PROGRAMMING A COMPUTER USING A CONTINUOUS SPOKEN LANGUAGE

by
Edward Strasbourger

Technical Report No. 28
November, 1973
Continuous speech can be represented in a compact digital form by using the low order terms of the analytic cepstrum with the voicing period (if any) as gleaned from cepstrum higher order peaking. The concept of the analytic cepstrum is introduced and discussed. Using this cepstrum representation a phoneme pattern recognizer has been constructed. A spoken programming language using continuous speech has been designed, and a system based on the phoneme recognizer has been implemented to recognize the language. On two trained speakers the resulting recognition rate is 50% to 75% on sentences and 75% to 95% on words. With the interactive reject and repeat feature, the recognition rate is 100%.
1. THE DREAM

Spoken conversation with a computer has long been a goal of computer scientists. Achieving this goal requires the solution of problems in pattern recognition and artificial intelligence. Recognition of inherently noisy data must be accomplished with extremely variable and unreliable prototype patterns. Means must be developed to cope (as the ear does naturally) with unrelated sounds and with variances in voices and in pronunciation.

2. CONVERSATIONAL CONTINUOUS SPEECH SYSTEMS

It is fruitful to investigate continuous spoken language as a medium of communication between man and machine. In 1969, Vicens [10] constructed a conversational system which recognizes continuously spoken sentences of a control language for a mechanical arm-hand picking up blocks off a table. Using a vocabulary of 14 words plus buzz words, 74 semantically distinct commands can be recognized. A sample command is, "Pick up the small block standing at the top right corner".

Many, more powerful systems are being developed. Only the Vicens-Reddy system currently operates in on-line real-time conversational mode.

3. MEANWHILE IN SEATTLE

Independently, at the University of Washington, a conversational system which recognizes a continuously spoken programming language has been developed [9]. This SPOken COMputing Language (SPOCOL) has the flavor of FORTRAN and can distinguish an indefinitely large number of semantically distinct statements spoken from
Edward Strasbourger

a vocabulary of 26 words. For comparison purposes, the system specifications are presented in figure 1, tabulated as in the ARPA speech understanding report [5]. SPOCOL syntax is shown in figure 2. The semantics is illustrated in figure 3 (which lists examples of the different types of SPOCOL constructions along with their roughly equivalent FORTRAN translations) and in figure 4 (a sample SPOCOL program). This language was designed to be a natural one in which to speak a program (much as FORTRAN is a natural one in which to write a program) and at the same time easy for the system to decode.

4. THE HARDWARE

A conversational system was constructed to process speech on-line, so that subtle differences of system response to variations in accent, dialect or context could be detected and tested. The hardware includes a crystal microphone, an amplifier, an analog-to-digital converter, a computer, and a memory scope display with keyboard. Intermediate speech representation in graphical form (presented optionally) provide further insight. As a statement is spoken into the microphone, the speech signal is digitized into 8-bit samples at 8000 samples per second. The statement is decoded and displayed along with the phoneme string and other optional representations (such as: voiceprint or loudness curve). System performance may be varied by using the keyboard to alter parameters used by the recognizer.
1. accepts continuous speech
2. from many
3. cooperative speakers
4. in a noisy computer room
5. over an inexpensive crystal microphone
6. with some system tuning per speaker
7. and some user training
8. with a vocabulary of 26 carefully preselected words
9. with moderate syntactic support provided by a highly artificial language
10. and moderate semantic support provided by a simple programming language (a focused task)
11. with no model or knowledge of the user
12. with spoken conversation interaction (for error correction)
13. allowing error rate in final semantic expression of up to 25% (75% of the sentences correct and the remainder correctable)
14. in less than ten times real time
15. using a dedicated small computer with speed of 500,000 instruction per second
16. and 500,000 bits of random access memory
17. with a simple program
18. at a cost of about 5¢ per second of speech
19. operational in June 1972

Figure 1. Specifications of the SPOCOL system
Edward Strasbourger

\[ <\text{digit}> ::= \text{ONE}|\text{TWO}|\text{THREE}|\text{FOUR}|\text{FIVE}|\text{SIX}|\text{SEVEN}|\text{EIGHT}|\text{NINE}|\text{ZERO} \]
\[ <\text{constant}> ::= <\text{digit}>^+ \]
\[ <\text{variable}> ::= (\text{NUMBER}<\text{constant}>)^+ \]
\[ <\text{factor}> ::= <\text{constant}>|<\text{variable}> \]
\[ <\text{term}> ::= <\text{factor}>\left(\begin{array}{c}
\text{PLUS} \\
\text{MINUS} \\
\text{STAR} \\
\text{SLASH} \\
\end{array}\right) <\text{factor}>^* \]
\[ <\text{expression}> ::= <\text{variable}>\text{EQUALS}<\text{term}> \]
\[ <\text{label}> ::= \text{SPOT}<\text{constant}> \]
\[ <\text{transfer}> ::= \text{NEXT}<\text{label}> \]
\[ <\text{iteration}> ::= \text{LOOP}<\text{expression}> \]
\[ <\text{condition}> ::= \text{SHOULD}<\text{expression}> \text{MINUS} \text{ZERO} \]
\[ <\text{end of routine}> ::= \text{STOP} \]
\[ <\text{name of routine}> ::= <\text{variable}> \]
\[ <\text{call}> ::= \text{PASS}<\text{name of routine}> \]
\[ <\text{edit out last sentence}> ::= \text{SCRATCH} \]
\[ <\text{input}> ::= \text{TAKE}<\text{variable}> \]
\[ <\text{output}> ::= \text{FETCH}<\text{variable}> \]
\[ <\text{statement}> ::= <\text{expression}>|<\text{label}>|<\text{transfer}>|<\text{iteration}>|<\text{condition}>|<\text{call}>|<\text{input}>|<\text{output}> \]
\[ <\text{routine}> ::= <\text{name of routine}><\text{statement}>^+<\text{end of routine}> \]
\[ <\text{program}> ::= <\text{routine}>^+ \]

Figure 2. Syntax of SPOCOL using modified Backus Naur Form.

Words in upper case are as spoken.
Words in lower case surrounded by < > are defined above.
-- means silence of at least 1/3 of a second.
::= means defined as.
| means or.
+ means taken one or more times.
* means taken zero or more times.
Brackets mean a choice of one of the included words.
NUMBER ONE EQUALS NUMBER ONE PLUS ONE
SPOT SIX
NEXT SPOT SIX
* LOOP NUMBER TWO EQUALS FIVE ZERO
SHOULD NUMBER THREE BE MINUS
SHOULD NUMBER FOUR EQUALS FIVE
PASS NUMBER TWO ONE
TAKE NUMBER EIGHT NINE
FETCH NUMBER SEVEN
NUMBER SIX NUMBER THREE EQUALS ONE
SHOULD NUMBER TWO NUMBER SIX NUMBER ONE BE PLUS
TAKE NUMBER ONE NUMBER TWO ONE NUMBER FOUR
PASS NUMBER SIX NUMBER FIVE NUMBER SEVEN
STOP
NUMBER SIX NUMBER ONE TWO NUMBER NINE

\[ N_1 = N_1 + 1 \]

6 CONTINUE
GO TO 6

DO 2 N2 = 1,50
IF (N3.GE.0) GO TO 3
IF (N4.NE.5) GO TO 4
CALL N21
INPUT N89
OUTPUT N7
N6(N3) = 1
IF [N2(N6,N1).LE.0] GO TO 2
INPUT N1(N21,N4)
CALL N6(N5,N7)
STOP or RETURN

SUBROUTINE N6(N12,N9)

Figure 3. SPOCOL statements with roughly equivalent FORTRAN translations.

* Note the double use of the first constant in LOOP and SHOULD statements. The constant is both part of a variable (the indexing variable in the LOOP "block"; the comparison variable in the SHOULD "block") and a pointer to the end location of the LOOP or SHOULD block. Figure 4 has examples of this in context.
NUMBER ONE ZERO
TAKE NUMBER ONE.
SHOULD NUMBER ONE BE MINUS.
NEXT SPOT TWO ZERO.
SPOT ONE.
SHOULD NUMBER ONE EQUALS ZERO.
NEXT SPOT TWO ZERO.
SPOT ONE.
NUMBER FOUR EQUALS ZERO.
NUMBER FIVE EQUALS ZERO.
LOOP NUMBER TWO EQUALS NUMBER ONE.
TAKE NUMBER THREE.
NUMBER FIVE EQUALS NUMBER FIVE PLUS NUMBER THREE.
NUMBER THREE EQUALS NUMBER THREE STAR NUMBER THREE.
NUMBER FOUR EQUALS NUMBER FOUR PLUS NUMBER THREE.
SPOT TWO.
NUMBER NINE EQUALS NUMBER FIVE SLASH NUMBER ONE.
NUMBER SIX EQUALS TWO STAR NUMBER FIVE STAR NUMBER NINE.
NUMBER NINE EQUALS NUMBER NINE STAR NUMBER NINE.
NUMBER SEVEN EQUALS NUMBER FOUR MINUS NUMBER SIX PLUS NUMBER NINE.
NUMBER EIGHT EQUALS NUMBER SEVEN SLASH NUMBER ONE.
FETCH NUMBER EIGHT.
SPOT TWO ZERO.
STOP.

Figure 4. A SPOCOL program to calculate the variance of a sequence of numbers (containing an error).
5. THE SOFTWARE

There are two phases in the recognition of a spoken statement: the reduction of the digitized speech signal into a phoneme string, and the conversion of the phoneme string into a syntactically correct word string. The reduction phase is further refined into three subphases: encoding the digitized signal into a sequence of patterns, matching each pattern against prototype patterns for the various phonemes, and condensing the sequence of best matches into a string of phonemes. The encoded speech pattern consists of the lowest 9 orders of the analytic cepstrum, together with a voiced/unvoiced flag which is set if a peak exists in higher cepstral orders. This encoding will be elaborated on in the following sections.

6. CEPSTRUM CONCEPTS

The cepstrum [2] (the spectrum of the logarithm of the spectrum) has been used effectively in speech processing for voicing detection [6] and for bandwidth compression [7] (speech encoding for resynthesis). Varying definitions of the cepstrum \( \hat{s}(t) \) (as a function of time or period or quefrency) have been used: the fully reversible complex cepstrum \( \hat{s}_C(t) \), or the (simple) cepstrum of Oppenheim \( \hat{s}_O(t) \), of Noll \( \hat{s}_N(t) \), or of Bogert \( \hat{s}_B(t) \). Let the Fourier transform of speech signal \( s(t) \) be denoted by \( S(\omega) = F[s(t)] \), and the (point by point) logarithm of the Fourier transform be denoted by \( \hat{S}(\omega) = \log S(\omega) = \log|S(\omega)| + i\theta(\omega) \).

Then

\[
\hat{s}_C(t) = F^{-1}[\hat{S}(\omega)] = F^{-1}[\log|S(\omega)|] = F^{-1}[\log|S(\omega)| + i\theta(\omega)],
\]

\[
\hat{s}_O(t) = F^{-1}[\text{Re}(\hat{S}(\omega))] = F^{-1}[\log|S(\omega)|],
\]

\[
\hat{s}_N(t) = |F^{-1}[2\text{Re}(\hat{S}(\omega))]|^2 = |F^{-1}[\log|S(\omega)|]^2| = 4(F^{-1}[\log|S(\omega)|])^2
\]

(the power spectrum of the log power spectrum),
and

$$\hat{s}_B(t) = |F^{-1}[2\Re(\hat{S}(\omega))]|^2 = \frac{1}{2} |F^{-1}[\log|S(\omega)|^2]|^2$$

(4)

where $\omega > 0$ or possibly $S(\omega) = 0$ for $\omega < 0$.

(the power spectrum of the one-sided log power spectrum eliminating negative frequencies).

7. THE ANALYTIC CEPSTRUM

Another useful cepstrum definition is the analytic cepstrum $\hat{s}_A(t)$, the complex spectrum of the one-sided log power spectrum:

$$\hat{s}_A(t) = F^{-1}[2\Re(\hat{S}(\omega))] = F^{-1}[\log|S(\omega)|^2]$$

(5)

where $\omega > 0$ or where $S(\omega) = 0$ for $\omega < 0$.

The real part of the analytic cepstrum (the cosine transform of the one-sided log power spectrum) is the Hilbert transform of the imaginary part of the analytic cepstrum (the sine transform of the one-sided log power spectrum) and vice-versa. The two parts will be referred to as the real cepstrum and the imaginary cepstrum respectively. The real cepstrum is simply double the Oppenheim cepstrum:

$$\Re[\hat{s}_A(t)] = \Re(F^{-1}[\log|S(\omega)|^2]) = 2\delta_0(t).$$

(6)

8. WHY THE ANALYTIC CEPSTRUM

Although the imaginary cepstrum can be calculated from the real cepstrum (by doing the Hilbert transform), information about both voicing and formant encoded phonemic typing that is not obvious in the real cepstrum can be observed in the imaginary cepstrum. An example of such additional voicing information is seen in figure 8, where a "doublet" appears at the voicing period in the imaginary cepstrum while no peak is visible in the real cepstrum. Contrast this with figure 5 having the voicing peak in the real cepstrum only, and with figure 6
Figure 5. A photograph of the CRT display screen showing a 32 millisecond segment of the sound EE from a male voice digitized at 8000 samples per second - including from top to bottom: the digitized signal, the log spectrum (magnitude curve with superimposed short pass liftered curve and voiceprint threshold line, and phase curve, and the analytic cepstrum (real curve and imaginary curve with lifting cutoff quefrecny of 2.5 milliseconds).
Figure 6. A negative photograph of the CRT display screen showing a 32 millisecond segment of the sound EE from a female voice digitized at 8000 samples per second - including from top to bottom: the digitized signal, the log spectrum (magnitude curve with superimposed short pass liftered curve and voiceprint threshold line, and phase curve), and the analytic cepstrum (real curve and imaginary curve with liftering cutoff quefrency of 2.5 milliseconds).
Figure 7. A negative photograph of the CRT display screen showing a 32 millisecond segment of the sound SH from a male voice digitized at 8000 samples per second - including from top to bottom: the digitized signal, the log spectrum (magnitude curve with superimposed short pass liftered curve and voiceprint threshold line, and phase curve), and the analytic cepstrum (real curve and imaginary curve with liftering cutoff quefrency of 2.5 milliseconds).
Figure 8. A negative photograph of the CRT display screen showing a 16 millisecond segment of the sound EE from a female voice digitized at 16000 samples per second — including from top to bottom: the digitized signal, the log spectrum (magnitude curve with superimposed short pass liftered curve and voiceprint threshold line, and phase curve), and the analytic cepstrum (real curve and imaginary curve with liftering cutoff quefrency of 2.5 milliseconds).
having both a peak and a doublet.

A good idea of the relative amounts of high frequencies and low frequencies present in a given phoneme is given by the first order of the imaginary cepstrum (i.e. the approximation to \( \hat{S}(\omega) \) contributed by the single sinusoid of figure 9a). This is very similar to the Hiss and Humph in Hill's ESOTerIC system [4]. The first order of the real cepstrum (i.e. the approximation to \( \hat{S}(\omega) \) contributed by the single cosinusoid of figure 9b) indicates the relative amounts of high and low frequencies compared to middle frequencies. Thus the SH sound with a general spectral shape of figure 9c consistently has a high first imaginary cepstral order (e.g. see figure 7) and the EE sound with a general spectral shape of figure 9d has a consistent high first real cepstral order (e.g. see figures 5 and 6).

9. SPEECH RECOGNITION PROCEDURE

Speech is encoded into a sequence of low order cepstral patterns by shifting a 32 millisecond wide Hamming (cosine weighting) window along the digitized signal in steps of 16 milliseconds. At each step the weighted signal is transformed (by Fast Fourier Transform, then logarithm, then inverse FFT) into a 16 millisecond analytic cepstrum. If any point (order) of either the real or the imaginary cepstrum at a quefrency above 2 milliseconds exceeds a threshold, then the speech segment is flagged as voiced. Spurious isolated voiced or unvoiced segments are eliminated by not changing the voicing flag until two consecutive segments agree on the voicing decision. Silence or the silent phoneme (such as that which occurs during a t, p, or k) is determined by comparing the zeroth order of the real cepstrum (called the pseudo-loudness) with a threshold. This silence threshold is set high enough to eliminate the room noise as silence. Pseudo-loudness is also used for syllabification. Boundaries between syllables are chosen as the major valleys of the pseudo-loudness. This syllable
Figure 9a. Sinusoidal shape of a log spectrum $\hat{S}(\omega)$ having a cepstrum with positive imaginary first order and all other orders equal zero.

Figure 9b. Cosinusoidal shape of a log spectrum $\hat{S}(\omega)$ having a cepstrum with positive real first order and all other orders equal zero.

Figure 9c. General shape of the log spectrum $\hat{S}(\omega)$ for the sound SH.

Figure 9d. General shape of the log spectrum $\hat{S}(\omega)$ for the sound EE.
definition is nonstandard in that the word STOP may have 3 or 2 syllables depending on whether the P is or is not pronounced, and the word SEVEN may have either one or two syllables depending on pronunciation.

For nonsilent segments the lowest 9 complex orders of the analytic cepstrum (not counting the zeroth order) are then matched for least mean square distance with the set of prototype phoneme cepstral patterns having the same voicing flag. The 24 prototype patterns are: EE (as in foot), I (sit), E (set), AY (day), AE (at), OH (old), AH (father), UH (up), OO (book), AW (draw), UU (loop), L (loop), R (red or worm), M (mom), N (no), NG (sing), Z (zoo), V (voice), ZH (leisure), DH (the), F (fee), SH (she), S (see), and TH (thick). All 24 are continuants which may be replaced by a new speaker reading through the list. SH, S, F, and TH are unvoiced. The remaining 20 are voiced. The segment is thus tagged either as silent or as a sample of the closest prototype phoneme of the same voicing. An example phoneme string resulting from the spoken work STOP is . - - - - TH S S S S S S S S S - - - - . TH TH UH AH AH AH AH AH AH AH AH AH AH AH AH UH - - - . where - represents silence and . represents a syllable boundary.

The phoneme string is condensed by deleting the initial silence, all phonemes which occur only once in succession, and all but one occurrence of a phoneme in a string of identical phonemes. For example, the spoken work STOP above is condensed to . S - . TH AH - . This condensation adequately solves the phonemic segmentation problem.

The condensed phoneme string is then transformed into a sentence of the language by a chronological scan of the string using a procedure based on properties of the syntax and vocabulary of the language. Effectively, each word is defined as a sequence of phoneme sets. For example, the word ZERO is defined as the sequence \{Z,S\}, \{E,I,EE,AY,\}, \{R\}, and \{OH,OO,UU,L,V,UH\}. An instance of the word in a condensed phoneme string is the sequential occurrence of one or more members of each set in turn.
An instance of ZERO might be "Z I E R OH OO UU." The procedure is generally to keep finding the longest possible next syntactically legal word. The last phoneme of a word may count also as the first phoneme of the next word, such as the S phonemes in EQUALS SIX SEVEN.

10. RESULTS AND RECOMMENDATIONS

As an experiment, the program as shown in figure 4 was spoken into the system. The object was to obtain a perfectly correct program in the computer (for subsequent compiling and execution) using the SCRATCH statement as necessary to edit incorrectly recognized statements. Using a rounded cross-section of SPOCOL syntax and vocabulary, there are 123 words in the 24-statement program varying from a statement of one word to a statement of 11 words and averaging 5 words per statement. With the author as speaker 18 of the 24 statements (75%) were accepted and correct at the first try. 5 were correct (20%) at the second try, and the remaining one (5) required three tries. A total of 175 words were spoken with 163 (93%) correct. With other speakers using the author's prototype phonemes, the results averaged around 50% statements correct at the first try ranging up to 6 tries to get a correct recognition. An average of 75% of the words were recognized correctly. For neither the author nor most of the other speakers was the system frustrating to use.

The results of this research indicate that the analytic cepstrum is useful for speech analysis and merits further investigation. Cepstral analysis could be a powerful complement to linear predictive coding in the acoustical analysis portion of a speech understanding system.
REFERENCES


