

AD-763 723

SPEECH UNDERSTANDING SYSTEMS

James W. Forgie

Massachusetts Institute of Technology

Prepared for:

Electronic Systems Division
Advanced Research Projects Agency

31 May 1973

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

**BEST
AVAILABLE COPY**



Technical Library

Speech Understanding Research

NATIONAL BUREAU OF STANDARDS
PHYSICAL LABORATORY

Lincoln Laboratory



DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Lincoln Laboratory, M.I.T.		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE Speech Understanding Systems			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Semiannual Technical Summary, 1 December 1972 through 31 May 1973			
5. AUTHOR(S) (Last name, first name, initial) Forgie, James W.			
6. REPORT DATE 31 May 1973		7a. TOTAL NO. OF PAGES 32	7b. NO. OF REFS 3
8a. CONTRACT OR GRANT NO. F19628-73-C-0002		9a. ORIGINATOR'S REPORT NUMBER(S) Semiannual Technical Summary, 31 May 1973	
b. PROJECT NO. ARPA Order 2006		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) ESD-TR-73-167	
c.			
d.			
10. AVAILABILITY/LIMITATION NOTICES Approved for public release; distribution unlimited.			
11. SUPPLEMENTARY NOTES None		12. SPONSORING MILITARY ACTIVITY Advanced Research Projects Agency, Department of Defense	
13. ABSTRACT The phonemic class segmentation and formant tracking algorithm have been extended and evaluated in some detail. Encouraging results have been obtained from a simple phoneme identification program working on the output of the segmentation program. The sentence generation program has been extended to produce questions and statements as well as commands appropriate to the task domain of the Lincoln experimental speech understanding system. A semantic component has been added to the heuristic search program used in processing utterances in this task domain (the vocal command of the speech data retrieval, analysis, and display system). A General Syntax Program has been implemented to support work on a variety of parsing strategies. A doctoral thesis on locally organized parsing for spoken input has been completed. The speech data base hardware and software have been actively used in supporting workshop meetings among the ARPA Speech contractors. Encouraging results are being achieved with an automatic phonetic labeling program. Network retrieval of data from the data base has been demonstrated. The TX-2 system has been extended to provide support for the various current areas of activity. A new disk system has been delivered and is being installed. A user authentication (password) scheme has been developed which is well suited to the open environment of the TX-2 system where the usual mechanisms for hiding user identification information are not available.			
14. KEY WORDS speech understanding systems linear predictive coding (LPC) phonetic recognition TELENET LPARS SURNET TX-2 System			

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

SPEECH UNDERSTANDING SYSTEMS

SEMIANNUAL TECHNICAL SUMMARY REPORT
TO THE
ADVANCED RESEARCH PROJECTS AGENCY

1 DECEMBER 1972 - 31 MAY 1973

ISSUED 2 JULY 1973

Approved for public release; distribution unlimited.

EXINGTON

MASSACHUSETTS

100

The work reported in this document was performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. This work was sponsored by the Advanced Research Projects Agency of the Department of Defense under Air Force Contract F19628-73-C-0002 (ARPA Order 2006).

This report may be reproduced to satisfy needs of U.S. Government agencies.

Non-Lincoln Recipients

PLEASE DO NOT RETURN

Permission is given to destroy this document
when it is no longer needed.

SUMMARY

The phoneme class segmentation and formant tracking algorithms have been extended and evaluated in some detail. Encouraging results have been obtained from a simple phoneme identification program working on the output of the segmentation program.

The sentence generation program has been extended to produce questions and statements as well as commands appropriate to the task domain of the Lincoln experimental speech understanding system. A semantic component has been added to the heuristic search program used in processing utterances in this task domain (the vocal command of the speech data retrieval, analysis, and display system).

A General Syntax Program has been implemented to support work on a variety of parsing strategies. A doctoral thesis on locally organized parsing for spoken input has been completed.

The speech data base hardware and software have been actively used in supporting workshop meetings among the ARPA Speech contractors. Encouraging results are being achieved with an automatic phonetic labeling program. Network retrieval of data from the data base has been demonstrated.

The TX-2 system has been extended to provide support for the various current areas of activity. A new disk system has been delivered and is being installed. A user authentication (password) scheme has been developed which is well suited to the open environment of the TX-2 system where the usual mechanisms for hiding user identification information are not available.

Accepted for the Air Force
Joseph J. Whelan, USAF
Acting Chief, Lincoln Laboratory Liaison Office

CONTENTS

Summary	iii
Glossary	vi
I. PHONETIC RECOGNITION	1
A. Segmentation	1
B. Formant Tracking	4
C. Phoneme Identification	5
II. LINGUISTICS	7
A. Lincoln Experimental System	7
B. General Syntax Program	10
C. LPARS - A <u>L</u> ocally <u>O</u> rganized <u>P</u> ARSer for Spoken Input	15
III. SPEECH DATA BASE	18
A. Speech Input/Output Hardware	19
B. Automatic Labeling	19
C. Data-Base Software	22
D. SURNET	23
IV. SYSTEM ACTIVITIES	23
A. TSP System	23
B. TX-2 System	24
C. TX-2 Password Scheme	24

Preceding page blank

CLOSSARY

APEX	TX-2 time-sharing system
ARPA	Advanced Research Projects Agency
BCPL	Basic Combined Programming Language -- intermediate-level language for computer programming
FDP	Fast Digital Processor -- Lincoln Laboratory computer designed for waveform processing applications
GSP	General Syntax Program -- a software system to support natural language processing
LPC	Linear Predictive Coding -- method for signal analysis being used in current speech research
SATS	Semiannual Technical Summary Report
SURNET	Specialized server process intended to provide network access to speech data base on TX-2
TELNET	Software which allows console on one network computer to function as console for another
TSP	Terminal Support Processor

SPEECH UNDERSTANDING SYSTEMS

1. PHONETIC RECOGNITION

Work in phonetic recognition is proceeding along two rather different lines. The first is concerned with the front-end problem of extracting basic information from the speech signal, given little or nothing in the way of linguistic support for the recognition process. The goal of the front end is to produce something like a phonemic transcription of the input speech which can serve as an input to linguistic processing modules. The expectation is that such a transcription will, at best, be an approximation to a nominal transcription made by a human listener. While significant amounts of error and ambiguity in such a transcription are to be expected and can probably be tolerated, the transcription must be accurate enough to guide the linguistic processing modules toward the hypothesis of the correct sentence in a reasonable amount of time.

The bulk of our phonetic-recognition work to date has been concerned with the front-end problem. We have developed and previously described algorithms for phoneme class segmentation and formant tracking. We are currently working on the identification of individual phonemes both by examining overall frequency-time patterns and by detailed examination of formant motions.

The second line of work in phonetic recognition is aimed at the verification of hypotheses generated on the basis of linguistic as well as acoustic information. The problem here may be to determine which of a set of proposed words, phrases, sentences, or whatever is the best fit to the input data; or the problem may be to determine whether a particular hypothesis is acceptable or not with respect to some arbitrary measure of goodness of fit. The verification process must expect to have to cope on occasion with minimal pair phonemic decisions and the effects of coarticulation, and it thus must make finer, more sophisticated judgments than are required of front-end recognition processes. On the other hand, the set of candidates to be discriminated among will generally be much smaller than the full set of possibilities which the front end must face. A previous study¹ of the interaction between acoustic discriminations and linguistic constraints suggested that the human listener adjusts the measurement space for the discrimination according to the set of alternatives suggested by the linguistic context. We feel that a speech understanding system will need to make similar use of context to make similar discriminations.

Our current work in the verification area is aimed at developing a practical means of defining an appropriate discriminant space given a limited set of candidate words. Since the set of candidate words is different each time a verification is to be made, the recognition space will be different and, hopefully, the maximum help will be gained from the context available at that moment. A simplified experiment has been planned to test the potential utility of this concept. Appropriate sentences have been recorded and entered into the speech data base. Further discussion of the experiment will be deferred until the next report in this series when results should be available.

The following sections indicate the current state of development and evaluation of the previously described phoneme class segmentation and formant tracking algorithms, and present some results from the phoneme-identification research.

A. Segmentation

Considerable development and data collection have been done on the segmentation algorithm. The most significant addition to the algorithm since the last SATS² is a generalized dip detector,

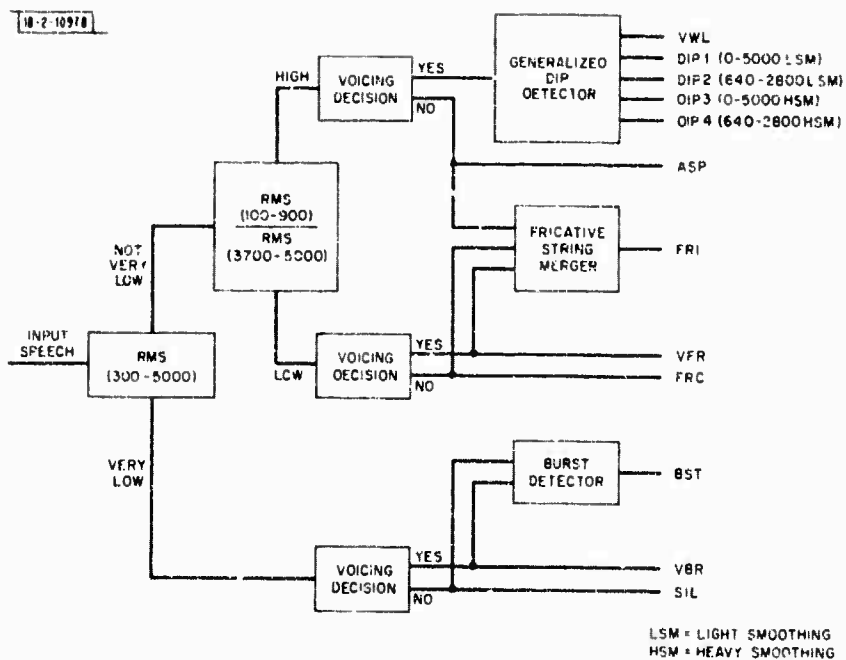


Fig. 1. Phoneme class segmentation algorithm. RMS (A-B) = root mean square of sum of squares of spectral components in range A-B.

18-2-10979

SEGMENT LABELS

	VWL	DIP	FRI	VBR	BST	ASP
VOWELS	265	9				7
NASALS, GLIDES	13	45		5		
FRICS F, TH S, Z SH, ZH		1	87	6		2
STOPS, AFFRICS		4	4	113	61	58
FLAPPED T	1	8				
H	2	7	1			4
V, DH	6	15	2	22	6	11

Fig. 2. Confusion matrix - phoneme class segmentation results for 38 sentences.

which allows detection and marking of dips in any of several volume (root-mean-square amplitude) functions during vowel-like segments. This generalized dip detector has considerably improved the detection of voiced consonants which are either between vowels or between a vowel and a strong fricative. Another addition to the segmentation algorithm is a burst detector, designed to locate the burst following a stop consonant.

The current segmentation algorithm is shown in Fig. 1. The indicated spectral measurements are performed on the linear predictive spectrum (previously, the homomorphic spectrum was used). Every 5 msec, one of the 12 segmentation symbols is assigned to the speech. (Note that the updating rate was previously every 6.4 msec.) A string of like symbols is referred to as a segment of a particular class. Considerable editing is done to eliminate unduly short segments.

The generalized dip detector processes all vowel-like segments. First, dips are sought in a lightly smoothed version of RMS (0-5000)*; these detected dips are marked as DP1. Remaining vowel-like segments are then successfully examined for dips in the three other volume functions.

The vowel-like segments which are not DIPs are marked as VWL. The segment classes FR1, VFR, and FRC are intended to locate the fricative sounds /S/, /Z/, /SH/, /ZH/, /F/, and /TH/.† The classes SIL and VBR are intended to locate closures in stop consonants. BST and ASP are useful in locating the burst and aspiration following a stop consonant.

A summary of some results obtained from the segmentation program is shown in Fig. 2. These results were obtained on 38 sentences, from several speakers. The sentences were phonemically labeled by observation of spectrograms and knowledge of what was spoken. The correspondence of phonemes to segment labels which was observed and entered into the matrix was not 1:1 in either direction. A phoneme could have more than one segment (e.g., STOP → SIL, BST, ASP or an insertion error like VOWEL → VWL, DIP, VWL) and a segment could correspond to more than one phoneme (e.g., FRIC, FRIC → FRC). A separate tabulation of all non-1:1 cases has been made. In tabulating the matrix, it was important to judiciously group both the phonemes and segment classes. For example, the phonemes /V/, /DH/ are separated in the tabulation from the other fricatives because they appear acoustically more like DIPs or stops. The flapped /T/ has been put in a separate category from the other stops, because its acoustic realization is generally as a short DIP rather than as a short silence. Nasals and glides which were adjacent to stops were not included in this data tabulation, since the algorithm does not yet have a strategy for segmenting these phonemes.

A number of interesting observations, which are suggestive of the course of further work, may be made from this matrix. All vowels in the corpus were marked at least during part of their duration by VWL and, conversely, all VWL segments contained at least one vowel. Hence, the job of vowel location is in hand, and we can proceed with vowel identification. Detection of the fricatives /F/, /TH/, /S/, /Z/, /SH/, and /ZH/ was quite reliable and this group of sounds seemed to be very promising for separation into classes which are more phoneme-like. Section 1-C reports our work on identifying these segments. STOP detection also was rather satisfactory. The dip detector found a total of 88 dips, of which 53 were DP1 and the rest were DP2, DP3, or DP4. The DIP segments included many different phonemes. Some work on further identification of DIPs is also reported below. The phonemes which spread most widely over the segment classes are /V/ and /DH/.

* RMS (A-B) = root mean square of the sum of the squares of the spectral components in the range A-B.

† In this report, phoneme symbols are represented in the computer-compatible two-character code which has been adopted for use in the ARPA Speech Understanding Research program.

V. Formant Tracking

The formant tracking algorithm, described in detail in the previous SATS,² has been slightly improved and thoroughly evaluated. The algorithm now keeps statistics on how often it had to perform its various correction measures and also generates a code word describing what sort of problems it encountered for each frame. The formant trajectories are being thoroughly studied to determine how the phonemes can be further segmented and identified on the basis of formant information.

Following is a brief review of the six steps in the processing of each frame:

- (1) Fetch all peaks (up to 4) in the spectrum in the frequency region from 150 to 3400 Hz.
- (2) Fill all 4 formant slots with the peaks on the basis of peak location relative to an educated guess.
- (3) If a peak fills more than one slot, keep it only in the slot that it fills best, and remove it from any other slots.
- (4) If there exists a peak which does not fill any slot as yet, try to find an empty slot and, if necessary, move peaks to new slots to accommodate it. If there are no empty slots nearby, or if the amplitude of the extra peak is sufficiently small, throw the extra peak away.
- (5) If there is an empty slot, recompute the spectrum on a circle of radius less than one, to enhance the peaks, and hopefully separate two merged peaks or bring out a peak which had been lost due to nasalization effects. Repeat steps (1) to (5) using the enhanced spectrum.
- (6) Accept formant slot contents as answers for this frame and as an educated guess for the next frame.

The only modification made in the processing of each frame is that now enhancement is tried over a larger range of radii. It was found that, if the radius was allowed to become as small as 0.88 (previously it had stopped at 0.955), then enhancement is far more successful at retrieving missing peaks.

After the six steps of processing have been applied at each voiced frame, each formant track is separately smoothed. First, a correction for gross errors is made - if one, two, or three frames are out of line but the surrounding four frames are in line, then each unaligned frame is corrected by interpolation. Finally, the formant trajectory is sent twice through the following smoothing filter:

$$F'_1(n) = 1/4F_1(n-1) + 1/2F_1(n) + 1/4F_1(n+1) \quad .$$

However, $F_1(n)$ is not replaced by $F'_1(n)$ if $|(F'_1(n) - F_1(n))| > 100$ Hz. The result is that the trajectory becomes very smooth where it was already reasonably smooth, but sharp changes in formant frequency, such as at the boundary between a nasal and a vowel, are retained. In addition, if the onset of voicing had been inaccurately determined, such that the formant data were badly out of line for a couple of frames, such erroneous data would not distort the good data adjacent to it, as the smoothing would not be applied there. This rule for not replacing $F_1(n)$ with the smoothed value takes effect on about 3 percent of the voiced frames.

Statistics were collected for some 50 sentences on how often the various correction measures were necessary. The statistics showed that it is much more common in linear prediction spectra for a peak to be missing than for a spurious peak to exist. Although statistics varied considerably from sentence-to-sentence enhancement was tried, on the average, in about 15 percent of the voiced frames. Out of these, a peak was found through enhancement in nine out of ten cases. In the other one-tenth, either the frame was mistakenly labeled voiced, or the formant was too strongly canceled by a nearby zero (in nasals and nasalized vowels) or, rarely, a peak merger was not successfully resolved.

In 1 percent of the voiced frames, F_3 was mistakenly called F_4 initially, and was later moved to the F_3 slot after enhancement failed to yield a peak. In another 3 percent, continuity constraints had failed in the initial slot-filling steps, and peaks had to be moved to new slots in step (4) to accommodate a peak about to be thrown away. An equal number of peaks were thrown away in step (4) either because they failed to pass the amplitude test or because there was no slot available for them. These extra peaks were usually due to nasalization effects.

Second-pass corrections of gross errors were rare, occurring only about 1.5 percent of the time.

C. Phoneme Identification

Programs have been developed to yield phoneme-like identification of speech events which have been isolated as fricative or dip segments.

The design of the fricative identification program is based on the assumption that a fricative segment contains one of the phonemes /S/, /Z/, /SH/, /ZH/, /F/, /TH/. This assumption is supported by the statistics appearing in the "VFR, FRC, FR1" column of Fig. 2. After several stages of experimentation, the fricative program emerged in the following very simple form: $RMS(0-2500)/RMS(0-5000)$ is used to distinguish the weak voiceless fricatives /F/, /TH/ from the rest (high for /F/, /TH/); $RMS(3500-5000)/RMS(2000-5000)$ is used to distinguish /S/, /Z/ from /SH/, /ZH/ (high for /S/, /Z/). The present program makes no attempt to distinguish between voiced and unvoiced phonemes (e.g., /S/ vs /Z/).

The dip-identification program is based on the assumption that the primary phonemes occurring as dipoles are the nasals, liquids, glides, flapped /T/, /V/, /DH/, and /B/ (pronounced without strong lip closure). This assumption is supported by the statistics in the DIP column of Fig. 2. The identification of flapped /T/ and /B/, /V/, /DH/ is based primarily on the use of duration information, and the behavior of the spectral derivative in the middle two-thirds of the segment. The distinction between flapped /T/ and (/B/, /V/, /DH/) is based exclusively on duration.

The results of these programs are summarized in Figs. 3 and 4. The input used for testing consisted of 60 sentences spoken by six speakers.

The performance of the fricative program was very good, as can be seen from the dominance of the main diagonal in Fig. 3.

The performance of the dip program is mixed (see Fig. 4). The identification of flapped /T/ is quite successful (13/14). The identification of /B/, /V/, /DH/ is only 5 out of 10; note, however, that no other phoneme is ever mislabeled BVDH, out of a total of 79 dipoles. Not surprisingly, it has been found to be quite difficult to accomplish good separation of nasals from liquids only on the basis of spectral shape or spectral change. In particular, large spectral change at segment boundaries is completely ambiguous as a nasal-liquid cue. We feel that the use of formant tracks holds the best promise for success.

IDENTIFICATION LABELS

18-2-10980

INPUT PHONEMES	IDENTIFICATION LABELS			
	SZ	'HZH	FTH	
	(S,Z)	62	0	0
	(SH,ZH)	0	7	1
	(F,T:H)	2	0	29

Fig. 3. Confusion matrix - fricative-identification program.

IDENTIFICATION LABELS

18-2-10981

INPUT PHONEMES	IDENTIFICATION LABELS			
	FLAPT	BVDH	NASLIQ	
	(flapped t)	13	0	1
	(B, V, DH)	0	5	5
	(NAS, LIQ) (et al.)	2	0	53

Fig. 4. Confusion matrix - dip-identification program.

II. LINGUISTICS

At Lincoln Laboratory work in the linguistics area falls into three general areas:

- (a) Work oriented toward the exploration of acoustic/linguistic tradeoffs in the context of the Lincoln experimental system task domain. This task is the vocal command of our speech data retrieval, analysis, and display system. Section II-A presents a progress report on our work in this area.
- (b) The development of a General Syntax Program (GSP) to support more general parsing strategies than those required for our limited experimental task domain. Section II-B is a presentation of the concepts and capabilities of GSP as we have implemented it.
- (c) Thesis work aimed at the exploration of other approaches to the design of speech understanding systems. One of the two doctoral thesis projects being supported on TX-2 was completed during this reporting period. The concepts involved in this thesis and the results of experimental evaluation are presented briefly in Sec. II-C. The program developed for the other thesis project has been successfully used in an experiment to integrate a complete speech understanding system by linking our current primitive phonetic-recognition modules with the dictionary lookup, scoring, and parsing modules of the thesis program. The resulting system correctly recognized a few sentences but, as we expected, performance was poor because of the limited capability of the front end (phoneme class information only), and no attempt is being made to pursue the experiment further until a more capable front end is available. Further discussion of this experiment and the second thesis will be deferred until completion of the thesis project.

A. Lincoln Experimental System

We have planned a series of experiments to gain quantitative data about the information available for correcting errors and resolving ambiguities at the various levels of linguistic processing. To support the experiments, we are developing a number of program modules, some of which may serve as components of our experimental speech understanding system. In the previous SATS,² we described programs for generating sentences appropriate to our task domain and for the lexical segmentation of garbled phonetic strings using a heuristic search algorithm. The following sections report the current status of these programs and describe the extension of the processing algorithms to make use of semantic constraints.

1. Sentence Generation

Work on sentence generation is intended both to provide phonetically transcribed sentences in sufficient quantity for statistical studies, and to gain insight into the complexity of the semantic model required to produce reasonable sentences within our restricted linguistic environment.

The program, described in some detail in the previous SATS, has been extended to generate questions and statements as well as commands. The verb is still chosen first and, if a subject is needed, the subject and its modifiers are chosen next. Finally, an object (if required) and its modifiers are selected. A possible subject is a question word ("which," "who," etc.) and selecting such a subject will cause a question to be generated.

The program has also been expanded to form complex sentences. The vocabulary has almost doubled, and is now about 300 words.

2. Lexical Segmentation and Parsing

The last SATS described the initial version of a system that applied the technique of heuristic search to the problem of lexical segmentation. Since that time, the system has been expanded to include a semantic component, and major changes have been made in the lexical segmentation component (now called the dictionary module).

a. Two-Stage Strategies

The conflicting constraints of real-time operation and large computational requirements for current algorithms have motivated an investigation into the use of two-stage strategies as a means of using processing time more fruitfully. By a "two-stage" strategy, we mean one that uses cues (sometimes called "insights" or "heuristics") to access a particularly promising set of possible answers. A separate algorithm performs the second-stage task of analyzing the selected subspace in more detail. Such strategies are common in human decision-making.

b. Dictionary Module

The dictionary now contains 300 words pertaining to our task domain. Stored with each word is its phonemic transcription, including stress indicators. For some words, the dictionary also contains pointers into the semantic grammar (see below).

The dictionary words are accessed through a three-level net whose arcs are elements of the most strongly stressed syllable in the word. This organization was chosen because the stressed syllable usually contains the most energy and is most likely to be correctly identified. We expect to augment this net by including access through other reliable indicators such as strong fricatives.

The dictionary module is called once for each stretch of the input which has been determined to be strongly stressed. The vowel in this stretch and the consonants on either side of it are used to calculate a set of syllable classes that are "near" to the one in the input. Other classes are added by postulating word boundaries before or after the vowel. This set is used to access a set of dictionary words. Each word is matched with the input by aligning on the stressed syllable and matching outward. Each word is assigned a score and the words are sorted into decreasing order according to their score.

This two-stage strategy reduces processing by limiting the expensive matching analysis to those words most likely to achieve a high score. The more reliable the input from the acoustic processing, the stricter we will be in defining "nearness," and the fewer words we will have to consider. There is no reason why nearness cannot be defined dynamically if the acoustic processing provides "confidence numbers" for each segment.

The output from the dictionary module is a sequence of ordered sets of words, one set for each stressed section of the input. The advantage of this technique over the more-common one of matching hypothesized words individually against the input lies in the fact that the absolute score of a word is often not as important as its relative score with respect to the alternatives for this position in the utterance.

c. Semantic Grammar

Semantic information is represented in a semantic grammar. This is a phrase structure grammar in which syntactic classes such as "adjective" are replaced by semantic classes such as "spectrogram-descriptor."

We use this formalism to define a series of sentence-forms, each of which is keyed to one particular type of command or question defined in our domain. The variations specified by the alternatives in the grammar express both the optional equivalent forms of stating the same request, and the allowable values for the parameters in the request.

The grammar is constructed so that a successful parse yields a functional representation of the sentence that can be easily mapped into an internal executable process.

Prior to run time, the grammar is compiled into a list-structured internal representation. At this time, the entry of each dictionary word that is in initial position in a sentence-form is modified to point to the head of that sentence-form (it may point to more than one).

d. Semantic Module

This module uses the output from the dictionary module and the semantic grammar to produce an hypothesis about the identity of the original spoken sentence. Those words in the first (left-most) set produced by the dictionary that can begin sentences are used to access a set of sentence-forms. Since all variations for each of these forms must be considered as candidates and the number of variations is usually very large, the heuristic search approach is used to organize and generate the hypotheses.

The heuristic search paradigm requires that we define an initial set of states, a new-state generator, an evaluation function, and a goal. In this instance, the initial states are the sentence-initial words found by the dictionary module. A new state is generated by using the semantic grammar to add one word to a previous state. The goal is to generate a sentence that differs from the input by less than some given amount. The evaluation function determines which state will be used to generate new states. By simply changing some of its coefficients and the level of error allowed, we can change the behavior of the system from slow but thorough, to fast but inaccurate, or anywhere in-between.

Implementation of an initial version of the module has been completed, but much experimentation will have to be done to evaluate the tradeoff between thoroughness and accuracy.

A major part of the effort in the next few months will be devoted to augmenting the types of associative links (from words to sentence-forms). The strongest candidates for new links are verbs in non-initial position. This may require use of the semantic grammar in reverse mode (right-to-left order) in some cases.

e. Results

Since the development of the acoustic processing module is proceeding in parallel and is not yet completed, all experiments with the system have been on data produced by a "garbler." The garbler takes typed sentences and generates phonemic output which has some given degree of error and ambiguity.

Several sentences have been successfully run through the system consisting of the garbler, the dictionary module, and the semantic module. The times average about 10 sec, but this is

an unrealistically low figure since only a very small fraction of the total semantic information has as yet been encoded into the semantic grammar. Moreover, the error level of the garbler was probably not as high as that to be expected from acoustic processing of actual data.

B. General Syntax Program

We have implemented a General Syntax Program (GSP) on TX-2 because it appears to be a good vehicle to study (1) several different parsing strategies within the same general scheme, and (2) the use of parallel processes in parsing. The basic structure is the same as the GSF programmed by Ron Kaplan,³ but our emphasis has been on taking advantage of the graphic capabilities of TX-2. So far, we have run with GSP some very simple transition net type grammars which were constructed basically to test the program, and a word finding grammar, also in the form of a transition net but compiled into a GSP grammar from a dictionary that gives, for each word, phonemic representations and syntactic categories.

GSP has two basic components: a grammar which specifies how structures are to be built, and a chart which is used to represent initial structure for the sentence and any structure found by the grammar. An application of the grammar to a part of the chart (e.g., an attempt to find a noun phrase beginning with a particular point of the input) is called a process. Since one of the things a grammar rule may do is to initiate a process, there will generally be several processes "alive" at any given point of parsing, although (since TX-2 is actually not a parallel processing machine) only one will be "active" at a time.

GSP could be thought of as a parallel processing machine executing several processes simultaneously. The grammar could be thought of as the program, the chart as the data which the program works on, and a process as an instantiation of a program working on a part of the data. The processes are like subroutines in that one process will start another to obtain information that it needs, but they differ in that (1) the results are not returned to the calling process, but placed in a communication port to be read by any other process (including the one that created it), and (2) a process has a life independent of its creating process and may well continue after the process that created it disappears (it may have even existed before a process that "creates" it existed, since an attempt to create a process to do precisely what an already-existing process is doing results in the use of the existing process).

1. Example of a Chart

The chart represents all structure found thus far and is shown graphically as a series of boxes, each of which may contain one or more shelves. Each box contains information about a kind of structure, and each shelf contains information about a particular structure of that kind. Each shelf has a label, a sister, and any number of other features with values. For example, Fig. 5 shows the chart in our graphic representation at the beginning of parsing the sentence:

"The big bear eats sweet honey."

Each word of the sentence (we are now assuming typed input, although we may use phonemes as the basic input) has been placed in a separate box as the only shelf in that box. Each shelf contains as its label the word and, as its sister, the number of the box that has the next word. A sister whose value is 0 (e.g., in box 6 of Fig. 5) indicates the last word of the sentence. Parsing involves placing boxes on top of these input boxes and adding shelves to existing boxes to express syntactic information about the sentence. Each shelf in such an added box contains

information about structure covering a part of the sentence from the word indicated by the lowest shelf of the column to just before its sister

Figure 6 shows the chart after a complete parse. Grammatical information relating to a particular point of the input string is represented in boxes placed in the same column as that word. The grammatical category represented is shown as the label of the shelf, and the extent is shown as the sister of the shelf. In looking at the chart, try to see, instead of a number as the value of the sister feature of each shelf (e.g., "sis 5"), an arrow pointing from that shelf to the column whose number is the value of "sis." We do not actually display the arrows because there are so many that the resulting picture would be very confusing.

Looking at the first column of boxes in Fig. 6, note that three boxes have been placed on top of the original box 1 [box numbers are in the lower-left corner and, except for the input string, bear no significant relationship to the meaning of the box (the numbers are assigned, like house numbers on a Tokyo street, in chronological order of creation of the box)] Box 1 has had features added to it to specify its status as a determiner (det), adjective (adj), noun (n), beginning of noun phrase (np), and beginning of sentence (s). The parse has determined that the first word of the sentence "the" is a determiner, is not an adjective or noun, and begins a noun phrase and a sentence. Further, information about its status as a determiner is stored in box 10, information about the noun phrases it begins is stored in box 9, and information about sentences it begins is stored in box 8. The boxes referred to are stacked on top of the box they relate to and we can see that box 8 shows that there are two sentences (s) beginning with "the," the first one covering the entire input string (since its sister is 0) and the second covering the first four words (its sister being 5). Note that each sentence is represented by a separate shelf in box 8. Box 9 shows information about the two noun phrases (np) beginning with "the" in the same manner. The internal representation of the chart has each np and s shelf contain information about the structure of the np or s it represents, but this could not be fitted into the graphical representation. The structures found by the parse could be represented by the trees shown in Fig. 7. (The sentence that does not cover the entire input string asserts that large people carry food.)

2. Example of a Grammar

The grammar is a collection of states, each of which has one or more arcs. It may be visualized by representing each state as a node in a network, and each arc as a line coming out of that node and pointing to another node (except for arcs that terminate a network). Figure 4 shows a representation of the grammar that was used to create the chart described above. Each arc is a little program which is run to determine whether the arc applies and, if it does, what is to be done.

Figure 8 gives a simplified form of the grammar, which is essentially a transition net. It says that a sentence is composed of a noun phrase followed by a verb followed by a noun phrase, and that a noun phrase is composed of an optional determiner followed by any number (including 0) of adjectives followed by a noun. Figure 8 is a concise description of the language parsed by the grammar, but it does not show any information about the process structure of the parsing system.

The user is not required to write a GSP grammar directly. Rather, he will be able to express his grammar in a form congenial to the way he thinks about it, and we will provide a compiler to translate his input into GSP's internal form. In this way, the GSP grammar language could be thought of as a direct language for a GSP machine with compilers provided from higher-level languages. Currently, we have a compiler which accepts as input a dictionary giving words

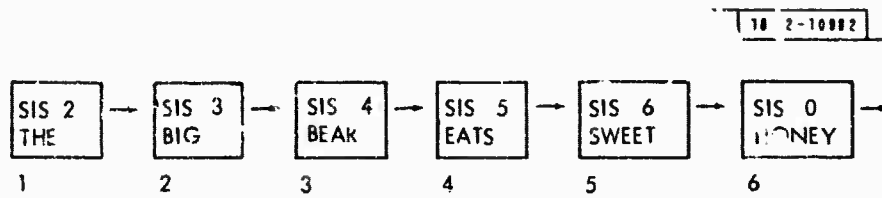


Fig. 5. GSP chart at beginning of parsing sentence: "The big bear eats sweet honey."

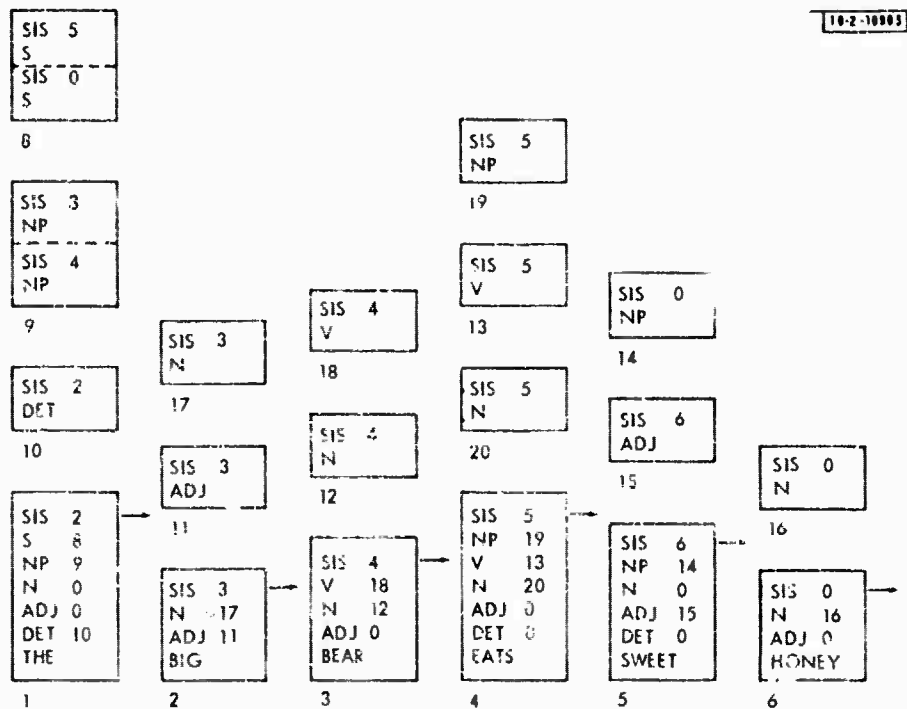


Fig. 6. GSP chart at end of parsing sentence: "The big bear eats sweet honey"

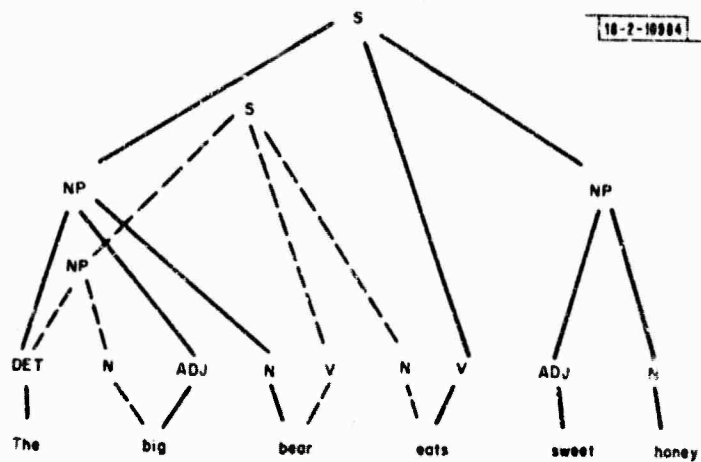


Fig. 7. Parse-tree representation of GSP chart of Fig. 6.

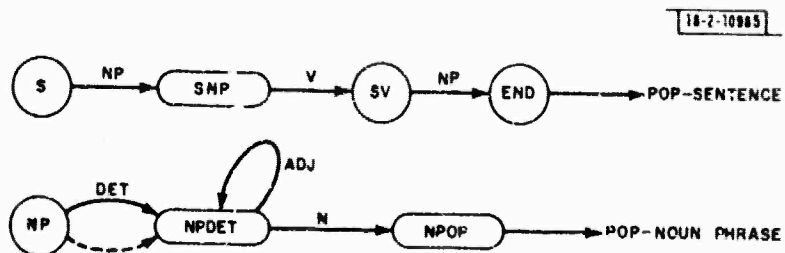


Fig. 8. Transition net representation of grammar used to obtain GSP chart in Fig. 6.

and their phonemic transcriptions, and which produces a CSP grammar to recognize words from a phonemic input.

3. Processes

The first process in a parsing is typically an initiator process which then itself initiates a process to actually begin searching for structure. In our examples, the initiator process starts at the first box but it could begin anywhere in the chart. Starting a process merely puts some data about its initial conditions on a process list. In principle, all processes run in parallel, but since the TX-2 is not a parallel machine there is a control program which determines which of the processes on the process list is to be run at a given instant. In the current implementation of GSP, we have arbitrarily picked a particular order, but we hope to explore the effects of varying the algorithm for determining which process is to be in execution at any given time.

Each process has a number of paths, each of which contains a grammar focus, a chart focus, and a list of registers. The grammar focus specifies a particular part of the grammar (i.e., state and arc), the chart focus specifies a place in the chart (i.e., shelf of a box), and each register specifies some conclusion about the structure the process is heading toward using that path [e.g., a path in the process looking for a sentence remembers the first noun phrase of the sentence as the value of the register "subj" (for subject)]. When a path in a process is given control, it applies the arc of the grammar specified by the grammar focus to the box and shelf specified by the chart focus.

When a process is initiated, it is necessary to specify the initial state and the name of the constituent being looked for. The process initiating program will check whether there already is a process with the same initial state and constituent being looked for initiated from the same shelf in the chart. If the process is new, then a process number is arbitrarily assigned the process, an empty box is created as the process communication port, and a feature is added to the shelf from which the initiating is done with that box number as its value.

During a parse, the chart is augmented by displaying, for each path, a process information rectangle inside the shelf that is its chart focus. This rectangle contains information about that path to enable the user to see which processes are paying attention to particular parts of the chart. An arrow is displayed to the left of the process information rectangle of the path that is currently being run.

4. Plans

It is our hope that the graphic chart display, with information about each process shown at the point in the chart that is the process chart focus, will enable the grammar writer to literally see what is going on when his grammar is applied to the input data. Since the chart will not, in general, fit on the scope, only part of it may be displayed at a given time. The user has controls that enable him to specify which parts he wishes to look at. The user also may have some control over the actual order in which the parsing occurs. He may specify situations in which the program will halt and he may point to the process he wishes to start next. In the near future, we plan to extend the ways in which the user may control a parse by allowing him to request information not shown in the chart.

Currently, processes have no communication with each other other than by what one process adds to a box being used as a communication port. However, it appears desirable to allow, say,

a syntax process, to make a suggestion to a word-finding process about what categories of words are expected. We plan to add this capability in the near future.

The only grammars now running with GSP are simple test grammars (except for the word-finding grammar which has a vocabulary of 300 words but assumes that there is no error in the input). The next step is to put some real grammars in and evaluate GSP as a grammar-developing tool as well as a parsing scheme.

In the course of developing GSP, we developed as tools a list processing system and a tree display and editing program. These programs may find other applications, and are being documented for the TX-2 user community. The list processor is a collection of subroutines in BCPL that allow the construction, reading, and writing of list structures. Even though the list processing system is different from LISP (our system uses cells of arbitrary numbers of addresses and reference count reclaiming rather than LISP's garbage collection), it has proved possible to come close to copying LISP programs directly. The tree display and editor have been implemented using the list processor and, with a program to convert any list structure to a tree, have proven to be a valuable aid in debugging programs which build list structure.

C. LPARS - A Locally Organized PARSer for Spoken Input

LPARS^{*} has been implemented on the TX-2. It is designed to process continuous speech with the help of syntactic and semantic information.

The LPARS system differs from traditional parsing methods in that it has no inherent left-to-right, or right-to-left, bias to its operation. Rather, it allows syntactic structures to be recognized locally in any part of an utterance. In fact, several phrase structures may be built up simultaneously in different parts of the sentence, and later connected together by searching for words that might reasonably exist between them.

Thus, words and phrases reliably recognized in any part of a sentence can be used to help guide the search for further words to complete the sentence.

1. LPARS Task Domain

LPARS' vocabulary contains approximately 70 words. It recognizes a very restricted, but linguistically natural and interesting, subset of English. Its semantics are defined in terms of a particular scene - a small two-room house containing people, furniture, fixtures, etc. - about which one may make statements, ask questions, tell a very simple-minded story, or command the system to manipulate the scene. Sample input sentences are:

"The coffeetable on which the ashtray is placed by Robert supports the dictionary."

"What does the sidetable support?"

2. Operation of LPARS

LPARS expects as input a string of phoneme candidates from a front-end phoneme recognizer. For the present work, the input was prepared by a phonetic scrambler program which simulated front-end behavior, rather than by a real phoneme recognizer. The input phoneme candidates may be ambiguous (i.e., several possibilities may be given for one segment). The input is also expected to contain a fairly large amount of error.

* The system was developed by P. L. Miller as part of a Ph.D. thesis at M.I.T. It will be published as Lincoln Laboratory Technical Report 503.

INPUT SENTENCE:

'THE LARGE COFFEETABLE SUPPORTS THE GREEN DICTIONARY'

13-2-10*86

(1) INITIAL SCAN: COFFEETABLE, GREEN

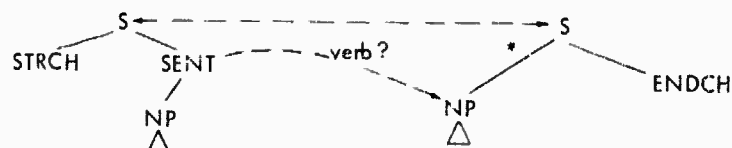
(2) LOCAL HIGHER DISTANCE SCANS:

STRCH	COFFEETABLE	GREEN	ENDCH
	(adj)	(prep)	(adj) (noun)
	(det)	(verb)	(det) DICTIONARY
	LARGE	NIL	THE
	(adj)		(verb)
	(det)		(prep)
THE		NIL	

(3) PARTIAL PARSE TREE CONSTRUCTION:



(4) CONNECTION OF PARTIAL PARSE TREES:



(5) RECOGNIZED SENTENCE:

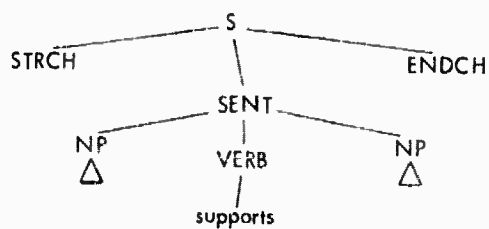


Fig. 9. Simple example of LPARS in operation.

LPARS operates by first making an initial scan through the sentence looking for longer words which are not too garbled. The scan returns a list of word candidates which match within a given error tolerance to specified sections of the input. It is likely that some of these word candidates are incorrect, and indeed some may overlap.

These word candidates are turned over to the higher-level part of the system which initiates scans for small words and for more highly garbled words in the areas adjacent to and between the words found in the initial scan, and groups the words together into parse structures. This processing is done systematically in an attempt to uncover the entire utterance.

3. A Simplified Example

Figure 9 is a simplified example of LPARS in operation. The input utterance being processed is the sentence: "The large coffeetable supports the green dictionary." In real operation, the sentence would be spoken, analyzed by a front-end phoneme recognizer, and a string of phoneme candidates produced for input to LPARS. Currently, this input is produced by a front-end simulator.

LPARS first makes an initial scan through the sentence, testing the words in its vocabulary against the entire length of the input. This scan is made at a low phonetic distance looking for longer words which are not too garbled. (Phonetic distance is a measure of how closely the phonetic spelling of a word matches a section of input.) Let us assume that, in this instance, the two word candidates "coffeetable" and "green" are found. These word candidates are turned over to the high-level part of the system, together with a start-of-sentence character (STRCH) and an end-of-sentence character (ENDCH) which the system adds.

The higher-level analysis consists of three fairly distinct stages, as Fig. 9 indicates. First, scans at higher phonetic distance are made based on fairly local cues, in the areas adjacent to the word candidates. In this example, these local scans are very simple. A scan in front of the noun uncovers the adjective "large" and a further scan in front of that word uncovers the determiner "the." A scan to the right of the noun for prepositions and verbs fails to find any word candidates. This means that the verb "supports" is too garbled to be picked up either by the initial scan or by the higher phonetic distance selective scan. Similar scans around the adjective "green" uncover the words "dictionary" and "the," but again fail to find the verb.

After these higher-distance scans have taken place, the system builds up as many parse trees as it can in the sentence. If all the words in the sentence have been found, then the entire sentence is constructed. Otherwise, the result is a number of partial parse trees (PPTs) in different parts of the sentence.

The system attempts to construct as many such PPTs as it can with the words it has found. In this example, it constructs only two PPTs. The first tree, "STRCH the large coffeetable," is straightforward; the second, "the green dictionary ENDCH," is somewhat unusual since it contains an "ancestor link" (the link labeled "*"). The ancestor link allows LPARS to recognize that, due to right recursion, many syntactic relationships are possible but to defer commitment until later when an attempt is made to join this structure to another structure by proposing words between them.

The third and final step in the parsing process consists of connecting PPTs to one another by using the grammar to propose words that might exist between them. Any words proposed are tested against the input at even higher phonetic distances than the previous scans.

In the present example, the algorithm discovers that the two parse trees can be connected if a verb is found between them. It therefore initiates a higher-distance scan which succeeds in finding the verb "supports." Thus, the entire sentence is recognized.

Notice that this example is a simplified one. No erroneous words were found. In a more-realistic example, erroneous words would be found, some local structures containing these words would be built up, and, additionally, some erroneous local parsings of the correct words could be constructed. Hopefully, attempts to build out upon the erroneous structures would eventually prove unsuccessful, while attempts to build out upon the correct structures would usually succeed.

4. Experimental Evaluation

The LPARS system was evaluated by processing 50 sentences. These input sentences were produced by a front-end simulator which approximated the accuracy of the Vicens-Reddy phoneme recognizer. The simulator operates roughly as follows: 15 percent of the phonemes are deleted; 10 percent of the remaining phonemes are badly scrambled (i.e., a stop might be changed into a vowel or fricative); the remaining phonemes are substituted for at random from a restricted class of phonemes (i.e., a stop might be changed into some other stop chosen at random).

During the evaluation, if several possible sentences were found, the recognition of the sentence was considered successful if the correct sentence was included among these. When several sentences were found, the correct sentence was almost always the best match.

During evaluation of LPARS, the 50 sentences were processed with the following results:

- 23 sentences were correctly recognized from words found only by the initial local processing.

- 19 sentences were correctly recognized by the PPT connection algorithm, after the initial local processing.

- 3 sentences were correctly recognized by fallback methods.

- 2 sentences resulted in incorrect recognition. In both cases, the sentence found differed from the input sentence in a single content word.

- 3 sentences resulted in failure to find any possible sentence at all.

Thus, the overall success rate of LPARS with the 50 input sentences was 90 percent. The evaluation of LPARS described above primarily establishes that, given a simulation of a fairly crude front end, the ideas embodied in LPARS can be made to work acceptably.

III. SPEECH DATA BASE

The speech data base has moved into a period of active use. Data for comparative analysis have been distributed to other ARPA-supported organizations for use in workshop meetings. The first such meeting was held at the Speech Communication Research Laboratory in Santa Barbara, California in March 1973. The second is scheduled for July 1973 at Carnegie-Mellon University. For the first meeting, we distributed audio tape copies of six utterances to six other organizations who presented displays of the results of their processing of the data at the meeting. For the second meeting, copies of 31 utterances have been distributed to twelve other organizations. Most of the groups received digital tapes, either 10- or 20-kHz sampled data, in the data-base format. Some groups received audio tapes made from the digital versions. We have already received indications that the uniformity resulting from digitization will greatly aid in evaluating the many analysis techniques to be compared at the workshop.

We now have the waveforms for 75 utterances in the data base and are at the practical limit of on-line storage. We do not expect to add more utterances until the new disk system (see Sec. IV-B) is available. Current activities to process and label the existing data will use the bulk of the remaining space.

A. Speech Input/Output Hardware

The specially designed speech-data input and output system has been through a shakedown phase as a result of data requirements for the July 1973 workshop. After some initial problems were corrected, satisfactory performance was achieved. The speech input/output system is designed to input analog speech data and to provide analog outputs of the processed speech. Conceptually, this is a simple analog-to-digital and digital-to-analog conversion process. However, to facilitate the handling of the large quantities of speech data and to meet other requirements, special-purpose hardware was developed; a block diagram of this hardware is shown in Fig. 10. One requirement of the speech data is that two sets of data are required for each of the sentences. One set is considered as normal speech with a channel bandwidth of 5 kHz. The other set is wideband or high-fidelity speech with a bandwidth of 10 kHz. These two sets of samples must be synchronized so that different processes can later be correlated. This requirement was met by feeding the speech through two sets of presampling filters, one of 5-kHz and the other of 10-kHz bandwidth. Then the filtered signals are fed to an analog multiplexer that commutates at a 40-kHz rate. The output of the analog multiplexer is then digitized. One out of every four conversions is not passed to TX-2, so that both the 5- and 10-kHz speech are sampled at their Nyquist rate. Thus, the samples of the two sets of speech data are automatically synchronized, and only one pass of the speech is needed to obtain the desired data.

To minimize the amount of storage and processing involved in dealing with silent intervals adjacent to speech signals, an analog threshold detector is used to detect the presence of speech and control the conversion process. However, some utterances begin with low-level sounds that have significant information. Therefore, the speech is fed through the equivalent of an analog delay mechanism realized by a modified tape recorder. The normal recording head is used as a pickup head for the threshold detector. The true speech is obtained through the normal playback channel. Thus, there is a delay of approximately half a second between the thresholding signal and the speech signal to be sampled. This period proves to be sufficient, and this arrangement works quite well.

One other requirement is to pre-emphasize the high frequencies of the input speech prior to the analog-to-digital conversion. The choice of a pre-emphasis characteristic involves a compromise between loss of information of weak signals and distortion of strong signals caused by either overloading or aliasing. After some experimentation, we have settled on the pre-emphasis characteristic shown in Fig. 11.

The speech output system de-emphasizes the high frequencies to complement the input characteristics so that flat overall frequency response is maintained. The combined input and output frequency response is shown in Fig. 12.

B. Automatic Labeling

The quantity of data involved with the buildup of the speech data base and the desire for strict consistency in labeling it indicated the need to automate as much of the labeling task as possible. A design for an automatic labeler was presented in the last SATS.² Essentially, it uses the results

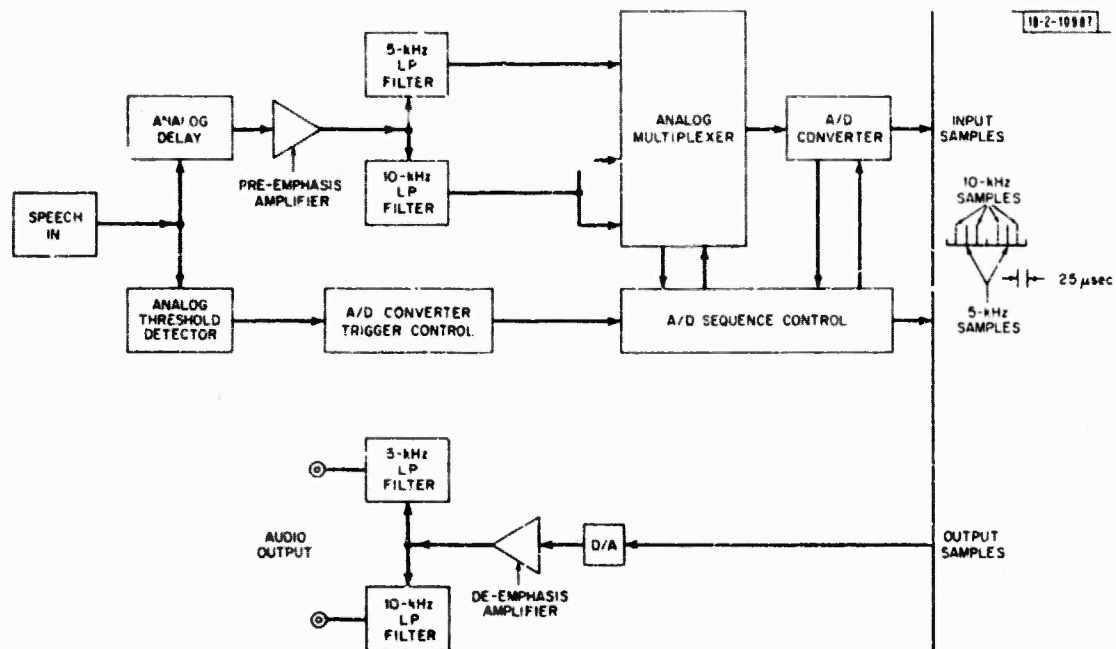


Fig.10. TX-2 speech input/output system.

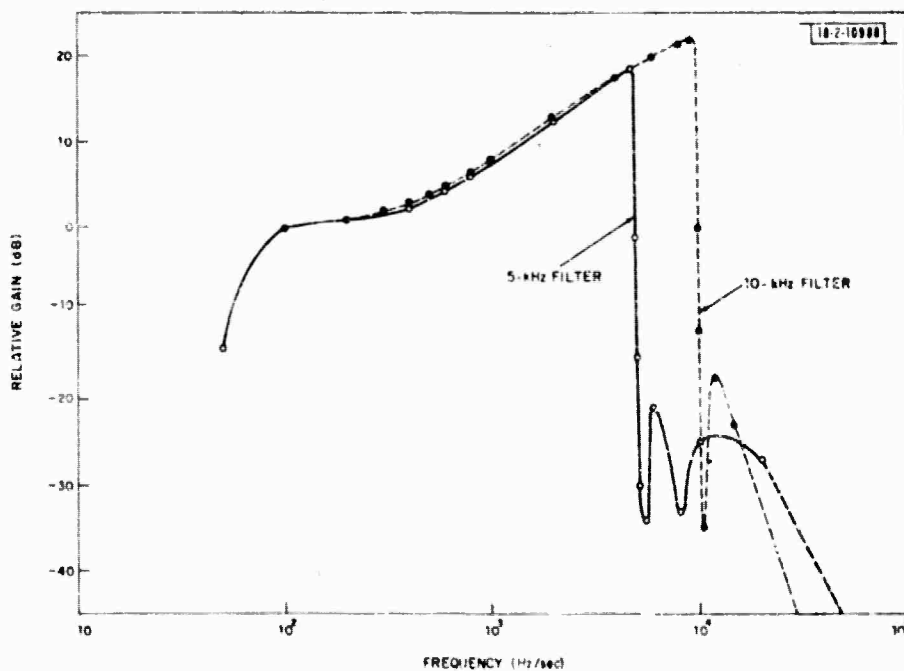


Fig.11. Frequency response of TX-2 speech input system showing net effect of pre-emphasis and presampling filters.

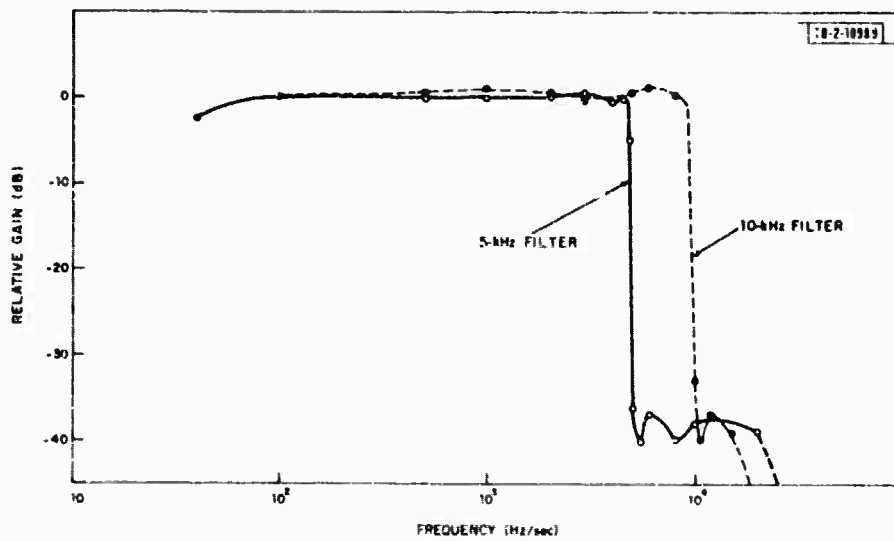


Fig. 12. Combined input and output frequency response for TX-2 speech input/output system.

of gross acoustic processing (segmentation and classification into such recognition units as vowel-like, aspiration, fricative, silence, etc.) to assign the start and end times for the phonemes in the phonemic transcription of an utterance. With some modifications, the design has been implemented and is undergoing extensive testing.

The results thus far are good enough to be useful since in the test run about 80 percent of the labels have been correctly placed, but better use of predicted durations and increased knowledge of the idiosyncrasies of the acoustic programs should produce even greater accuracy.

The basic problem involves the recovery from wrong assignments caused by errors in the acoustic processing. At any point, there are three possible assignments:

- (1) Assign the next phoneme to the next acoustic segment.
- (2) Assign the next phoneme to the current acoustic segment. (This should be done if the phoneme was not isolated as a separate acoustic segment, as is often the case with semivowels.)
- (3) Assign the current phoneme to the next acoustic segment. (This should be done if the acoustic segment is a sub-phonemic event, such as the burst in a plosive.)

Of the errors and lack of specificity in the current acoustic processing, it is often not clear at the moment which of the three should be done. Only later does one find that he is on the wrong track and should return to some earlier decision point to take a different path.

The heuristic search paradigm is used to organize the decision-making. Each state (sequence of assignments) is assigned a value by an evaluation function, and the best scoring of all previously generated states is made the current state. Thus, backup is not in any predetermined sequence, but instead is decided upon dynamically as a function of the goodness of fit of the alternate hypotheses. Moreover, the evaluation function can actually be a subroutine of any degree of complexity, and thus can take into account phonological rules, expected durations, and any knowledge of the types of error most likely to occur in the acoustic processing.

For those phonemes which were not isolated as separate acoustic segments, boundaries are estimated on the basis of the expected duration of the phoneme, given its context.

C. Data-Base Software

Software written specifically for speech research on TX-2 is now quite extensive. While most of this has been described in earlier reports, substantial improvements are being made or have been completed for these facilities, in response to needs, suggestions, and reactions of users. All speech workers on TX-2 use the Speech Processing Controller (SPC). This is an operating system and command interpreter. New capabilities and improvements have been added to it over the last year-and-a-half, and it is now extremely well-tailored to the project.

The A/D input received heavy use for the first time, and this resulted in a number of modifications to the associated software. Validation of the data at several stages was, of course, essential. Listening to the D/A output was especially helpful for this. Several programs were written to facilitate data access and D/A of the 10- and 20-kHz waveforms at three stages: after digitization but before entry into the data base; after entry into the data base; after being written to tape. To simplify the validation process, the A/D and D/A programs have now been rewritten to use the sub-directory facility and hence be consistent with other software. Certain critical pieces have been recoded in machine language to speed up the entire operation.

One of our users is making a systematic study of the effects of phonetic context on vowels. This work requires a particular CVC syllable to be spoken in a carrier sentence. In response to the need to get these syllables into the data base, a selective input facility has been built. To achieve reasonable storage efficiency, a number of individual syllables are collected together into a "pseudo utterance." This facility allows an operator to input the syllable in its carrier sentence via the A/D converter, and use the graphic display facility and D/A to select the syllable of interest. The waveforms for the individual syllables are entered into the data base as separate fields, until all those that are to be put into a single pseudo utterance have been collected. Then a single field is constructed containing the concatenated syllables, with silences inserted between them. The short individual fields are then deleted.

D. SURNET

The SURNET server described in the last SATS has been implemented. The current version only provides the minimum response to a "query" command. After establishing a connection to SURNET, the user process must transmit a simple parameter list consisting of the query command code, the entry number, and field type of a field to be retrieved from the data base. If the search is successful, the field length and the data are transmitted to the user process. If the requested field cannot be located or problems with the network arise, SURNET closes the connections. In-house tests have established that this version of SURNET is operating correctly. We are presently running tests between BBN TENEX and TX-2. Following these tests, the server will be developed further to accept a complete parameter list instead of the simple two-word list, to return a complete network header with the data instead of field length alone, and to return informative error codes in response to system failures rather than close the connections. The "store" command will be added when space allocation conventions and controls have been determined.

IV. SYSTEM ACTIVITIES

A. TSP System

A number of spurious errors (bad parity indications and memory addressing failures) led us to suspect that one of the eight memory controllers was erratic. It was replaced and now the hardware reliability is adequate for a production system. There are still occasional failures, however, which we are continuing to track down.

We are in the process of implementing a text editor for the TSP. It uses the strategy pioneered by the TX-2 storage scope editor of continuously displaying a page of text around the current cursor position. Changes to the text or movement of the cursor off the page cause the scope to be repainted. The heart of the editor has been implemented, allowing us to edit text stored in the TSP core memory. The interfaces to the TSP disk and the ARPA network are being worked out.

The basic disk-handling routines have been completed, allowing us to write, read, and delete blocks of data on the disk. These routines keep track of which areas of the disk are used and which are free. Data blocks are identified by 16-bit numbers (actually, the track and sector address of the beginning of the data). We are working on a file system which will allow us to reference the data with character string names.

B. TX-2 System

A number of changes have been made to TX-2 hardware and software in order to provide support for the various current areas of activity.

Drum space allocated on a fixed basis for dedicated-machine projects was reorganized and drastically reduced. The reclaimed space, having a capacity for about 3 million TX-2 words, is now being used for the Speech Data Base.

The magnetic-tape facility was expanded by the addition of a fourth tape unit.

In designing for the incorporation of the disk into the APEX time-sharing system, a study of file size distribution on the drum was performed. Results revealed that while the mean file size is eight pages, a disproportionate number of one-page files (about 23 percent of all files) reduces the median file size to four pages.

Early design work was done on an accounting system to be implemented in APEX. This system would maintain running records of the usage of system resources by the various projects.

The IBM 3830/3330 disk memory has been delivered and is currently being installed on TX-2. The TX-2 I/O channel adapter which emulates an IBM block multiplexer channel is also being completed, and operation of the disk on TX-2 using this channel adapter should begin within the next month.

The TX-2 cycle stealing I/O processor has been operating on a separate main memory bus port for several months with nearly maximal overlap with the TX-2 CPU. Accesses to main memory by the eight I/O processor channels are mapped through the same virtual memory mechanism used by the CPU. This mechanism has been somewhat restructured to simplify system I/O programming, and the first enhancement of channel programming capabilities is to allow channel programs to manipulate directly the contents of the mapping memories in SPAT.

C. TX-2 Password Scheme

In many computer operating systems, a user authenticates himself by typing a secret password known only to himself and to the system. The system compares this password with one recorded in a Password Table which is available only to the authentication program. The integrity of the system depends on keeping the table secret. The TX-2 operating system lacks mechanisms for insuring such secrecy. We have developed a password scheme which does not require secrecy and which is, therefore, better suited to the open environment of TX-2. All aspects of the system, including all relevant code and data bases, may be known by anyone attempting to intrude.

The scheme is based on using a function H which the would-be intruder is unable to invert. This function is applied to the user's password and the result compared with a table entry, a match being interpreted as authentication of the user. The intruder may know both H and the table, but he can penetrate the system only if he can invert H to determine an input that produces a given output.

A paper has been written for submission as a journal article which discusses the issues surrounding selection of a suitable H. A plausible argument is given that penetration would be exceedingly difficult. Apparently, no more rigorous result can be obtained: It appears that any analysis adequate to "prove" the scheme would also lead to enough knowledge to penetrate it.

We plan to implement the scheme for TX-2 password protection in the near future.

REFERENCES

1. A. N. Stowe, W. P. Harris and D. B. Hampton, "Signal and Context Components of Word-Recognition Behavior," *J. Acoust. Soc. Am.* 35, 639-644 (1963).
2. Speech Semiannual Technical Summary, Lincoln Laboratory, M.I.T. (30 November 1972), DDC AD-754940.
3. R. Kaplan, "A General Syntactic Processor," in Natural Language Processing, R. Rustin, Ed. (Algorithmics Press, New York, 1973), pp.193-241.