THE VOCAL SPEECH UNDERSTANDING SYSTEM

Stephen E. Levinson
Yale University
New Haven, Connecticut USA

Abstract

This paper describes the VOCAL (Voice Operated CALculator) speech understanding system. VOCAL is a software package that lets Its user program a computer to perform numerical calculations by speaking to It in English-like sentences. To accomplish this, VOCAL uses processes for acoustic, grammatical, and semantic analysis. These individual procedures, which are relatively simple, are embedded in a control structure that uses the information from each component to arrive at a meaningful interpretation of spoken sentences.

One unique feature of VOCAL, which is essential to the development of speech understanding systems, is that it is complete and self-contained. Coded in standard FORTRAN, it is compact enough to run on many minicomputers and can be used in a real-time, on-line environment on slightly more powerful machines.

Testing has shown that despite a correct word identification rate of leas that 60% the VOCAL system usually correctly interprets even very long sentences.

Introduction

This paper describes the present state of the VOCAL speech understanding system. Briefly, VOCAL is a complete, self-contained speech understanding system that has undergone sufficient testing to indicate that it has potential both as a practical voice programming system and as a research tool for the investigation of more general problems of natural, spoken man/machine communication.

VOCAL interprets a spoken utterance as the well formed, meaningful sentence that sounds most like its acoustic transcription of the input according to a quantitative similarity measure and demonstrates its understanding by compiling the input into executable machine language code. To accomplish this, VOCAL uses several relatively simple procedures linked by a control structure- The VOCAL language was designed to be simple and powerful in specifying numerical algorithms. Within the constraints of small vocabulary and simple syntax, the VOCAL language bears a reasonable similarity to the special subset of English used by a mathematician to describe equations he is writing on a black-board. Finally, VOCAL is fast, efficient, and portable.

VOCAL operates on a limited vocabulary, as all speech understanding systems do [1,15,16,20,21,22, 23]. Sentences are spoken by cooperative subjects in quiet rooms. The VOCAL syntax is well defined by a mathematical model and the problem domain is limited to a specialized task so that heuristics for semantic analysis are applicable.

VOCAL does not, however, accept truly continuous speech input. Our data was recorded under ideal conditions and the subjects were required to articulate clearly. Thus VOCAL can reliably locate word boundaries by detecting brief pauses between words. As a result, VOCAL begins its acoustic processing at the lexical rather than the phonemic level.

There are 42 words in the VOCAL lexicon ("to" and "two" are considered the same word at the acoustic level) so that word identification becomes a difficult but tractable pattern recognition problem.

At the acoustic level, VOCAL represents lexical information in the form of a spectrogram. An extremely efficient Fast Fourier Transform (FFT) generates the spectrogram, which contains all the information necessary for word recognition in sufficiently compact form so that the full power of mathematical pattern recognition techniques can be brought to bear. (Very early work by Potter, Kopp, and Kopp [14] and recent verification by Klatt and Stevens [7] demonstrate that a spectrogram can contain all this information, although the more recent trend in acoustic signal processing has been toward digital filtering [17] and predictive coding [8,12,23] for spectral estimation.)

In order to solve the grammatical problem of parsing unreliably transcribed input strings [11, 22], the parser is permitted to insert hypothetical words or strings based on acoustic and semantic information while processing the input.

The difficulty often encountered is providing a speech understanding system with semantic capabilities was partly solved by the choice of problem domain. Cherry [3] defines semantics as the relationship between the symbols of a language and the real objects or concepts for which they stand. For the VOCAL system the objects to which the symbols are related are sets of machine language instructions that cause the computer to perform the operations specified by those symbols. The semantic relationships are those required by an ordinary compiler for the VOCAL language.

The control structure of VOCAL has two important properties. First, information extracted at each level of processing is used by every other level. Second, the actual strategy used by VOCAL to understand a sentence is largely determined by that utterance of that sentence. In this process, no absolute quantitative criteria (e.g. empirically determined confidence levels) are used and no unbounded computations (e.g. exhaustive searches) are performed.

Finally, there is the matter of the cost of speech understanding computations in processing time and resources. Because VOCAL is coded in FORTRAN at a level very close to the machine and uses efficient FFT, pattern recognition, and parsing algorithms, it has only modest core requirements and will run on-line in real time on large minicomputers or medium-size machines. If the spectral analysis is performed by an easily constructed bank of five analog bandpass filters, the system can run on-line in real time even on a small minicomputer. This opens up avenues for testing, evaluation, and development that are closed to larger systems.

The VOCAL System

VOCAL is currently organized into one main program and six subroutines. The main program, called CNTRL, governs (and records for subsequent analysis) all of the operations of the subroutines. The subroutines and their tasks are listed below.

(1) SPGRM generates the spectrogram of single-word utterances.

(2) SWIFFT is a special purpose Fast Fourier Transform algorithm used to compute short duration spectra.

(3) KNN3 is the pattern recognition algorithm. It uses a generalized k-nearest-neighbor scheme with preprocessing [13].

(A) PAUSE is a push-down automaton that not only accepts well formed sentences but also makes hypothetical insertions where the acoustic transcription leads to an ungraramatical string.

(5) SEMANT forms hypotheses about the meaning of a sentence from clues it finds in the acoustic transcription and grammatical structure.

(6) COMPIL generates machine language code corresponding to the final interpreted meaning of input sentences. Unfortunately this routine is machine-dependent, so it now stops short of generating the actual machine language code and merely sets up the internal form representation from which the code can be generated.

Acoustic Processing

SPGRM performs all of the signal processing required to segment the speech along word bounaries and produce a compact representation of the spectrograms of each of the words.

Word boundaries are located by a simple thresholding technique. When the signal drops to a relative minimum and remains at that level for several consecutive samples, a word boundary is assumed.

The first stage in the generation of a spectrogram is the sampling of the short duration spectrum of the speech waveform every 3.2 msec, and storing the results in a matrix S *= (S_{mn}) given by equation (1).

$$(1)\quad S_{mn} = \left| \frac{1}{N} \sum_{k=0}^{N-1} W_k X_{k+(m-1)N}\, e^{\frac{-j2\pi kn}{N}} \right|$$

$$m = 1,2,\ldots,M$$
$$n = 1,2,\ldots,(N-1)/2 = 31$$
$$j = \sqrt{-1}$$

where:

$$\{X_k\} = \begin{cases} \text{contiguous samples of the speech waveform} \\ \quad \text{for } 0 \le k \le 31 \\ 0 \text{ for } 32 \le k \le 63 \end{cases}$$

$\{W_k\}$ are the prolate spheroidal weighting coefficients of Eberhard [4].

The computation of equation (1) continues for increasing m until all the samples within the word boundary are used.

The matrix S defined by equation (1) is then reduced to a more compact form S' = (S'_{mn}) by averaging its entries over predetermined time and frequency intervals according to equation (2).

$$(2)\quad S'_{mn} = \frac{1}{c} \sum_{j=b_{m-1}+1}^{b_m} \sum_{k=t_{n-1}+1}^{t_n} S_{jk} \qquad \begin{matrix} 1 \le m \le 5 \\ 1 \le n \le 10 \end{matrix}$$

where

$$c = [(b_m - b_{m-1}+1)(t_n - t_{n-1}+1)];\ b_0 = t_0 = 1$$

and where $t_i = [iM/10]$ $i = 1,2,\ldots,10$; [*] is the greatest integer function and $\{b_i\}$ = 3,7,12,20,32 This choice of $\{b_i\}$ results in band pass filtering the original speech waveform with a bank of contiguous filters having cutoff frequencies of 450, 900, 1800, 3000, and 5000 hz. The choice of $\{t_i\}$ simply partitions the spectrogram into ten equal time segments.

Finally, the S' matrix is normalized by its largest entry to produce the matrix S" = (S''_{mn}) given by equation (3).

$$(3)\quad S''_{mn} = \begin{cases} 1 \text{ if } S'_{mn} = \max\{S'_{mn}\} \\ \dfrac{S'_{mn}}{\max\{S'_{mn}\}} \text{ otherwise.} \end{cases} \qquad \begin{matrix} 1 \le m \le 5 \\ 1 \le n \le 10 \end{matrix}$$

The result of these transformations is a 5*10 matrix that represents the normalized time and frequency smoothed spectrogram of a single spoken word.

The discrete Fourier Transform (DFT) implicit in equation (1) is actually performed by the special-purpose FFT algorithm of subroutine SWIFFT. The desired speed is achieved in this algorithm by employing all of the following techniques :

1) Real transform of length N computed as a complex transform of length N/2
2) Mixed radix representation of subscripts [2,19]
3) Pruning the first stage to account for the zero padding [9]
4) Elimination of all trivial operations (e.g. multiplication by 0, ±1)
5) In-line coding with no loops and no address calculations
6) No explicit binary sort.

Pattern Recognition

KNN3 is the pattern recognition subroutine. It performs two separate operations, feature selection and nearest neighbor ordering, implicit in which is the identification process.

The feature selection process is accomplished by means of a Karhunen Loeve expansion applied to feature selection problems according to methods described by Meisel [10] and Patrick [13]. Recall that the spectrogram for each word is represented by the 5*10 matrix, S", of equation (3). Let Y be a 50-vector whose components, yi. 1 < i < 50, are the elements *of* S" ordered columnwise. We reduce this vector, Y, to a ten-dimensional vector, X, by the linear transformation T.

(A) X = T Y.

T is a matrix whose rows are the eigenvectors corresponding to the ten largest eigenvalues of an estimate of the covariance matrix U of the training set. That is.

(5) $\hat{U} = (\hat{u}_{ij}) = \frac{1}{M} \Sigma (y_i - \bar{y}_i)(y_j - \bar{y}_j)$

$i,j = 1,2,\ldots,50.$

The $\bar{y}_i$ are estimates of the mean of y_i computed from

(6) $\bar{y}_i = \frac{1}{M} \Sigma y_i \qquad 1 \le i \le 50.$

In equations (5) and (6) the summation is over all M samples in the training set.

Finally we take $\hat{U}$ computed from equation (5) and solve the eigenvalue problem.

(7) $\hat{U}\vec{q} = \lambda_k \vec{q} \qquad k = 1,2,\ldots,10.$

The λ_k are estimates of the variances of the *kth* feature in the direction of the *kth* eigenvector, $\vec{q}_k$, and are the eigenvalues of $\hat{U}$. The matrix T is then expressed as

(8) $T = \begin{vmatrix} \vec{q}_1 \\ \vec{q}_2 \\ \cdot \\ \cdot \\ \vec{q}_{10} \end{vmatrix}$ where $\vec{q}_k$ is the eigenvector corresponding to the *kth* largest eigenvalue $1 \le k \le 10$.

Of course, the computations described by equations (5), (6), (7), and (8) are done only once in an off-line step and the matrix T is stored in KNN3, which simply uses the transformation according to equation (4) to get a ten-feature representation of a spectrogram.

An examination of the eigenvalues of $\hat{U}$ from equation (7) reveals that the intrinsic dimensionality of the feature space is between 6 and 16. That is, λ_k becomes very small for $k > 16$. But in order to keep the implicit estimates of the probability distribution function of the feature values consistent with the number of samples in the training data, we restricted ourselves to a ten-dimensional approximation.

The next function of KNN3 is to use the vector $\vec{X}$ from equation (4) in the generalized K Nearest Neighbor decision rule of Patrick [13] to classify an utterance.

(9a) $f(\vec{X}) = n$

$$\Longleftrightarrow \sum_{k=1}^{K} \sum_{j=1}^{10} (X_j - X_j^{(n(k))})^2$$

$$\le \sum_{k=1}^{K} \sum_{\ell=1}^{10} (X_\ell - X_\ell^{(m(k))})^2 \quad \text{for all } m$$

where n is the class index, x_j is the *jth* component of the vector $\vec{X}$, $j = 1,2,\ldots,10$, and $x_j^{n(k)}$ is the *jth* component of the vector of the *kth* nearest neighbor to $\vec{X}$ in the *nth* class. We have fixed $k = 7$ and $1 \le n \le 42$.

The decision rule of equation (9) simply says assign the vector $\vec{X}$ to class n if and only if the total distance (in the Euclidean sense) from $\vec{X}$ to its k nearest neighbors in the *nth* class is less than its total distance to its k nearest neighbors in any other class.

Ln order to use the decision rule of equation (9a) a set of discriminant functions $\{d_n(\vec{X})\}$ $1 \le n \le 42$ are evaluated where

(9b) $d_n(\vec{X}) = \sum_{k=1}^{k} \sum_{j=1}^{10} (X_j - X_i^{(n(k))})^2.$

Then the $\{d_n(\vec{X})\}$ are sorted in ascending order so that

(9c) $d_{n_1}(\vec{X}) \le d_{n_2}(\vec{X}) \le \ldots \le d_{n_{42}}(\vec{X}).$

The significance of the ordering (9c) is that $\vec{X}$ is classified as nj, but n_2 *is* the next most likely candidate, followed by n3, etc. This information is stored for later use by PARSE and SEMANT.

The naive implementation of this algorithm requires one distance calculation for each training sample. But carefully reordering samples within each class and storing their positions on each axis can eliminate many distance calculations because it is known in advance that they cannot be candidates for nearest neighbors. This preprocessing technique outlined by Shustek et al. [18] greatly reduces the computation cost of implementing equation (9a).

Syntax

PARSE is the subroutine that performs the necessary analysis for understanding a sentence. PARSE also serves as a "front end" for the final compilation of subroutine COMPIL.

VOCAL has 43 symbols and eleven different sentence schemata, most of which have two forms. The first form permits the use of line numbers, which COMPIL treats as instruction addresses. The second form is a "stand alone" form in which each sentence is a complete program unto itself. These command structures are of three types, executive (e.g. start, stop, edit), minor utility (e.g. unconditional branching, subroutine linking, input), and major utility (e.g. conditional branching, assignment, indexing). The third are the real work horses of the language and contain arithmetic expressions composed of variables that may or may not be subscripted, floating-point constants, and all the usual arithmetic operators, delimiters, and elementary functions.

The VOCAL language L(G) is generated by a context-free grammar, $G(V_n, V_t, S, P)$, where

V_n is a set of non-terminal symbols
V_t is the set of terminal symbols (i.e. the VOCAL alphabet)
S is a start symbol $S \in V_n$
P is a set of production rules of the form

(10) $p: a \rightarrow Ab$

for all $p \in P$, $a \in V_n$, $A \in V_t$, $b \in V_n^*$

where V* implies the set of all strings of elements of V.

There are 192 productions of the form of equation (10) in the VOCAL language so that the language is specified in the Greibach normal form [5]. In addition, we have the constraint

(11) for every $p_1, p_2 \in P$

there exists no $p_1: a \rightarrow A\alpha \wedge p_2: a \rightarrow B\beta$.

$A,B \in V_t;\ \alpha,\beta \in V_n^*$.

Equation (11) insures that the grammar

$G(V_n, V_t, S, P)$ is deterministic and can therefore be accepted by a deterministic push-down automaton. $M(V_t, V_n, S, \delta)$ where

V_t is the VOCAL alphabet
V_n is a set of stack symbols
S is the initial stack symbol
δ is a transition function $\mid \delta: (V_t \times V_n) \rightarrow V_n^*$

so that whenever $p: a \rightarrow A\alpha$, there is an entry in δ of the form:

(12) $\delta(A,a) = \alpha$.

Equation (12) then defines the operation of M as follows. On scanning input symbol A with non-terminal symbol a on top of the stack, accept A and replace a by the string a. If the stack is empty after the input sentence has been scanned, it is a well formed sentence in L(G). This operation is the heart of PARSE.

The transition function of equation (12) is stored in the form of a table with two levels of pointers. One is for internal use and insures that the table will be scanned efficiently according to an algorithm described by Gries [6], The second level provides a link to SEMANT, which we shall describe shortly.

In addition to determining whether a sentence is well formed, PARSE is also capable of inserting words according to the ordering of (9c) in a sentence to replace grammatically incorrect ones. This replacement routine involves a link to the acoustic and semantic levels and will therefore be discussed when we consider the control structure.

Semantic Analysis

The function of SEMANT is to determine the meaning (in terms of what COMPIL can understand) of well formed sentences. SEMANT generates these hypotheses on the basis of acoustic and grammatical clues plus some tabulated knowledge of its own.

SEMANT treats phrases and sentences as meaningful if COMPIL can either make appropriate entries in its symbol table (e.g. data or instruction addresses) or can generate code (e.g. to compute the value of an arithmetic expression). For example, the word "seven" by itself is semantically meaningless but the phrase "X seven" is a variable for which COMPIL would assign a storage location. Or the phrase "three point seven" could be a line number for which COMPIL would assign an entry address. SEMANT scans the input strings for clues and uses acoustic information to form the best phrase (the smallest metric computed from equation (9b)) having the assumed meaning. Information concerning the grammatical structure of the phrase is contained in the transition function of equation (12). The appropriate set of production rules are located by the second level of pointers in PARSE described above. For example, the words "three" and "one" in the phrase

three (?) one (?) (?)

would serve as a clue to SEMANT that a floating-point constant was present. SEMANT would then go to the transition table and look up the production rules for the formation of floating-point numbers. Then, referring to the values of the distance function computed by KNN3 according to equation (9b), the best such phrase would be formed and the result might be

three point one four one.

SEMANT forms four hypothetical command structures (since there are four major utility Instructions). The arguments of the command structure are resolved (e.g. variable names, line numbers), and finally arithmetic expressions are determined including proper parenthesizing. If the chosen command structure cannot contain an arithmetic expression, of course, SEMANT does not look for clues for one.

COMPIL is not a necessary part of the speech understanding process but rather provides a means by which VOCAL can demonstrate that It has indeed understood a sentence. Unfortunately a subroutine that produces machine language code is necessarily machine-dependent. For that reason, COMPIL has been purposely left unfinished. In its present state, COMPIL simply puts a sentence in a convenient internal form from which the appropriate subroutines could generate the object code.

Control Structure

The organization of VOCAL is shown in figure 1. In this diagram the single lines represent logic paths and the double lines, data paths. The required control of its operation is performed by the main program, CNTRL, and may be described as follows.

Digitized speech is processed to produce the nearest neighbor ordering for each word in the sentence according to equations (9b) and (9c). These results are stored for later use.

The sentence is then parsed. If the acoustic transcription of the sentence is grammatically well formed then all processing ends since no better interpretation for the sentence can be found.

Most often, however, misclasslficatlons do occur, and they result in ungrammatical strings. In this case the parser replaces the incorrect word by other words in the vocabulary In the ordering (9c) for the offending word. The values of the discriminant functions of (9b) are added up as the sentence is parsed. When the entire sentence has been scanned this total is a measure of the similarity of the modified string to the acoustic transcription. For example, let the acoustic transcription of the sentence H_0 be represented by $V_{1,n_1}, V_{2,n_1}, \ldots, V_{p,n_1}$ where each $V_{i,j}$ is a member of V_t. Suppose that after this string was parsed it had been modified to $H_1 = V_{1,n_{k_1}}, V_{2,n_{k_2}}, \ldots, V_{p,n_{k_p}}$. Then the similarity of H_0 to H_1 is given by:

(13) $$\rho(H_0, H_1) = \sum_{i=1}^{p} [d(\vec{X}_{i,n_{k_i}}) - d(\vec{X}_{i,n_i})]$$

where $d(\vec{X}_{i,n_1})$ is the value of the first discriminant function in the ordering (9c) for the *ith* word, V_{i,n_1}, in H_0 and $d(\vec{X}_{i,n_{k_i}})$ is the value of

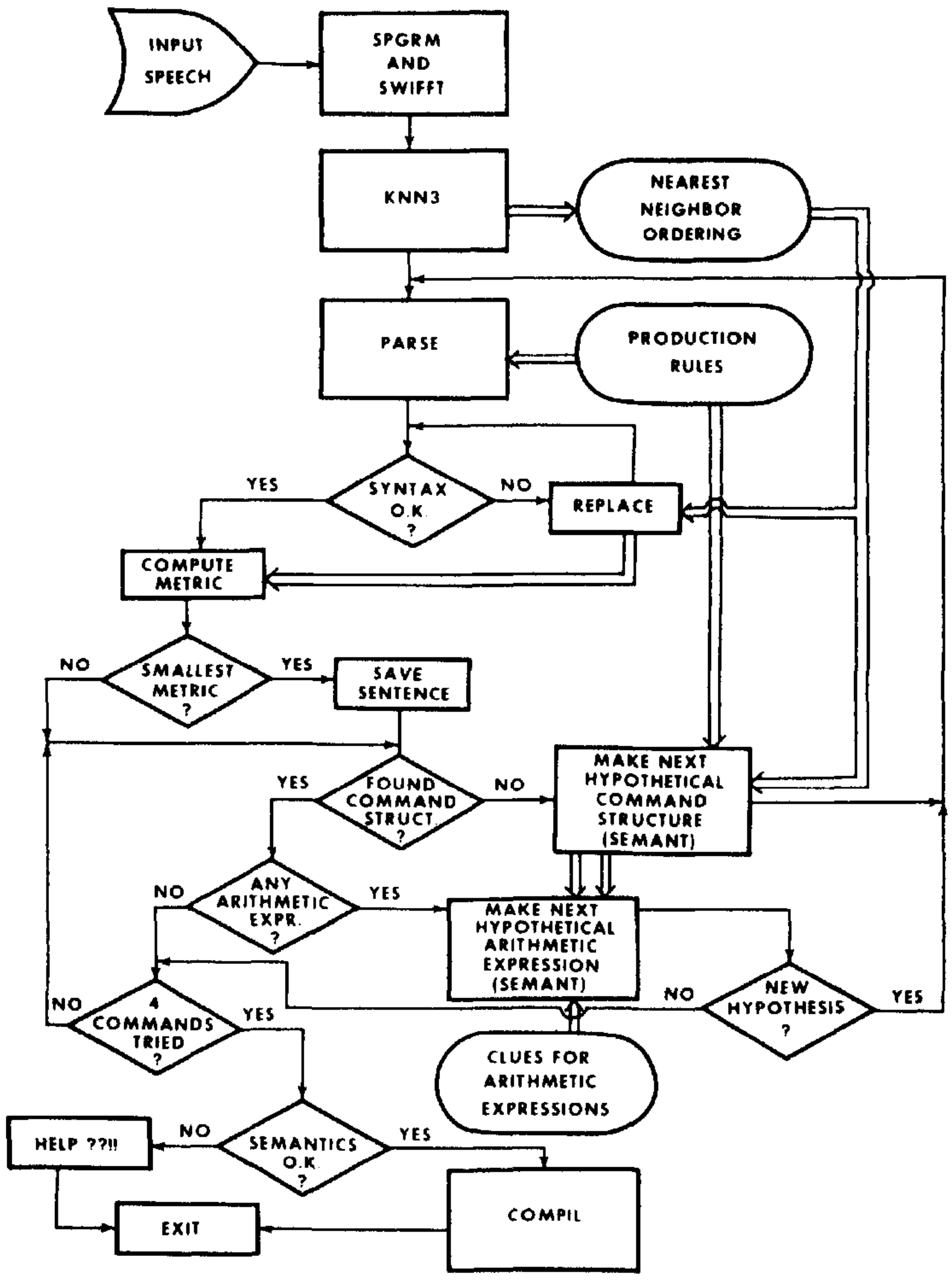

FIGURE 1 FLOW CHART OF THE VOCAL SYSTEM

the $n_k th$ discriminant function for the corresponding word, $V_{i,n_{k_i}}$, in H_1.

After the acoustic transcription has been processed, PARSE only operates on sentences that have been generated by SEMANT. First SEMANT chooses the best (in the sense of equation (13)) command structure. Then it builds a sentence around this command structure by replacing words or groups of words with well formed, meaningful strings. Replacements are always made in the order of (9c) and a total of the replacement similarities is kept. The resulting string, H_2, is returned to PARSE for syntactic analysis and then sent back to SEMANT for refinement. This loop is executed until SEMANT cannot offer any more information ($i.e.$ $H_i = H_{i-1}$). Then the next command structure is tried and the process is repeated. The best four command structures are tried in this way.

When PARSE is called to analyze the ith hypothesis, H_i, it computes $P(H,H_i)$ according to equation (13). At the ith iteration, $\rho(H_0,H_i)$ is compared to (H_0,H_{i-1}); the sentence correspond!ng to the smaller metrie is retained and the smaller metric is stored for comparison at the i+lst stage.

When this process terminates, the well formed sentence corresponding to the smallest metric is compiled and executed. If no such sentence is found, VOCAL asks for help, indicating which parts ot the sentence it finds confusing.

In the course of the understanding procedure VOCAL may generate and test twenty hypotheses for a short sentence or fifty for a long, complicated one. In theory, there is no limit to the number of times it may try to refine a hypothesis. In practice, however, the maximum observed has been twelve.

Test Data and Results

Testing of VOCAL thus far has been on a data set of 1266 words divided into a training set of 88? words and a test set of 22 sentences containing a total of 384 words. The training set con--sts of 21 spoken samples of each of the 42 words in the VOCAL lexicon. The test set sentences are spoken by eleven different speakers, four of whom were not subjects for the training set.

The speed) was recorded in an anechoic room on a NACRA IV scientific recorder on low-noise tape at 15 ips. The analog signals were then low pass filtered at 5 Khz, sampled to 10 bits at 10 Khz, and stored on magnetic tape.

The training set was, of course, used to compute T of equation (4) and then the transformed, labeled training samples and T were stored in KNN3.

The acoustic recognition score on the test set was 210 correct identifications out of 384 words. This resulted in transcribing two sentences correctly. The number of correct Identifications rose to J17 for the full VOCAL system. This resulted in an exactly correct understanding of eight sentences, a nearly correct understanding of thirteen more, and a poor interpretation of one sentence. The meaning of "nearly correct" and "poor" will be clear from the examples below. In each, the actual sentence appears first, followed by its acoustic transcription and then by its final interpretation. In the first example, the understanding is perfect; in the second, nearly correct; and in the third, poor.

Actual sentence:
One point three read X two two next.

Acoustic transcription;
One close three read X two two scratch.

Perceived sentence:
One point three read X two two next.

Actual sentence:
Let X one five sub three three equal term minus X one times E two point oh close close over term sine X two close plus cosine X two sub three close close to the two point seven eight one eight three five next.

Acoustic transcription:
Scratch X one five scratch three three equal sine minus X one times E two point oh close point over term sine X two close log cosine X two scratch three point oh to the two close sine three one eight three five read.

Perceived sentence;
Let X one five sub three three equal sine minus X one times E two point oh close close over term sine X two close plus cosine X two sub three close close to the two point one three one eight three flve next.

Actual sentence:
Four point five six step X seven from three point oh to eight point one by sine three of point oh close let X nine equal arctan X eight sub seven close next.

Acoustic transcription:
Four point by eight from X sine oh begin close oh equal eight point norm five norm three oh point oh log log X log equal arctan X link scratch close log norm.

Perceiih'd sentence:
Four point five eight let X one of) sub one oh equal eight point one flve one three oh one oh nine plus X nine to the arctan X eight sub seven close next.

The 22 sentences of the test set represent about six minutes of speech. On Yale University's IBM 370/158 computer, just under six minutes was required for VOCAL to process them.

References

[1] J. Barnett. A vocal data management system. IEEE Transactions on Audio and Electroacoustics AU-21(3), 1973.

[2] C. D. Bergland. A guided tour of the Fast Fourier Transform. IEEE Spectrum 6(7), 1969.

[3] C. Cherry. On Human Communication. MIT Press, 1968.

[4] A. Eberhard. An optimal discrete window for the calculation of power spectra. IEEE Transactions on Audio and Electroacoustics AU-2K1), 1973.

[5] S. A. Greibach. A new normal form theorem for context free phrase structure grammars. JACM 12(1), 1965.

[6] D. Gries. Compiler Construction for Digital

Computers. John Wiley & Sons, 1971.

[7] D. H. Klatt & K. N. Stevens. On the automatic recognition of continuous speech: implications from a spectrogram reading experiment. IEEE Transactions on Audio and Electroacoustics AU-2K3), 1973.

[8] J. Makhoul. Spectral analysis of speech by linear prediction. IEEE Transactions on Audio and Electroacoustics AU-21(3), 1973.

[9] J. D. Markel. FFT pruning. IEEE Transactions on Audio and Electroacoustics AU-19(4), 1971.

[10] W. S. Meisel. Computer Oriented Approaches to Pattern Recognition. Academic Press, 1972.

[11] P. L. Miller. A locally organized parser for spoken input. CACM 17(11), 1974.

[12] A. Newell et al. Speech Understanding Systems. American Elsevier, 1973.

[13] E. A. Patrick. Fundamentals of Pattern Recognition. Prentice-Hall, 1972.

[14] R. K. Potter, C. A. Kopp, & H. G. Kopp. Visible Speech. Dover, 1966.

[15] R. D. Reddy, L. D. Erman, & R. B. Neely. A model and a system for machine recognition of speech. IEEE Transactions on Audio and Electroacoustics AU-21(3), 1970.

[16] R. D. Reddy, L. D. Erman, R. D. Fennell, & R. B. Neely. The Hearsay speech understanding system; an example of the recognition process. 3rd IJCAI, Stanford University, 1973.

[17] R. W. Schafer and L. R. Rablner. Design of digital fliter banks for speech analysis. BSTJ 50(10), 1971.

[18] 1.. J. Shustek, F. Baskett, & J. H. Friedman. A relatively efficient algorithm for finding nearest neighbors. SLAC-PUB-1448, 1974.

[19] R. C. Singleton. An algorithm for computing the mixed radix Fast Fourier Transform. IEEE Transactions on Audio and Electroacoustics AU-17(3), 1969.

[20] R. B. Thosar. Recognition of continuous speech: segmentation and classification using signature table adaptation. Stanford Artificial Intelligence Laboratory Memo AIM-213, 19 73.

[21] D. E. Walker. Speech understanding through syntactic and semantic analyses. 3rd IJCAI, Stanford University, 1973.

[22] W. A. Woods & J. Makhoul. Mechanical inference problems in continuous speech understanding. 3rd IJCAI, Stanford University, 1973.

[23] W. A. Woods et al. Speech understanding research: collected papers. BBN Report 2856, 1974.