checked for membership in T_i, then for membership in T_{i-1}, then in L_i. Transition occurs the first time membership is detected, and no further tests are made. If no membership is detected, the transition is to the automaton's initial state. It is easy to verify that the result is exactly the same as if the set differences were used explicitly. Implementing this automaton thus only requires explicit representation of the transition and loop letter sets, and not their differences.

The autor.ata of figures 8 and 9 differ slightly in the contents of the L_i which are required to accept a given body of training data. The sets of sounds represented by the L_i in the latter figure are called "loop letter sets."

Finding Loop Letter Sets

Computer program LOOPER (appendix B) is an implementation of an algorithm for finding the "smallest" collection of loop letter sets which will accommodate a given body of speech training data. The procedure used in effect operates the finite automaton depicted in figure 9, noting when a loop letter set must be enlarged to accommodate a new sound.

The class of allowed letter sets is not closed under the operation of adding a new letter, or union with singletons. Therefore, this enlarging has a generalizing effect, reflected in the fact that the loop letter sets finally formed contain more individual sounds than are actually observed in the appropriate parts of the training data words. Since the chosen class of allowed letter sets is a lattice under set inclusion, and contains all singletons, there is a unique smallest letter set (easily found) which enlarges a given letter set to include a given letter (sound). Furthermore, this operation is associative, so the order in which the enlargement is done is irrelevant, and a simple serial process can be used to successively open the loop letter sets to accommodate new examples. The result is the smallest loop letter set which contains all the observed letters.

Technical difficulties encountered in LOOPER have primarily to do with the fact that sometimes there are several points in an example where the simple serial automaton of section VIII can be started and reach the recognition state. In general each such path through the word generates a different collection of loop letter sets, and the problem is to choose a start point for each example in such a way that the resulting loop letter sets are as small as possible, giving the sharpest possible characterization of the word.

Loop Letter Sets Found

Program LOOPER was exercised on the hand-marked example spaces for each vocabulary item, using the transition letter sets found for each vocabulary item as described in section VIII. The loop letter sets which were found are shown in figure 10, interlaced with their associated transition letter sets. (The format of this figure was described in connection with figure 6.)

In several cases no sounds were observed to occur between the last sound in one transition letter set and the first sound of the next. In these cases, the loop letter sets are empty, and are indicated by writing out "empty set," since the empty set does not have a representation of the standard form. These cases are the ultimate in discrimination, as any sound observed between the associated transition letter sets will cause rejection of the word.

The Nature of Residual Sound Groups

The remarkable feature of the residual sounds is that they only occur infrequently. The transition letter sets contain most of the sounds which make up the entire word, or at least that portion which occurs after the first member of the first transition letter set, as the following demonstrates. In the process of generating the loop letter sets, 96 examples of each digit were e were examined. Of the 19 nonempty loop letter sets occurring in the words "zero," "one," and "seven," 7 loop letter set states were entered 4 or fewer times, 11 were entered 7 or fewer times, and 16 were entered 25 or fewer times. The most frequent occurrence was the second loop letter set state for the word "one," which was entered 56 times. It appears that the median probability that a randomly selected nonempty loop letter set state will be entered on a randomly chosen example is about 6 or 7 percent.

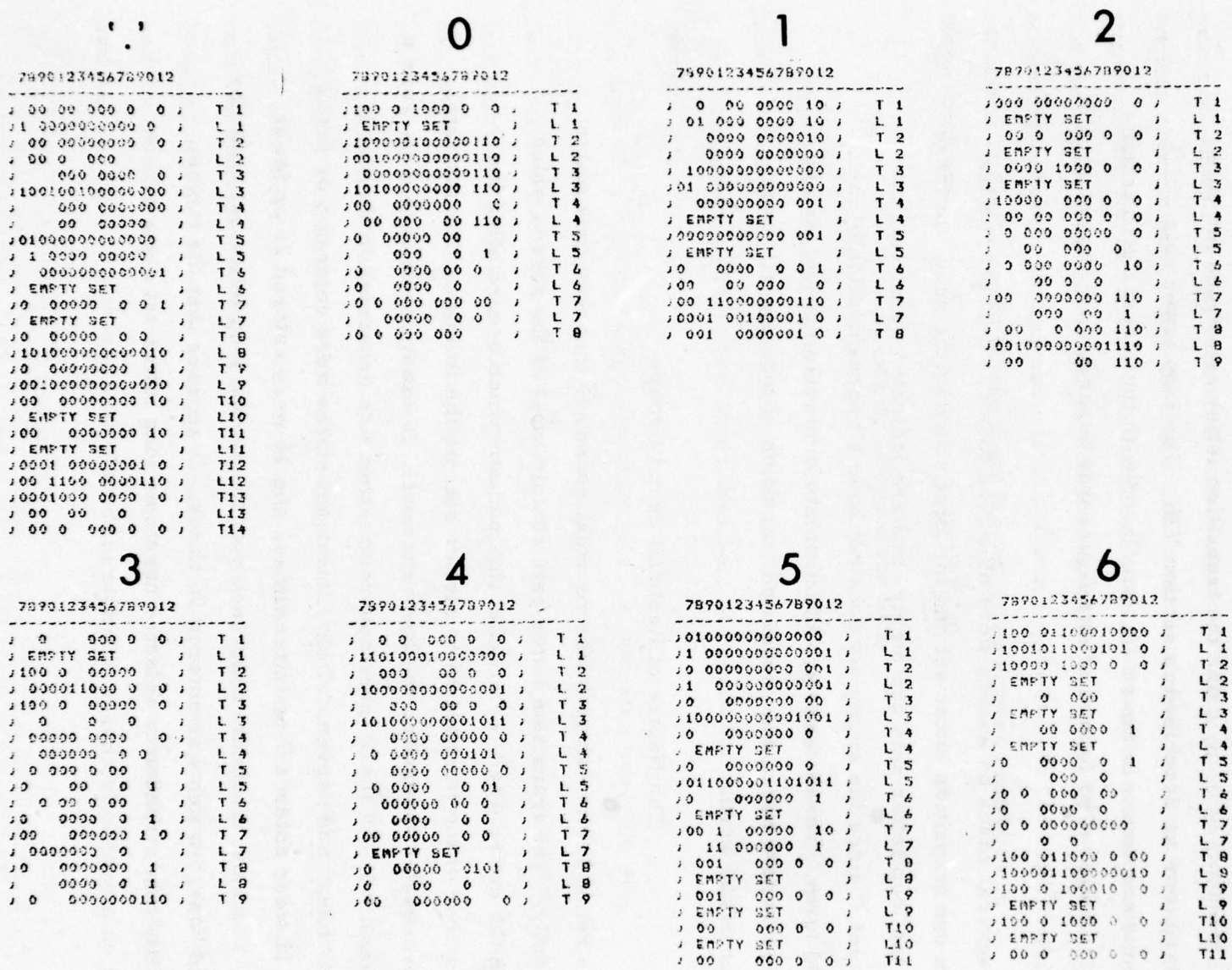


Figure 10. Loop Letter Sets Interlaced with their Associated Transition Letter Sets (Sheet 1 of 2).

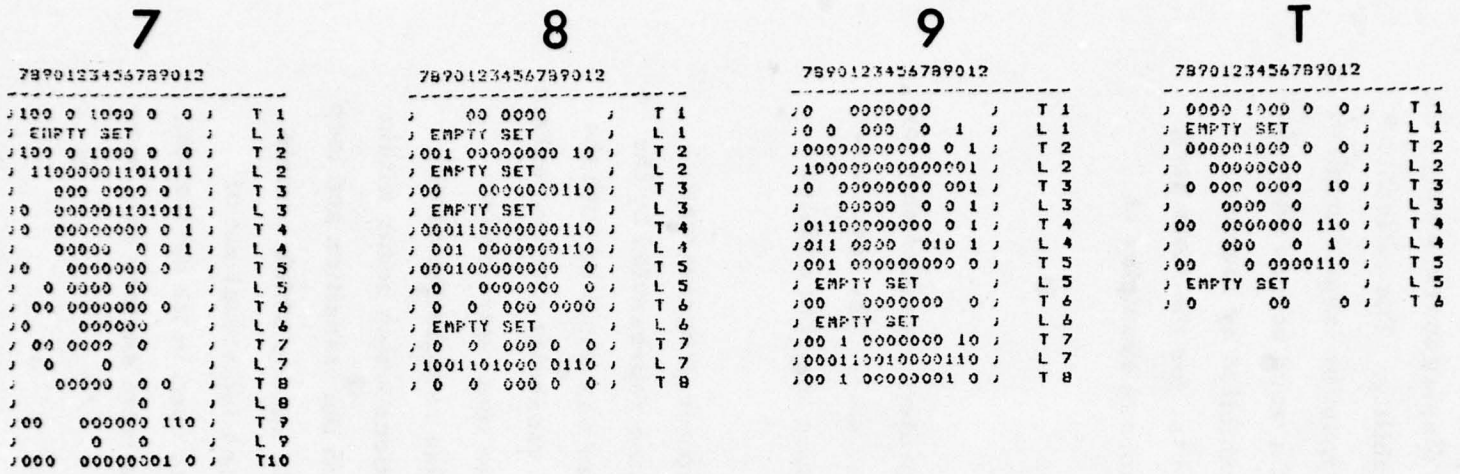


Figure 10. Loop Letter Sets Interlaced with their Associated Transition Letter Sets (Sheet 2 of 2).

The fact that loop letter set states are relatively seldom entered may be interpreted as an indication that the VIP-100 data display more inter-example variability than localized intraexample variability. The rationale for this observation is that the transition letter sets must be large enough to accommodate the variability of a localized portion of a word across examples. But the result given above shows that the variation exhibited by sounds occurring between contiguous locations within a word is, for the most part, contained in (and therefore less than) the variation across examples at those locations.

Tests of Loop Letter Sets

Resources available in this phase of the LCSR project provided only for a very limited test of the usefulness of the loop letter sets. The test indicates how effectively the loop letter sets help to reduce the rate of false recognitions.

As will be described in the next section, a computer program was developed which implements the recognition procedure represented by the finite automaton depicted in figure 9. This program was supplied with the transition and loop letter sets for each of the LCSR vocabulary items, except noninitial "two." These recognition algorithms were then exercised on training speech data sets A and C to observe the false recognition rates. Notice that false recognitions are apparent recognitions which occur within utterances not containing the word recognized. Both the transition and loop letter sets were derived from hand-marked example spaces which contain only the individual word of interest. Therefore, each individual set of loop and transition letter sets was tested on data not used in its derivation, and the results are as valid as they would be if data from another data base had been used.

Data base sets A and C, combined, contain 350 words in 110 utterances. When the 11 recognition algorithms were exercised on these data, 507 potential recognitions occurred, 157 of them false, or an average of 14.3 false recognitions per vocabulary item. As there were approximately 318 words other than the one each individual automaton should recognize, the false recognition rate was 0.045 false recognitions per word per vocabulary item.

In section VIII, the false recognition rate using only transition letter sets was evaluated for the word "seven." The result obtained was approximately 0.25 false recognitions per word. When tested with both loop and transition letter sets, there were only two false recognitions in the 318 words other than "seven," or 0.006 false recognitions per word, indicating a very significant improvement.

Among the false recognitions observed in this test, the word "two" was exceptionally common, 57 of 157 being that word. The average false alarm rate for vocabulary items other than "two" was 0.031 false recognitions per word per vocabulary item, but for "two" the result was 0.179. This observation gave the impetus to distinguish between initial and noninitial occurrences of that word. It was noticed that the initial stop associated with the phoneme /t/ was a reliable feature of the data for the noninitial cases, but was often absent in the initial cases. A special example space was then prepared by simply removing the initial cases from the original example space. Transition letter sets were then derived for the new example space. The first transition letter set in the newly produced group clearly reveals the stop features. (See T1 for "two" (not T) in figure 6.)

SECTION X

CHARACTERIZATION BY TIME

Concept

As discussed in section IV, the mathematical machine approach for LCSR assumes that the durations of, and intervals between, the characterizing sounds identified by the transition letter sets are an additional vehicle for characterizing vocabulary items, and for rejecting false recognitions.

The resources available in Phase 0 of this project permitted only a very preliminary test of this concept, and it is reported in this section.

Counters for timing the duration of a mathematical machine's stay in any particular state were introduced at the beginning of section IX, in connection with a finite automaton which provides for one counter for each transition letter set, and for the interval between the last letter of one transition letter set and the first letter of the following transition letter set. Letters are received from the VIP-100 preprocessor at a constant rate of 500 per second, so accumulating a count of letters received while in any state is equivalent to timing the duration of stay in that state.

One approach to using the durations and intervals to reject false recognitions is the following. Let $c(T_i)$ and $c(L_i)$ be a counter associated with a transition letter set state and a loop letter set state, respectively. Then when the recognition state is reached (after the Kth transition), 2K-1 counter values will have been accumulated. By observing the counter values occurring in the training data, one can estimate the mean, variance, and covariances among the individual counter values. Treating all the individual counter values as a vector, one obtains from training data a mean vector and the covariance matrix of the counter vector.

The stochastic deviation (or the Mahalanobis distance) of any particular recognition from the population of recognitions experienced during training is then given by

$$\delta = [(\bar{c}-\bar{p})' \; C^{-1} \; (\bar{c}-\bar{p})]^{1/2}$$

where $\bar{c}$ is the vector of observed counter values.

$\bar{p}$ is the vector of counter means.

C^{-1} is the inverse of the covariance matrix.

The stochastic dispersion, δ, indicates how similar any particular recognized word is to the training data for that word, on the basis of the durations of, and intervals between, its characteristic sounds. Many, if not all, false recognitions can be expected to be eliminated on the basis of peculiar timing details, revealed by large values of δ, computed as indicated above.

Preliminary Test

A greatly simplified form of the concept just stated was applied to real and false recognitions obtained using only the transition letter sets for the word "seven." This procedure was a continuation of the test of the false recognition rate using the simplest recognition algorithm and only the transition letter sets, described in connection with characterizing power in section VIII.

As described in section VIII, an "old" set of speech data containing 175 words in 55 utterances revealed 56 sequences of letters satisfying all of the transition letter sets for the word "seven." Of these, 40 were false recognitions occurring in the 159 words of the speech data which were other than "seven." Among the 40 false recognitions, 24 were eliminated on the basis of extremely long duration from first to last state, leaving 16 false recognitions and 16 real recognitions for detailed investigation.

Since only the transition letter sets were available at the time this test was performed, counters were in effect attached to each transition letter set, and only the intervals between each transition were observed. In effect, only the sum of $c(T_i)$ and $c(L_i)$ was used, rather than their individual values. Thirty-eight samples in the training data were examined to "calibrate" these counters. Mean values and standard deviations were computed for each, and correlation coefficients were computed for a few pairs of counters. On the basis of these preliminary data, it was decided to simplify the hand calculations by summing the counters for the first pair, and for each of the last two pairs of transition letter sets. In total, seven counter values, either individual or summed, were used in the remainder of the test.

The distribution of several of the counter values was distinctly asymetrical. The mean deviation above and below the total population mean varied as much as 26 percent. To rectify the effective distribution of counters, a weighted deviation from the mean was used, defined by

$$\xi = \frac{c - \mu}{w}$$

where w = the mean of positive deviations if $c - \mu \geq 0$.

= the mean of negative deviations if $c - \mu < 0$.

To further simplify the hand calculations, correlation among counters was not considered, and a modified form of the stochastic deviation was computed, defined by

$$M = \sum_{i=1}^{7} \left(\frac{c_i - p_i}{w_i} \right)^2$$

where the weighting factor, w_i, was chosen as described above.

Notice that the similarity measure used in this hand analysis has several weaknesses relative to the similarity measure which will be used in the final recognition algorithm. Specifically:

a. Correlation among sound durations is ignored.

b. No duration data for loop letter sets is used.

c. The statistics are based on only 38 samples in the training data.

When the similarity measure was computed for the 32 sequences of letters in the test data surviving the preliminary screening, reasonably good separation was obtained. Eleven of the 16 sequences not due to a "seven" had M scores over 100, while all but one sequence due to a "seven" had M scores less than 30. Three sequences not due to "seven" had M scores less than the largest "seven" sequences, and one "seven" sequence had an M score larger than the smallest nonseven. By placing an acceptance threshold at 27.1, no false acceptances would have occurred, with one false rejection.

Upon closer examination, it was found that the three lowest M scores not arising from "seven" occurred within the word "zero." Previous experience had shown that "seven" and "zero" are reliably separable on the basis of the behavior of features 17 and 30. Relating this experience to the offending sequences, it was noticed that the number of letters occurring after the last letter in the second, but before the third, transition letter set was closely related to the known phenomenon. (These letters will be counted and associated with the second loop letter set in the final implementation of the recognition algorithm.) A modified similarity measure was therefore formulated, using the new letter count in the same way as the others are used.

When the modified M scores were computed, the sequences corresponding to occurrences of "seven" were perfectly separated from the others. The maximum score for a "seven" was 76, and the minimum score for other sequences was 90. Perfect recognition, with no false rejection, would

therefore occur with a threshold value of 80. Figure 11 is a histogram showing the computed modified M scores for the 16 occurrences of "seven," and the four other lowest scores.

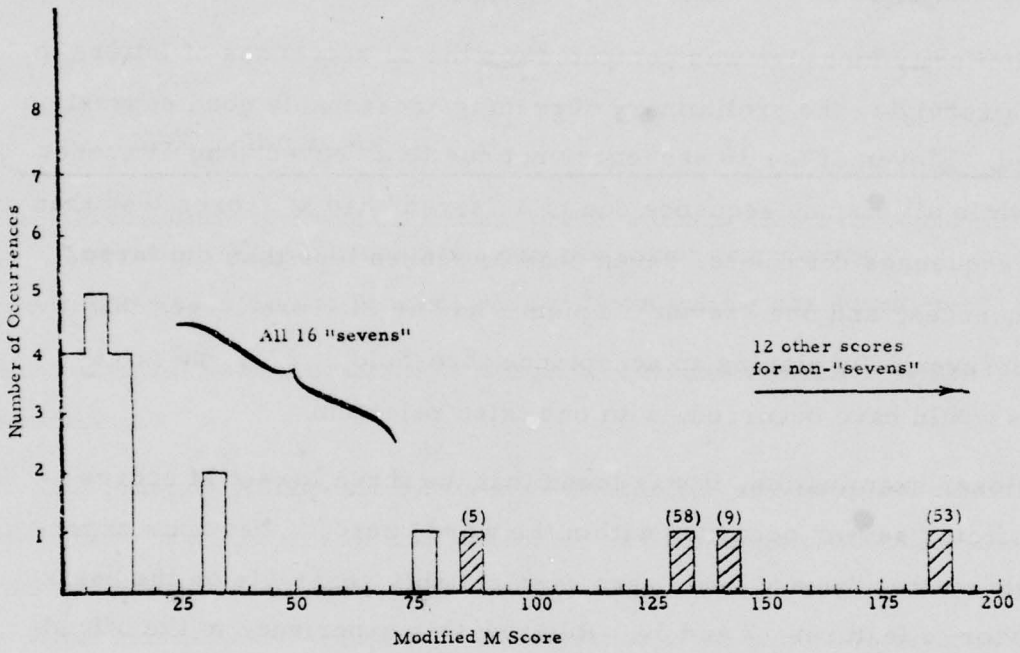


Figure 11. Histogram of Modified M Scores for 16 "Sevens" and Four Lowest Scoring Non-"Sevens."

It is very encouraging that complete separation was obtained between the 16 "sevens" and the 159 other words in this limited test set, especially in view of the many features of the recognition technique which have been ignored or weakened to make a hand check possible. This success should be considered only an indication of the true power of the mathematical machine recognition technique; evaluation of its limitations will have to wait for complete implementation of the algorithm and thorough testing.

SECTION XI

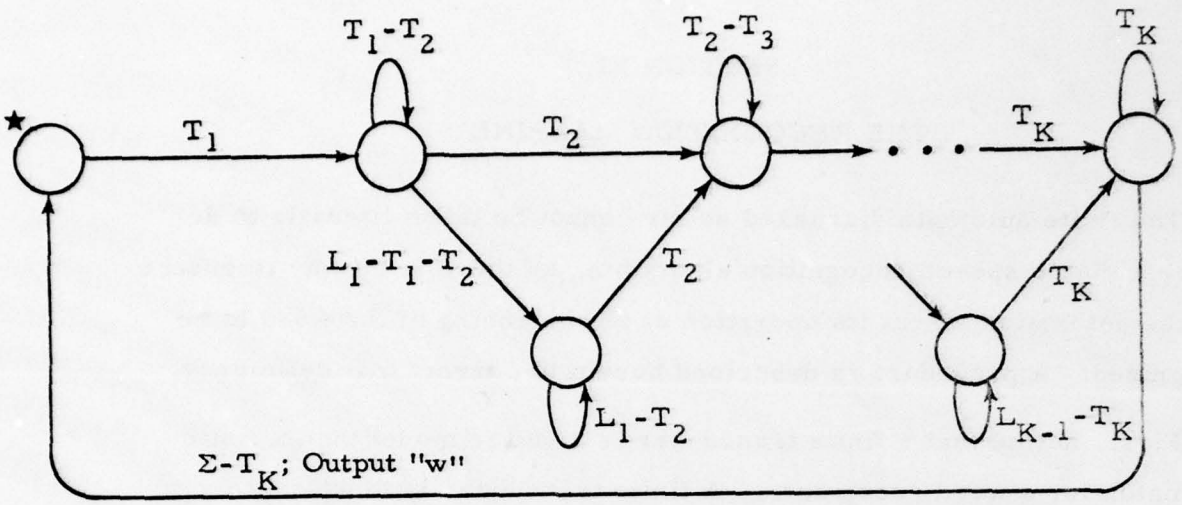
THE RECOGNITION MACHINE

The finite automata discussed so far cannot be taken literally to describe a viable speech recognition algorithm, as there is no way to ensure that the automaton starts its operation at the beginning of the word to be recognized. A procedure is described herein to correct that deficiency.

First, notice that a finite transducer is a better model than a finite automaton for a word recognizer. A finite transducer is identical to a finite automaton except that it has the capability to produce output when it makes a transition. A word recognizer must make known to the outside world the fact that it has detected the presence of a word. The finite transducer model then only produces output when recognition occurs, and, for convenience, it is assumed that its output alphabet is the same as the Arabic form of the LCSR vocabulary. Then the "seven" recognizer, on recognizing its word, emits "7."

The final transition within the automata described earlier is to the recognition state. However, it is desirable to count the duration of the last characteristic sound as well as the others, to gain one more piece of information for temporal characterization. The transducer should therefore stay in the "recognizing" state until the last sound has been measured. Afterward, it should return to the initial state and await the possible reoccurrence of the word it is designed to recognize. The finite transducer model which results has the appearance shown in figure 12.

The finite transducer is still incapable of recognizing words reliably because of the difficulty of getting started at the proper time. Thinking in terms of a computer algorithm, the solution seems clear; all that is necessary is to have many "copies" of the transducer available, starting one at each occurrence of a letter from T1. A mathematical machine model which is



Note: Transitions to the initial state occur when no other transition is possible. These transitions are not shown for clarity.

Figure 12. Finite Transducer for Recognizing the Word "w."

almost equivalent to this concept is that of a nondeterministic finite transducer.

The primary difference between deterministic and nondeterministic automata or transducers is that the latter can make transitions in several different directions at once. (There is no randomness to their operation as the name erroneously suggests.) Recognition of an input sequence is formally defined in such a way that, if any possible choice of allowed transitions through the states leads to a recognition state (or an output), the input is recognized.

The difference in appearance between a deterministic and nondeterministic finite transducer can be subtle. All that is required is that some state have more than one possible transition for a given input letter. In terms of sets of transition-causing letters, the intersection of two-transition causing letter sets must be nonempty. The automata and transducer described previously are easily seen not to have that property.

To model the starting of a new copy of the transducer each time a letter from T_1 is received, it is only necessary to make the finite transducer of

figure 12 nondeterministic at its initial state. The desired behavior on receipt of a letter from T_1 is to let one copy transition to the first noninitial state, and create a new copy which is still in the initial state, awaiting the next occurrence of a starting letter. As suggested above, the actual difference is vast, but the difference in appearance is subtle. Figure 13 depicts the nondeterministic finite transducer model.

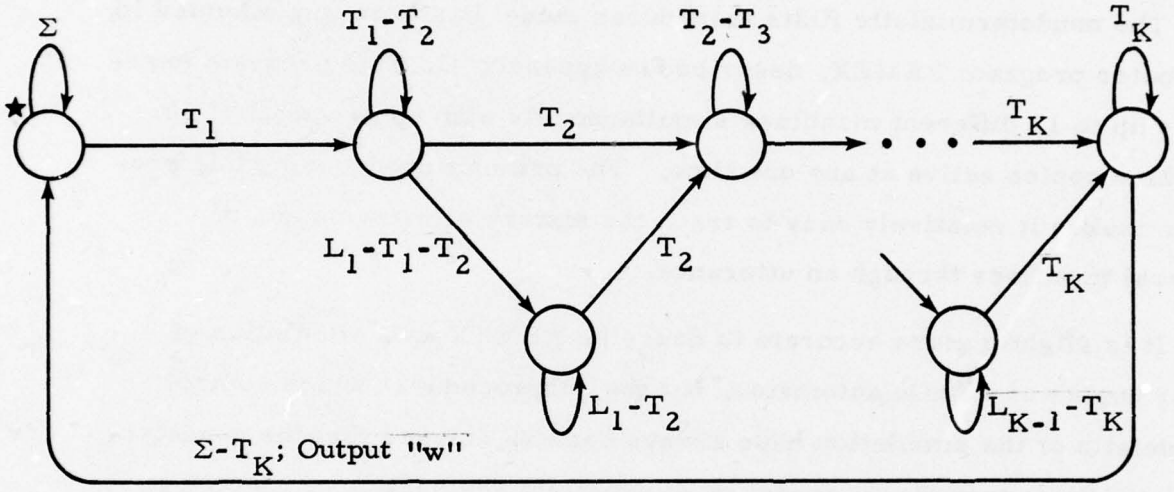


Figure 13. Nondeterministic Finite Transducer for Recognizing the Word "w."

The nondeterministic finite transducer model carries the implication that, when two or more copies of the underlying machine move into the same state, only one of them is retained. In a computer simulation of this type of device, different copies could be represented simply as entries in a table, the entry for a given copy being the state that particular copy is in. Unless one is intent upon simulating a nondeterministic transducer precisely, there is no absolute requirement to guarantee that entries in the table are distinct; i.e., guarantee no two copies of the device are simultaneously in the same state. In practice, however, it has been useful to allow only one copy of a

machine in each state, because it reduces the number of machine copies required. The general rule followed for dropping machine copies is to retain the copy which has the greatest potential to ultimately result in recognition.

Computer Implementation

The nondeterministic finite transducer model has been implemented in computer program REMEX, described in appendix C. This program exercises up to 12 different machines simultaneously with up to a total of 50 machine copies active at any one time. The printout produced by this program makes it relatively easy to trace the history of many copies of several machines through an utterance.

It is slightly more accurate to describe REMEX as a simulation of many copies of a finite automaton, but the subprocedures which control the details of the simulation have always been so devised that the nondeterministic finite transducer model is an excellent description of its operation.

Notes from Formal Languages[7]

It is a well known fact that any nondeterministic finite automaton or transducer has an equivalent (in the sense that the language accepted by both is identical) deterministic form. It may occur to the informed reader that the simulation implemented in REMEX might be simplified by transforming the nondeterministic machine to its deterministic equivalent before simulation. It is a curious fact that REMEX can easily be interpreted in several ways; viz.: (1) as a single deterministic machine with a complex state description; (2) as a simulation of many simultaneously operating, but simpler, machines; or (3) as a nondeterministic machine.

---

7. For a definition of the concepts from formal languages used here, see, for example, Theory of Automata. 1969. Arto Salomaa. Pergamon Press. Oxford, London, Edinburgh, New York, etc.

The sequence of machines described in this report can be described from the point of view of the formal languages they accept. Let Σ, as before, represent the set of all letters and T_i and L_i denote subsets of those letters.

The simple serial finite automaton of section VIII accepts the language described by the regular expression

$$W_1 = (\Sigma - T_1)^* \, T_1 \, (\Sigma - T_2)^* \, T_2 \cdots (\Sigma - T_k)^* \, T_k$$

In section IX, when loop letter sets are introduced to reduce false alarms the language accepted by the first form of the machine has the description

$$W_2 = (\Sigma - T_1)^* \, T_1 \, (L_1 - T_2)^* \, T_2 \cdots (L_{k-1} - T_k)^* \, T_k$$

Modifying the automaton to provide for timing both the duration of, and intervals between, characteristic sounds modifies the expression but not the language accepted. The new regular expression describing the language is

$$W_3 = (\Sigma - T_1)^* \, T_1 \, [(T_1 - T_2)^* + (L_1 - T_1 - T_2)(L_1 - T_2)^*] \, T_2 \cdot$$
$$[(T_2 - T_3)^* + (L_2 - T_2 - T_3)(L_2 - T_3)^*] \, T_3 \cdot$$
$$[(T_{k-1} - T_k)^* + (L_{k-1} - T_{k-1} - T_k)(L_{k-1} - T_k)^*] \, T_k$$

Adding a loop to the last state for timing the duration of sounds in the last transition letter set, and "recognizing" when that state is exited, causes the description to become

$$W_4 = W_3 \, T_k^* \, (\Sigma - T_k)$$

where W_3 is the regular expression developed above. When the finite automator model is dropped and a finite transducer is substituted, with output "w" emitted on transition back to the initial state, the result is a transducer which emits the output sequence w^n when the input sequence is a member of the language described by

$$W_5 = [W_4]^n (\Sigma - T_1)^*$$

Finally, the nondeterministic finite transducer model is designed so as to emit the output "w" once for every unique decomposition of the input sequence into a word of the form $\alpha\beta\gamma$, where α and γ are in Σ^* and β is in the language described by W_4.

These formal language descriptions of the mathematical machines for recognition, and the algorithms underlying them, may be useful for investigating further developments, such as extended error tolerance.

SECTION XII

SUMMARY, RESULTS, CONCLUSIONS, AND RECOMMENDATIONS

Summary and Results

This report has traced the work accomplished and the results to date of an investigation of a limited continuous speech recognition (LCSR) technique. The study began by formulating the unique requirements for recognition of continuous speech to support Navy training systems. It was shown that these systems typically involve a small, fixed vocabulary where higher knowledge sources, such as semantics, are often irrelevant. Moreover, a highly accurate real-time algorithm is required which can be supported by minicomputer-based processing. Speaker independence, while convenient, is not a necessity.

The speech recognition literature was reviewed to determine if any existing technique satisfied these particular requirements. None were found, but the survey did reveal certain trends in current speech research, namely:

a. Treating the speech preprocessor as a sound classifier rather than a phoneme detector.

b. Emphasizing the derivation of the recognizable speech characteristics from real speech data.

c. Decoding the speech signal sequentially instead of by exhaustive hypothesis-and-test methods.

A novel approach toward the LCSR problem was conceived by Logicon in 1976 which follows these trends. Based on concepts from the theory of mathematical machines, a simple sequential recognition procedure was

modeled as a finite automaton. Continuous speech, it was postulated, could be characterized and recognized on the basis of observing: 1) characteristic classes of output from a preprocessor, 2) the order in which these occur, and 3) the characteristic time durations between the output samples. The method of discovering the characteristic output classes and time durations is direct automated examination of speech data. An initial implementation effort was defined to determine if indeed these assumptions were valid for the 10 digits and the word "point." A Threshold Technology preprocessor and Nova 1200 minicomputer supported the research.

The investigation began by collecting a large corpus of utterances from a single speaker. The utterances were carefully chosen in order to observe the speech data in the presence of various contextual influences. Nine-hundred-ninety utterances were recorded for a total of 3150 words. These data were divided into a "training" set, an "interim test" set, and a "test" set. The training data were further divided into example spaces for each vocabulary item. This segmentation of the data was supported by a statistical model which predicts that, aside from random fluctuations, the duration of a word within an utterance is equal to the product of the duration of the entire utterance, an intrinsic duration factor depending only on the word, and a "stretch" factor which depends only on the number of words in the utterance and the order of the word within the utterance. The intrinsic word length and stretch factors were found by a nonlinear least squares fitting of the duration of words within utterances predicted by the model, to the durations found in hand-marked data. These factors predicted the observed fractional length of words within utterances with a standard deviation of only 3.9 percent.

A class of sets of sounds output by the chosen preprocessor was defined. Borrowing some terminology from the theory of formal languages, the sounds input from the preprocessor are called letters. The characterizing sets of sounds which this mathematical machine approach postulates are

termed transition letter sets. An heuristic algorithm for finding the transition letter sets, and their order, was used to search the example spaces containing each vocabulary item. A remarkable amount of structure was found indicating that there are invariant structural features in the speech data which are reliable enough for use as a basis for recognition.

Three preliminary tests were performed on the transition letter sets. The results of these tests indicate that the search algorithm is quite stable in the presence of extraneous data in the example space, and hence not overly sensitive to the training data; that the transition letter sets are descriptive of speech samples other than those from which they are found, and hence are general enough to form the basis of a recognition procedure; and, finally, that the number of false recognitions triggered by the transition letter sets when used in a very simple recognition algorithm is not large, indicating that they accurately characterize (that is, identify uniquely) the vocabulary items.

Having found the sound groups which reliably occur in samples of each vocabulary item, attention turned to the residual sound groups in the speech data. These data, termed loop letter sets, were seen to be potentially effective in reducing the number of false recognitions. A computer program was implemented for finding the "smallest" collection of loop letter sets which accommodate the example spaces. Surprisingly, the resulting residual sounds were found to occur infrequently, indicating that the transition letter sets contain most of the sounds which make up the entire word. Nevertheless, preliminary tests indicated that the loop letter sets are effective contributors toward reducing the false-alarm rate. Exercised over approximately 350 words, the false recognition rate (using only transition and loop letter sets, excluding time duration data) was 0.045 false recognitions per word per vocabulary item.

Some very preliminary investigations were conducted of the usefulness of time durations of, and intervals between, the transition letter sets as an

additional vehicle for characterizing vocabulary items. The stochastic deviation of any particular recognition from the population of recognitions experienced during training was defined. In theory, many, if not all, false recognitions can be expected to be eliminated on the basis of peculiar timing details revealed by large values of the computed dispersion.

Implementation of this concept was not possible during this study, though a greatly simplified hand check of the concept was performed. The "seven" machine was exercised over 175 words, including 16 bona fide "sevens," and complete separation was obtained. This result is particularly encouraging in view of the many features of the recognition technique which were ignored or weakened to make a hand check possible.

Finally, addressing the problem of starting the mathematical machine (automaton) at the proper time, a nondeterministic finite transducer was described. A new copy of the transducer is started whenever a letter in the first transition letter set is received from the preprocessor. When two or more copies of the underlying machine move into the same state, only one of them is retained. This model has been implemented in a computer program called the Research Machine Exerciser, although results are not yet available.

Conclusions and Recommendations

The mathematical machine approach toward LCSR is specifically tailored to the unique requirements of continuous speech in training system design. The results of this initial effort indicate that, though not without risk, the approach offers high promise for a highly accurate real-time recognition capability. It is recommended that development of this technique continue. Reasonable goals for that development, we feel, is the demonstration of real-time continuous digit recognition for at least one speaker before the end of 1977; and to extend the approach to handle all speakers (through some form of adaptation or "training") by the end of 1978.

REFERENCES

1. B. Atal (Bell Labs), "Automatic Recognition of Speakers from Their Voices," IEEE Proceedings, pp. 460 – 475, April 1976.

2. L. R. Bahl, J. K. Baker, et al. (IBM), "Preliminary Results on the Performance of a System for the Automatic Recognition of Continuous Speech," Conference Record 1976 ASSP Conference, pp. 425 – 429, April 1976.

3. L. R. Bahl, F. Jelinek (IBM), "Decoding for Channels with Insertions, Deletions and Substitutions with Applications to Speech Recognition," IEEE Trans. on Information Theory, pp. 404 – 411, July 1975.

4. J. K. Baker (IBM), "Stochastic Modeling for Automatic Speech Understanding," pp. 521 – 542 in 31.

5. R. W. Christiansen, C. K. Rushforth (Univ. of Utah), "Word Spotting in Continuous Speech Using Linear Predictive Coding," Conference Record 1976 ASSP Conference, pp. 557 – 560, April 1976.

6. P. S. Cohen, R. L. Mercer (IBM), "The Phonological Component of An Automatic Speech-Recognition System," pp. 275 – 320 in 31.

7. C. Cook (BBN), "Word Verification in a Speech Understanding System," Conference Record 1976 ASSP Conference, pp. 553 – 556, April 1976.

8. P. Denes (Bell Labs), "Automatic Speech Recognition: Old and New Ideas," pp. 73 – 82 in 31.

9. N. R. Dixon, H. F. Silverman (IBM), "The 1976 Modular Acoustic Processor (MAP): Diadic Segment Classification and Final Phonemic String Estimation," Conference Record 1976 ASSP Conference, pp. 15 – 20, April 1976.

10. N. R. Dixon, H. F. Silverman (IBM), "The 1976 Modular Acoustic Processor (MAP): Signal Analysis and Phonemic Segmentation," Conference Record 1976 ASSP Conference, pp. 9 – 14, April 1976.

11. F. Hayes-Roth, V. R. Lesser (CMU), "Focus of Attention in a Distributed-Logic Speech Understanding System," Conference Record 1976 ASSP Conference, pp. 416 – 420, April 1976.

12. F. Hayes-Roth, D. S. Mostow (CMU), "Syntax and Semantics in a Distributed Speech Understanding System," Conference Record 1976 ASSP Conference, pp. 421 – 424, April 1976.

13. D. R. Hill (Univ. of Calgary), "Man-Machine Interaction Using Speech," Advances in Computers, vol. II, pp. 165 – 230, 1971.

14. S. R. Hyde (British Post Office), "Automatic Speech Recognition: A Critical Survey and Discussion of the Literature," Human Communication: A Unified View, 399 – 438, Pub 1972 – reports through 1967.

15. F. Jelinek (IBM), "Continuous Speech Recognition by Statistical Methods," IEEE Proceedings, pp. 532 – 556, April 1976.

16. B. T. Lowerre (CMU), "The HARPY Speech Recognition System," Dissertation; Dept. of Computer Science, CMU, April 1976.

17. J. Makhoul (BBN), "Linear Prediction in Automatic Speech Recognition," pp. 183 – 220 in 31.

18. J. Makhoul (BBN, MIT), "Speaker Adaptation in a Limited Speech Recognition System," IEEE Trans. on Computers, pp. 1057 – 1063, Sept 1971.

19. J. D. Markel, A. H. Gray, Jr. (SCRL), "Linear Prediction of Speech," Pub. by Springer-Verlag, 1976.

20. T. B. Martin (Univ. of Penn.), "Acoustic Recognition of a Limited Vocabulary in Continuous Speech," Dissertation, Dept. of Electrical Engineering, Univ. of Penn., 1970.

21. T. B. Martin (TTI), "Practical Applications of Voice Input to Machines," IEEE Proceedings, pp. 487 – 501, April 1976.

22. T. B. Martin, et al. (RCA), "Speech Recognition by Feature Abstraction Techniques," AL-TDR-64-176, Aug 1964.

23. E. P. Neuberg (NSA), "Philosophies of Speech Recognition," pp. 83 – 95 in 31.

24. A. Newell (CMU), "A Tutorial on Speech Understanding Systems," pp. 3 – 54 in 31.

25. B. T. Oshika, V. W. Zue, et al. (SCRL), "The Role of Phonological Rules in Speech Understanding Research," IEEE Trans. on Acoustics, Speech and Signal Processing, pp. 104 – 117, Feb. 1975.

26. K. W. Otten (NCR), "Approaches to the Machine Recognition of Conversational Speech," Advances in Computers, vol. II, pp. 127 – 163, 1971.

27. L. R. Rabiner, M. R. Sambur (Bell Labs), "Speaker Independent Recognition of Connected Digits," Conference Record 1976 ASSP Conference, pp. 202 – 205, April 1976.

28. L. R. Rabiner, R. W. Schafer (Bell Labs and Georgia Tech), "Digital Techniques for Computer Voice Response Implementations and Applications," IEEE Proceedings, pp. 416 – 433, April 1976.

29. D. R. Reddy (CMU), "Speech Recognition by Machine: A Review", IEEE Proceedings, pp. 501 – 531, April 1976.

30. D. R. Reddy, P. J. Vicens (Stanford Univ.), "A Procedure for the Segmentation of Connected Speech," Journal of Audio Eng. Society, pp. 404 – 411, Oct. 1968.

31. D. R. Reddy (ed), "Speech Recognition," 1975.

32. M. R. Sambur, L. R. Rabiner (Bell Labs), "A Speaker-Independent Digit Recognition System," Bell System Technical Journal, pp. 81 – 102, Jan. 1975.

33. R. M. Schwartz, V. W. Zue (BBN), "Acoustic-Phonetic Recognition in BBN Speechlis," Conference Record 1976 ASSP Conference, pp. 21 – 24, April 1976.

34. L. J. Siegel, K. Steiglitz (Princeton Univ.), "A Pattern Classification Algorithm for the Voiced/Unvoiced Decision," Conference Record 1976 ASSP Conference, pp. 326 – 329, April 1976.

35. T. E. Skinner, D. R. Kloker, M. F. Medress (Univac), "A Speech Recognition System for Connected Word Sequences," Conference Record 1976 ASSP Conference, pp. 434 – 437, April 1976.

36. L. R. Tabert, G. F. Groner, et al. (Stanford), "A Real-Time Adaptive Speech Recognition System," ASD-TDR-63-660, May 1963.

37. C. C. Tappert (IBM), "A Markov Model Acoustic Phonetic Component for Automatic Speech Recognition," Conference Record 1976 ASSP Conference, pp. 25 – 28, April 1976.

38. C. C. Tappert, N. R. Dixon, A. S. Rabinowitz (IBM), "Application of Sequential Decoding for Converting Phonetic to Graphic Representation in Automatic Recognition of Continuous Speech," Trans. on Electroacoustics, pp. 225 – 228, Jun. 1973.

39. T. K. Vintsiuk (USSR), "Generative Grammars and Dynamic Programming in Speech Recognition with Learning," Conference Record 1976 ASSP Conference, pp. 446 – 449, April 1976.

40. T. K. Vintsiuk, O. N. Gavrilyuk, A. G. Shinkazh (USSR), "Phoneme-by-Phoneme Recognition of Speech Composed of the Words of Given Vocabulary," Conference Record 1976 ASSP Conference, pp. 450 – 452, April 1976.

41. R. S. Vonusa, L. Jacops, B. Beek (RADC), "An Automatic Isolated Word Recognition System-Study and Evaluation," RADC-TR-70-129, Oct. 1970.

42. C. L. Weinstein, S. S. McCandless, et al. (MIT), "A System for Acoustic-Phonetic Analysis of Continuous Speech," IEEE Trans. on Acoustics, Speech and Signal Processing, pp. 54 – 67, Feb. 1975.

43. G. M. White (Xerox), "Speech Recognition: A Tutorial Overview," Computer, pp. 40 – 53, May 1976.

44. G. M. White, R. B. Neely (Xerox), "Speech Recognition Experiments with Linear Prediction, Bandpass Filtering, and Dynamic Programming," IEEE Trans. on Acoustics, Speech and Signal Processing, pp. 183 – 188, Apr. 1976.

45. W. A. Woods, M. Bates, et al. (BBN), "Uses of Higher Level Knowledge in a Speech Understanding System: A Progress Report," Conference Record 1976 ASSP Conference, pp. 438 – 441, April 1976.

46. B. Yegnanarayana (India), "Effect of Noise and Distortion in Speech on Parametric Extraction," Conference Record 1976 ASSP Conference, pp. 336 – 339, April 1976.

APPENDIX A

THE (REJECTED) EMPIRICAL APPROACH: RECOG

B.1 GENERAL

As the result of earlier research, the RECOG system had been developed to the point that the need for systematic evaluation of various parameters was required. Prior to the LCSR project, the resources needed for such an evaluation (in particular, a data base) were not available. The intent of the RECOG study under the LCSR contract is simply to determine whether or not the technique is promising enough to warrant further investigation. Therefore, the RECOG scheme was incorporated into an executive which utilizes the raw data base described in section VI. Recognition data were gathered for the utterances in table 4 spoken by two male and two female talkers.

TABLE 4. NUMBER SET B.

670	.418	1764	8.2
0529	2875	9003	751
3956	.114	.892	163.
0225	9.3	2740	4.97
.04	3851	50.8	1336
4962	6109	2447	015
721.	3558	126	5.73
4669	237	6084	8320
348	7195	9431	577.
82.6	.542	6880	459
0653	7991	56.	9307

A description of the technique, a summary of results, and some recommendations are provided in the following paragraphs.

B. 2 SEGMENTATION

Visual study of the raw features generated by the VIP-100 for the 10 digits spoken continuously confirmed that these strings contained the characteristic patterns seen in the isolated words. Thus if the beginning and end of each word could be detected, recognition should be possible. The difficulty of establishing these boundaries is well known.

RECOG's approach requires foreknowledge of the number of digits in the utterance. The initial segmentation simply divides the utterance into equal sized segments corresponding to each word spoken. Processing begins with the last digit. Ten upper boundary values are tested in six sample increments, about the proposed boundary. Figure 14 shows the portions of the input tested in a two-word utterance in which 200 2-millisecond samples were collected. The best upper boundary is chosen as the lower boundary for the previous digit, and the process is repeated (except for the first word where the starting boundary is known).

B. 3 SCORING

Determination of the best boundaries is a scoring problem. In this approach, then, the scoring algorithm must do more than simply determine what was spoken – it must also show the subtle differences between scores for the word chosen which are the result of the different boundary choices. It is this discrimination which enables the selection of the best upper boundary. TTI's recognition algorithm was found to be inadequate for this purpose. One of its shortcomings is that the score is directly proportional to the number of bits set in the reference array. This, in turn, is related as a sawtooth-shaped function to the number of repeats. Even when the number