DOCUMENT RESUME

ED 179 199 IR 007 860

AUTHOR	Harry, D. P.: And Others
TITLE	Voice Interactive Analysis System Study. Final Report, August 28, 1978 through March 23, 1979.
INSTITUTION	LOGICON, Inc., San Diego, Calif.
SPONS AGENCY	Naval Training Equipment Center, Orlandc, Fla.
FEPORT NO	NAVTRAEQUIPCEN-78-C-0141-1
PUB DATE	Jun 79
CONTRACT	N61339-78-C-0141
NOTE	177p.
EDRS PRICE	MF01/PC08 Plus Postage.
DESCRIPTORS	*Computer Oriented Programs: Computer Programs: Data Analysis: Data Bases: *Information Retrieval: *Information Systems: *Man Machine Systems; Military Training: Models
IDENTIFIERS	Minicomputers: *Voice Generated Data

ABSTRACT

The Voice Interactive Analysis System study continued research and develcpment cf the LISTEN real-time, minicomputer based connected speech recognition system, within NAVTRAEQUIPCEN'S program of developing automatic speech technology in support of training. An attempt was made to identify the most effective features detected by the TTI-500 model speech preprocessor, and objective measures were used to demonstrate the presence of, and to evaluate, the various types of information used in LISTEN. Interword timing and structural peculiarities were the twc most useful information sources for the two speakers investigated. Critical examination of the statistical models of the information sources r ,ealed several ways to simplify and improve the LISTEN algorithm. Users manuals fcr analysis programs and for voice reference data generation programs are appended. (Author/CMV)

Technical Report NAVTRAEQUIPCEN 78-C-0141-1

VOICE INTERACTIVE ANALYSIS SYSTEM STUDY

D. P. Harry

J. E. Porter

W. J. Satzer

Logicon, Inc.

Tactical & Training Systems Division
Post Office Box 80138
San Diego, California 92138

June 1979

Final Report 28 August 1978 - 23 March 1979

FOREWORD

Earlier efforts by LOGICON to develop a real-time connected speech recognition system resulted in a system for using hardware designed for isolated word recognition (IWR) but enhanced with connected speech recognition software. This LISTEN system was reported in a series of technical reports referenced herein.

The effort reported here has developed two products to enhance the use of the concept of using high quality acoustical hardware, such as used for IWR, in conjunction with sophisticated software for connected speech recognition. One product is a set of software for formation of voice reference patterns. The second product is a users' manual, included as an appendix here, which details the techniques required to form reliable reference data.

R Breaux

R. BREAUX, Ph.D.
Scientific Officer

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF TABLES

SECTION I

INTRODUCTION

PURPOSE

This report documents the work accomplished and results obtained during the Voice Interactive Analysis System (VIAS) study project.

BACKGROUND

The VIAS study was undertaken as part of a continuing effort to obtain a capability for automatic recognition of connected speech which meets the requirements of the Naval Training Equipment Center (NAVTRAEQUIPCEN) for application in training systems. It is the natural outgrowth of previous projects which led to the development of Logicon's Initial System for the Timely Extraction of Numbers (LISTEN), a minicomputer based, real-time connected speech recognition system.

Projects which led to the development of LISTEN in December of 1977 did not include extensive testing of that system, with the result that at their conclusion the potential of LISTEN to support naval training applications was not unambiguously demonstrated. Good speech recognition accuracy had been obtained for one speaker (MWG), and poor but ambiguous test results were obtained for another speaker (BRO), apparently due to equipment problems or anomalous changes in the second speaker's voice.

At the termination of LISTEN's development it was also very difficult to generate the voice reference data needed to use the system with a new speaker or a new vocabulary. LISTEN relies heavily on processing a large sample of voice data in order to produce a large amount of structural and statistical data descriptive of the speaker's voice, with these data in a form suitable to support real-time connected speech recognition. The voice sample processing programs left after developing LISTEN were mostly dual purpose programs, serving to support both research into the nature of the voice data, and the extraction of voice parameters once those characteristics with promise for recognition had been identified. The processes used included minicomputer programs, programmable calculator procedures, manual graphing and manual calculations. Upwards of forty hours of both minicomputer and manual data processing were required to develop the voice reference data.

The VIAS study was thus undertaken with two main purposes: to further test and analyze LISTEN's performance, and to bring together the collection of voice reference data generation procedures into a coherent set of computer programs which could be delivered to the government. The additional test and analysis of LISTEN was to be based upon a set of computer programs for automatically classifying and gathering performance data. Two auxiliary goals were also attached to the project. First of these was to transfer LISTEN technology from the speech preprocessor (feature extractor) with which it was originally developed to a newer version of that device, as the previous model has gone out of production. Second was the extension upward of LISTEN's initial vocabulary size of eleven words, as far as could easily be managed without major software modification, toward thiıty words.

REPORT OVERVIEW

Five groups of tasks were identified in the VIAS Project Work Plan Report, appropriate to the project goals just described. The relationship of each individual task to this report is described below.

TASK GROUP 1 — TECHNOLOGY TRANSFER. This group included four tasks addressing the problem of transferring LISTEN technology from the Threshold Technology Model VIP-100 speech preprocessor to its replacement, Model TTI-500. Tasks 1a and 1b entailed gathering speech data for a single speaker, and using the previously developed computer program GZEC to discover structure in those data. These tasks are not described in detail, as their purpose was to provide the data for tasks 1c and 1d. Task 1c was a major analytic task, directed toward determining which acoustic features extracted by the TTI-500 preprocessor are most useful for recognition. This analysis is described extensively in Section IV. Task 1d, directed toward verification of the feature selection, is also reported in that section.

TASK GROUP 2 — DEVELOP THE VOICE DATA GENERATION SYSTEM (VDGS). The four tasks in this group brought together the various procedures used for generating voice reference data to support real-time speech recognition by LISTEN, in the form of a unified body of computer programs and a user's manual. Tasks 2a, 2b and 2c entailed programming tasks and are not reported upon further. Their end result is the VDGS, a set of computer programs constituting a separate deliverable of this project. The fourth task, 2d, was to produce a users guide for the VDGS, which is introduced in Section II of this report, and included in its entirety as Appendix A.

TASK GROUP 3 — EXPAND VOCABULARY. The single task in this group was executed in conjunction with tasks 2a, 2b and 2c. It entailed increasing the maximum number of vocabulary items which can be accommodated by the individual programs of the VDGS, when practicable, toward thirty words. Results obtained are discussed in Section II, in connection with the VDGS.

TASK GROUP 4 — DEVELOP PERFORMANCE ANALYSIS SUBSYSTEM (PASS). The four tasks in this group entailed the design, implementation and application of a r w set of computer programs for collecting and organizing data about LISTEN's recognition performance. Also, the initial task, 4a, was directed toward converting the real-time recognition components of LISTEN (the programs LTRGEN, MEX and MINT) to opeate in a new computer (Data General S-130) and speech preprocessor (TTI-500) environment. The programming tasks, 4a, 4b and 4c are not described further, as their end result is the set of programs comprising PASS, a separate deliverable. The programs are, however, introduced in Section III of this report, and a Users Guide for those programs is included as Appendix B. Task 4d, the application of some elements of the PASS to automatically classify recognition errors committed by LISTEN, is described in Section IV.

TASK GROUP 5 — CRITICALLY EXAMINE INFORMATION SOURCE MODELS. The four tasks in this group were directed toward a detailed examination of the strengths and weaknesses of LISTEN, by determining the relative importance of the various information sources used in that system to achieve recognition. As these were all analytical tasks, they are discussed extensively in Section IV.

KNOWLEDGE OF LISTEN ASSUMED. As LISTEN is a complex system, based on some unique approaches to obtaining automatic recognition of connected speech, this report would become excessively long if the principles and details of operation of LISTEN were described in a self-sufficient way here. The remainder of this report is therefore written assuming the reader has an understanding of LISTEN, to a level of detail easily accessible in the final reports of the projects which led to its development. For convenience, these reports are identified below.

a. Use of Computer Speech Understanding in Training: A Preliminary Investigation of a Limited Continuous Speech Recognition Capability; Technical Report NAVTRAEQUIPCEN 74-C-0048-2; Logicon, Inc.; June 1977.

b. LISTEN: A System for Recognizing Connected Speech Over Small, Fixed Vocabularies, In Real Time; Report NAVTRAEQUIPCEN 77-C-0096-1; Logicon, Inc.; April, 1978.

SECTION II

THE VOICE DATA GENERATION SYSTEM (VDGS)

DESCRIPTION

The VDGS consists of a collection of computer programs for collecting and processing voice data to generate the voice reference data necessary to recognize connected speech in real time with LISTEN. The end product of these programs is a large set of data in the format of a standard Data General data file, called the MIND file.

The twenty-nine programs comprising the VDGS are written in FORTRAN IV, FORTRAN V and Data General Assembly Languages. The programs are capable of, and intended for, use on a Data General S-130 minicomputer equipped with at least 32K words of memory, a 10-megabyte disc, the RDOS operating system and standard peripherals. They may, however, be recompiled for execution on other Data General minicomputers, such as the Nova 3.

Appendix A is a Users Manual for the VDGS. It contains instructions whereby a qualified speech research technician, familiar with the principles and details of LISTEN's operation, can collect speech data (given the necessary equipment) and produce a MIND file for use with LISTEN.

The VDGS contains all programs necessary to collect speech data and produce a MIND file. Of the twenty-nine programs, twenty-four must be used in this process. The remaining five programs are often useful, but in general are not needed to produce MIND files. All manaual and extra-computer procedures required to generate voice reference data prior to this project have been automated and implemented as programs in the VDGS. However, human surveillance and occasional modification of the generated data are essential if the recognition performance of LISTEN is to be optimized. The VDGS therefore exists in two forms: as a collection of independent programs for individual execution, and as a "pushbutton" system requiring a minimum amount of human intervention, known as CHAINMIND.

CHAINMIND consists of three segments; EXTRACT, GENTL and MAKEMIND. EXTRACT is a program which facilitiates gathering speech data samples in a format suitable for use by the remainderof VDGS. It includes prompting of the speaker via the CRT display, with utterance contents taken from files provided. Since the voice data are usually taken over several separate recording sessions, (perhaps over several days), there is a natural division of the MIND file generation process at that point where all necessary speech data have been recorded on disk, and the voice data processing can begin.

Another reason the EXTRACT process is kept separate from the remainder of the CHAINMIND version of VDGS is that a decision must be made at that point with regard to separating the collected voice samples (which consist mostly of multiple word utterances) into individual vocabulary items. This can be done either automatically or manually. However, initial results using the automatically generated individual vocabulary examples indicates that reasonable recognition results cannot be obtained in this way. (See Section IV for specifics about Example Set generation.) The User's Manual and the VDGS contain instructions and aids for generating vocabulary item examples manually.

The second segment of the CHAINMIND version of the VDGS, GENTL, consists of programs which culminate in the generation of Transition Letter Sets for each vocabulary item. Although this process needs no human intervention, the Transition Letter Sets are so fundamental to the successful operation of LISTEN that prudence dictates that they should be examined and in some cases modified before continuing the MIND file generation process. This is particularly true since the method used to generate Transition Letter Sets (the algorithm GENRLIZ in the program GZEC) is heuristic in nature and subject to the influence of extraneous details, such as the order in which speech samples are presented to it.

The third segment of CHAINMIND contains the majority of the programs and requires the majority of processing time. Here too, prudence dictates human surveillance of every step of the process if LISTEN's performance is to be optimized. Critical points at which intervention may be required cannot be identified at this time, as only a small number of speakers' data have been processed. The Users Manual contains some suggestions and remarks which may be helpful in identifying anomalies.

For reasons mentioned above, it is recommended that the VDGS be used as a collection of individual program elements in accordance with the Users Manual, with careful scrutiny of results at every step.

VOCABULARY SIZE

LISTEN was initially developed for an eleven-word vocabulary, and many of the programs now included in the VDGS were developed to operate with about that many vocabulary items. Under Task 3a, the vocabulary capacity of many of these programs has been extended toward thirty, and programs developed during this project for inclusion in the VDGS have, as far as possible, been constructed to accommodate the larger vocabulary.

The tabulation below gives the program name and the vocabulary size capability of the individual programs in the VDGS, as delivered.

EXTRACT	any	INVERT	30	MUTE	30
ESG	13	CROAK	15	GLOVE	30
GZEC	13	REVEX	13	TAILOR	30
RESCUE	30	ADDER	13	BUILDER	any
SIGH	30	AVRAJ	13	DEALER	13
LOOPER	13	CRAP	13	PHEW	30
REVEXA	13	GAPSTER	13	GASP	15
RVDIT	13	SORTA	any	ESDIT	11
COVERT	15	SORTB	any	ESGDIT	13

Extending the vocabulary size capability of the programs in VDGS which are not yet capable of handling thirty vocabulary items would require more or less extensive modification of those programs. Doing so during this project was judged an inappropriate use of project resources, as discussed in the Work Plan Report. In this connection, it should be noted that two of the three programs needed for real-time recognition (MEX and MINT) are limited to thirteen vocabulary items, and modification of one of them (MEX) to accommodate a larger vocabulary would require several labor months of effort. The difficulty of such an extension stems from the complexity of the MEX data structure, not from any inherent limitation of the recognition algorithm.

SECTION III

THE PERFORMANCE ANALYSIS SUBSYSTEM (PASS)

The PASS consists of three computer programs for collecting, processing and plotting data useful in analyzing many aspects of LISTEN's performance. These programs are supported by a special version of LISTEN in which the MEX program produces data files in the format required for processing by the PASS program BIGMINT. The other two PASS programs (STATSUM and LICVAT) operate on data provided by BIGMINT.

The programs in the PASS operate in the same environment as LISTEN and the VDGS.

Appendix B is a Users Manual for the PASS. It contains instructions for using these programs for extracting and processing data from files produced by LISTEN. Section IV of this report describes several analytical investigations which were based on data derived and processed by the PASS. Those analyses are thus examples of the varied possible uses of data generated by the PASS.

Specific information elements developed by programs in PASS are discussed below.

BIGMINT. This program provides the following data:

a. An annotated listing of the entire MIND file.

b. Date and identification of the MEX-generated file used to produce the following data items for each utterance in that file.

c. Compressed speech data file identifier.

d. Potential recognitions detected by MEX, with the following data for each potential recognition.

(1) Machine type

(2) Vocabulary item

(3) T-state counter statistic QT

(4) L-state counter statistic QL

(5) Start time

(6) Recognition time

(7) Associated vocabulary items and forms

(8) a priori cost

(9) Violation category cost

(10) QT cost

(11) QL cost

(12) Total cost reported by MEX

(13) Association cost

(14) Total cost assigned to the potential recognition in MINT

(15) Identification of optimal predecessor

(16) Interword gap cost to optimal predecessor

(17) Total cost from this node upward along optimal path.

e. Costs for all interword gaps between potential predecessors.

f. Identification of the ten lowest cost paths through the graph of the utterance, and for each:

(1) whether correct or incorrect

(2) total cost

(3) vocabulary items

(4) nodes.

g. Vocabulary items actually spoken.

h. For the entire file of utterances, the number correctly and the number incorrectly recognized.

STATSUM. This program provides the following data:

a. An annotated listing of the entire MIND file.

b. For each utterance processed by BIGMINT, the index of the utterance within the MNSET, identifier of the compressed speech data file, and what was acutally spoken.

c. For each of the ten best paths through the graph of the utterance, in increasing order of total path cost,

(1) whether correct or incorrect

(2) total a priori cost

(3) total violation cost

(4) total QT cost

(5) total QL cost

(6) total of costs reported by MEX for all nodes of the path

(7) total association cost

(8) total of costs assigned by MINT to all nodes of the path

(9) initial delay cost

(10) total interword gap cost

(11) final delay cost

(12) total interword timing cost

(13) total of all costs for the path

(14) nodes of the path.

d. Category and type of the recognition problem posed by this utterance (as defined in Section IV).

e. The difference in all costs listed in c. above, between all incorrect paths and the best correct path, or an indication that no best path exists.

LICVAT. This program provides the information listed below. Several of the quantities mentioned are defined in Section IV.

a. An enumeration of utterances in Category 0, with identification of the compressed speech data file and the index of the utterance within the MNSET.

b. The enumeration of utterances in Category 1, of type other than (0,1), (1,0) or (1,1). For each utterance the following data are given:

(1) compressed speech data file identifier

(2) utterance index within its MNSET

(3) Whether it was correctly or incorrectly recognized

(4) all costs enumerated in item c. for the program STATSUM, for the best path

(5) utterance type.

c. An enumeration of utterances in Categories 2 and 3, with compressed speech data file identifier and utterance index within its MNSET

d. An enumeration of utterances in Category 1 in which the best incorrect path is a direct start-to-end node connection (corresponding to no spoken word), with compressed speech data file identifier and utterance index within its MNSET.

e. A list of all utterances in Category 1, ordered on the basis of the difference in costs of various kinds between the best incorrect and the best correct path. (The M value defined in Section IV). For each utterance the following data are given:

(1) compressed speech data file identifier

(2) utterance index within its MNSET

(3) cost difference (M value)

(4) whether correctly or incorrectly identified

These ordered lists are generated for each of the cost contributions mentioned in connection with program STATSUM, item c.

f. A computε generated plot of the cumulative distribution of M values, for all utterances and for incorrectly recognized utterances only. Histogram data are also given for M increments of 10, for all utterances and incorrectly recognized utterances only.

g. All data described in e. and f. above, but restricted to utterances in Category 1 of the following types:

(1) (0,1)

(2) (1,0)

(3) (1,1)

h. For all real recognitions, counts of the number of associated recognitions of each vocabulary item and form, and the total number of associated recognitions, by vocabulary item and form.

i. As in h., but for all artifactual recognitions.

j. For all real recognitions, counts of occurrences of each violation category, and the total overall violation categories, by vocabulary item and form. Also, the total number of real recognitions (overall vocabulary item) of each violation category.

k. As in i. above, but for all artifactual recognitions.

l. For all real recognitions, a computer-generated plot of the cumulative distribution of the QL linearizing function, f, with histogram data.

m. As in l. above, but for all artifactual recognitions.

n. The following data about initial delays, categorized by real vice artifactual recognition and vocabulary item and form:

(1) total number of recognitions in the category

(2) number of zero delay values

(3) fraction of cases which were zero

(4) the average of the non-zero initial delays

o. As in n. above, but for final delays.

p. A computer generated plot of the cumulative distribution of the interword gap normalizing function, f, for all interword gaps between contiguous real recognitions.

q. As in p. above, but for all interword gaps between recognitions and their potential predecessors, other than contiguous real recognitions.

SECTION IV

ANALYSES

The analyses performed in the VIAS study are reported in this section. These analyses were major parts of Task Groups 1 and 5 and a minor part of Task Group 4. The analyses fall naturally into two categories. The first category (Task Group 1) is concerned with transferring the connected speech recognition capability developed with the VIP-100 speech preprocessor to its successor, the TTI-500. The second category (Task Groups 4 and 5) is concerned with a critical examination of the LISTEN speech recognition algorithm, to determine its strengths and weaknesses, in hopes of discovering fruitful approaches to improving its performance and easing the task of applying it in automated training systems.

Underlying each type of analysis were several steps, starting with determination of the types of data needed, design of an algorithm for extracting the data, implementation of a program or program segment for extracting the data (as part of the Performance Analysis Subsystem, PASS) and, finally, extracting and analyzing the data thus obtained. In the description of the analyses presented below, only the nature of the data used, the data itself and the analysis of the data are discussed. Designing, implementing and exercising the relevant portion of the PASS are not discussed, although those efforts consumed a significant portion of project resources. Instructions for using the PASS to develop data of the type presented in connection with the following analyses are given in Appendix B, the PASS Users Manual.

The remainder of this section is divided into five parts addressing:

a. The experimental bases used in the analyses.

b. The transfer of technology (Task Group 1).

c. The contribution of each information source to recognition (Task 5d).

d. The analysis of recognition errors (Tasks 4c and 5d).

e. The critical examination of information source models (Tasks 5a and 5b).

EXPERIMENTAL BASES FOR THE ANALYSES

Voice data were collected, and MIND files were created, for two new speakers (LHN, JEP) during the course of this project. Voice data for testing were also collected for these two speakers, and LISTEN was exercised on these data. Finally, the performance analysis programs in the PASS were exercised on output obtained from LISTEN for speakers MWG, LGN and JEP. In this way the programs of the VDGS and the PASS were validated, and data were generated for the analysis tasks of the project.

Speech data were collected in several (five to ten) sessions over a few days. After collecting all of the data from each speaker, it was divided into three equal parts called Training, Interim Test and Test data. These terms were inherited from the LISTEN development project wherein the second set of data was used to test some initial concepts. In this project the Interim Test data were simply used to extract certain speech characteristic data not obtainable from Training data.

Each set of data consisted of six "Magic Number Sets". Each of these is fifty-five utterances of from one to four words, arranged in a format which makes the numbers appear to the speaker to be quite random. They are actually a carefully balanced set of vocabulary items, combined in such a way that each digit occurs an equal number of times, and the word "point" nearly as often, and so that each transition between vocabulary items appears exactly once.

Each major data set (Training, Interim Test and Test) thus consisted of three hundred thirty utterances containing one thousand fifty words. Training data were used to generate structural characteristics of each vocabulary item (Transition Letter Sets and Loop Letter Sets), and some statistical properties of the voices were extracted from Interim Test data. Test data were used only for testing purposes. Most results in the following are therefore based on Test data. The exception is the investigation of statistical models, where data are compared for Interim Test data and Test data, to determine the validity of certain statistical assumptions.

The voice data taken from the two new speakers were processed differently, with qualitatively different recognition performance results, as is described in the following paragraphs. New data for only one speaker (LHN) were therefore usable in the detailed analyses of LISTEN performance.

An experimental variable of considerble interest in connection with the VDGS was also investigated in this study. This variable, Example Set Generation, relates to the way in which speech data are separated into sets of examples of individual vocabulary items. This step is necessary for the generation of Transition Letter Sets by the program GZEC. Two approaches have been used to segment the samples of connected speech. Originally, computer printouts of speech preprocessor data were scanned by eye, and segments within each utterance which contained each individual vocabulary item were identified visually and recorded manually. These segments were selected to contain the vocabulary item w..n high confidence, but with as little additional material as possible, in the judgement of the person marking the data. This remains the recommended way to produce the needed example sets.

As described in Reference 1, an automatic method of generating sets of individual vocabulary samples has also been developed. This procedure, embodied in the program ESG (Example Space Generator) applies statistics derived from MWG's voice to excise segments of a multiword utterance which contain individual vocabulary items with high confidence. This, of course, entails a tradeoff between taking a large segment including extraneous material, and taking a smaller segment with attendant increased risk of excluding some portion of the spoken word. Since the nature of GZEC makes it much more sensitive to the deletion of parts of words than to the inclusion of extraneous material, the safety fa-tors used in ESG (to accommodate statistical variability in articulation and still extract unclipped vocabulary item examples) are quite high. Not surprisingly, the safety factors used in ESG make the word length statistics derived from MWG's voice data apply to other speakers as well. Using ESG to generate vocabulary item samples is therefore an alternative to doing so manually when the VDGS is applied to a new speaker. This is the additional experimental variable which was investigated in this project, by using the manual procedure for LHN's voice data and the automatic procedure embodied in ESG for JEP's voice data.

Although the transition letter sets generated by these methods do not appear qualitatively different (see Figures 1 and 2), the recognition performance obtained using ESG was significantly inferior to that obtained by using the manual procedure, as the following data show:

Speaker	Speech Data	Example Generation Method	% Utterances Correct
MWG	Interim Test	Manual	94
MWG	Test	Manual	86
LHN	Interim Test	Manual	74
LHN	Test	Manual	70
JEP	Interim Test	Automated (ESG)	42
JEP	Test	Automated (ESG)	44

Detailed examination of LISTEN's performance for speaker JEP shows that the Transition Letter Sets are not effective; MEX very frequently does not detect a word actually spoken as potentially present in the utterance. This occurs relatively infrequently for MWG and LHN. The difference in Transition Letter Sets is presumably due to the extraneous material in the set of examples from which the JEP Transition Letter Sets were derived.

The failure of MEX to detect the potential occurrence of an actually spoken word also seriously perturbs the voice reference data extraction process, which contributes to the poor performance. (Notice that recognition results for JEP are better on Test data than on Interim Test data, even though statistical data are extracted from the former.) For these reasons, the data for JEP, while contributing a significant result to the project as a whole, were not used in the detailed examination of LISTEN as its performance with bad reference data is not indicative of its true potential.

```
TLS No.     "ZERO"

   1    ; 0000   0   0000;
   2    ;10000   00  0000;
   3    ; 0000 1 00000000;
   4    ; 000      000  00;
   5    ;1 00 1    000 000;
   6    ;1 00 1    000 000;
   7    ;1 00 1   0000 000;
   8    ;1 00 1   0000 0 0;
   9    ;1 00 1   0000 0 0;
```

```
TLS No.     "ONE"

   1    ;         000000000 0;
   2    ;      1  000000000 0;
   3    ; 1 11000000000 0;
   4    ;01   1100000000 0;
   5    ;01 0 1 0000000 0;
   6    ;0  0 1 0 00000 0;
   7    ;0  0       00000 0;
   8    ;0  0     000000000;
   9    ;0 00    00 000000;
  10    ;0 0     000000000;
```

```
TLS No.     "TWO"

   1    ;   00    0 000 00;
   2    ;    0      000000;
   3    ;   000  00000 00;
   4    ;  0000  0   0 00;
   5    ;  000   00000000;
   6    ;  000   00000 00;
   7    ;  000     000 000;
   8    ;  000     0000 00;
   9    ;  00      0000000;
  10    ; 000    000000000;
  11    ;  0       0 000 0 ;
```

```
TLS No.     "THREE"

   1    ;  00          0 000;
   2    ;   00    0 0 0 00;
   3    ;  000   0     0000;
   4    ;  0000  0     0000;
   5    ;  0000  0     0 00;
   6    ;  00     00000000;
   7    ;  00     0000 000;
   8    ;  00      0000000;
   9    ;  0           000  0;
  10    ;  0         000 0 0;
```

```
TLS No.     "FOUR"

   1    ;1     0000000000 0;
   2    ;1     0000000000 0;
   3    ;1 01  000000000 0;
   4    ;1 0     00000000 0;
   5    ;1 0 1  00000000 0;
   6    ;  0        0000000 0;
   7    ;    0         000 0 0;
   8    ;    0         000 0 0;
   9    ;   00        000    0;
  10    ;   00      00000   0;
  11    ;              0 000 00;
```

```
TLS No.     "FIVE"

   1    ;           0000000 0;
   2    ; 1       000000000 10;
   3    ;01 0 1 000000010;
   4    ;0110      0000000 0;
   5    ;01100 1 000000 0;
   6    ; 1 00 1 0000 0 0;
   7    ; 1 00      000    0;
   8    ;    00      0000000;
   9    ;   000    00000000;
```

```
TLS No.     "SIX"

   1    ; 000000001110000;
   2    ;  0000 00     0000;
   3    ;  000    00  0 00;
   4    ;1 00   1 0000   00;
   5    ;1 000 1100000 00
   6    ;1 0000 100000 00;
   7    ; 000000   0000000;
   8    ;  0000   00000000;
   9    ; 0000             0000;
  10    ; 00 00  0     0000;
```

```
TLS No.     "SEVEN"

   1    ;000000001 10000;
   2    ;  000   00    0000;
   3    ;  000    00        0;
   4    ;11 00 1 0000      0;
   5    ;11 00 1   000      0;
   6    ; 1 00 1   000 0 0;
   7    ; 1 0        0000 0 0;
   8    ;     0 1 00000 0 0;
   9    ;1 00  1 0 0000000;
  10    ;   00 1 0000000 0;
  11    ;   00 1 00000 0 0;
  12    ;    0     00000 0 0;
  13    ;0            000000 0;
  14    ;0 0       000000000;
  15    ;0 00      00000000 ;
```

```
TLS No.     "EIGHT"

   1    ;  000     0000     0;
   2    ;  0000   00000 00;
   3    ;  00000 00000 00;
   4    ;  00000   0000 00;
   5    ; 000000   0000000;
   6    ; 00000000 000000;
   7    ; 000   00        0000;
```

```
TLS No.     "NINE"

   1    ;0   0       000000 0;
   2    ;01100 1   00000 0;
   3    ;01100 1 000000 0;
   4    ;01  00 1100000000;
   5    ;0   00      00 00000;
   6    ;0 000        0 00000;
   7    ;0 0   00    0000000;
   8    ;   0000     0000000;
```

```
TLS No.     "POINT"

   1    ; 1 1 000000000 0;
   2    ; 1 1 0000000010;
   3    ;     11 00 0000010;
   4    ;01    11 0 00000 0;
   5    ;01 0 110 00000 0;
   6    ;01 00 1   00000 0;
   7    ;0   000 100000000;
   8    ;   0000      0000000;
   9    ; 00 000  0   0000;
  10    ;00   0 00     00000;
  11    ;0 0000000     0000;
```

```
TLS No.      "ZERO"

 1   ; 00000 001 10000;
 2   ;  000   00000 00;
 3   ;1 000  1 000   0;
 4   ;1 000 11 000   0;
 5   ;1 000 1  0001  0;
 6   ;1 000 1  000 0 0;
 7   ;1100 1   000 010;
 8   ;1100   000000010;
 9   ; 1    00 00000 0;
10   ;      00  0 00 0;
11   ;      00    00 0;
```

```
TLS No.      "ONE"

 1   ;11  0000000000 0;
 2   ; 1    00000000 0;
 3   ;    0 1  0000 0 0;
 4   ;  0       000   0;
 5   ;  0       000 000;
 6   ;        0 00 000;
```

```
TLS No.      "TWO"

 1   ;0  0000000000000;
 2   ;  0000000   0000;
 3   ;  00000000 00000;
 4   ;  0000  00000 00;
 5   ;  0000 100000 00;
 6   ;1 000   0000  00;
 7   ;  000      000 00;
 8   ; 000   00 000000;
 9   ;  0    00 0 0000;
```

```
TLS No.      "THREE"

 1   ; 00000000  00000;
 2   ;  00000    00000;
 3   ;  0000 000000000;
 4   ;  00    00000000;
 5   ;  00  1 0000 0 0;
 6   ;1 00111 0000 0 0;
 7   ;1 00 11 000010 0;
 8   ;1 000 1100001  0;
 9   ;1 0000 100000 00;
10   ;1 0000  00000 00;
11   ; 000 00 00000 00;
12   ;    000     0 00;
```

```
TLS No.      "FOUR"

 1   ;11  0000 00000 0;
 2   ;1   0000000000 0;
 3   ;1    000000000 0;
 4   ;1 0  000000000 0;
 5   ;  0   00000000 0;
 6   ;  0        000 0 0;
 7   ;  0        000 0 0;
 8   ;            00 0 0;
 9   ;         0   0  0;
```

```
TLS No.      "FIVE"

 1   ; 1     0000000010;
 2   ;011     000000010;
 3   ;011 0 1000000010;
 4   ;011 0 1  0000010;
 5   ; 11 00 1 000 0 0;
 6   ; 11000 1 000 0 0;
 7   ;  00    0000  00;
 8   ;  0 0      000000;
 9   ;    0    0 0   00;
```

```
TLS No.      "SIX"

 1   ;  000000 1110000;
 2   ;  0000000 1 0000;
 3   ;  0000  0     00;
 4   ;  000   0   0 00;
 5   ;  0000 100000 00;
 6   ;  0000  00000 00;
 7   ; 00000   0000000;
 8   ;00000      000000;
 9   ;0 00          0000;
10   ; 0000     1  0000;
11   ; 0000        0000;
```

```
TLS No.      "SEVEN"

 1   ; 000000001110000;
 2   ;    000000   0000;
 3   ;11 0001 0000 0 0;
 4   ;11 00 1 0000 0 0;
 5   ;11 001  0000 0 0;
 6   ;11 0 11  000 0 0;
 7   ;11000 1 0000 0 0;
 8   ;1100      000 0 0;
 9   ;1 0      0000 000;
10   ;          0000000;
```

```
TLS No.      "EIGHT"

 1   ;   00    00      0;
 2   ;1  000  0000     0;
 3   ;1  000  100000 00;
 4   ;1 00000 00000 00;
 5   ;   000  00000000;
 6   ;  0000    0000 00;
 7   ;  0 000   0000 00;
```

```
TLS No.      "NINE"

 1   ;  0 0   0000 0 0;
 2   ; 1 0  1 0000 0 0;
 3   ; 1 00 1 0000 0 0;
 4   ;01 00 110000 0 0;
 5   ; 1 00011 000 0 0;
 6   ; 1 000 1 00000 0;
 7   ; 1 000 1 0000000;
 8   ;110000 100000 00;
 9   ;     000  0000 00;
10   ;     00 0 0000 00;
```

```
TLS No.      "POINT"

 1   ;     0000000000 0;
 2   ;1111000000000010;
 3   ; 1 1100000000010;
 4   ; 10 1 0000000010;
 5   ; 100    0000000 0;
 6   ; 1 0  1  000 0 0;
 7   ;1100      000   0;
 8   ; 10 0     000   0;
 9   ;1  0   00000   0;
10   ;   0000 0 000 00;
```

Figure 2. Transition Letter Sets (TLS) for Vocabulary Items "ZERO" tnrough "NINE" and "POINT" for Speaker JEP, Generated from Manually Produced Examples

TRANSFER OF TECHNOLOGY

The LISTEN connected speech recognition system was developed using Threshold Technology Corporation's speech preprocessor Model VIP-100, which is no longer being produced. Its successor, the Model TTI-500 is based on a similar principle of operation, provides output which is identical to that of its predecessor in terms of the electrical interface and digital format, and is expected to be available for a considerable time into the future. It is therefore both feasible and desirable to bring LISTEN into accomodation with the newer version of the speech preprocessor.

The principal difference between the older and newer preprocessor is the acoustical significance of some of the speech features recognized by the respective devices. Only eight features are common to the two devices. In both cases thirty-two features are determined to be either present or absent at a nominal rate of 500 times per second, and this determination is encoded as two sixteen-bit binary words, transmitted to the central processor during detected periods of speech. As LISTEN was purposefully developed to discover and to recognize patterns in a stream of binary data, without recourse to the acoustic significance of the data, very little change in LISTEN is required to accommodate the new preprocessor. As LISTEN uses only sixteen of the thirty-two bits, or features, received from the preprocessor every two milliseconds, the only requirements to adapt LISTEN to the new preprocessor are to select which sixteen of the available thirty-two features to use, and to change the interface accordingly. The analysis required in support of the transfer of LISTEN technology to the new preprocessor thus reduces primarily to selecting the features to use and secondarily to verifying the selection.

FEATURE SET SELECTION. The only constraint which must be met in selecting sixteen of the thirty-two features available is that the long pause feature, LP_4, which indicates the end of an interval of vocalization must be among them. Any other fifteen features could be used in conjunction with LP_4. The vocalization indicator, LP_4, must be included in the selected features because it is used as the indicator for end of utterance processing in LISTEN. The problem is thus reduced to selecting fifteen features among thirty-one available.

Using a single feature several times, i.e., forming a sixteen-bit computer word by selecting less than fifteen features (plus LP_4) and setting several bits equal to a single feature indicator, has no utility. This is because LISTEN is sensitive only to the information content of each feature position, so that the same results would be obtained by using a smaller number of features, each represented only once. Since adding <u>different</u> features to a pre-existing set of distinct features has the potential (at least) of increasing the available amount of information about what was spoken, only sets of fifteen distinct features need be considered.

An "ideal" method for selecting the set of features to be used would be to directly evaluate the recognition performance obtained with alternate sets of features. Any other method must be considered to be indirect, and must be based on some assumptions about how the recognition performance would be affected by different feature characteristics. Direct evaluation of even a few alternative feature sets is quite impractical, however, as more than forty hours of computer processing time is the minimum required to evaluate recognition performance. Since there are over three hundred million subsets of fifteen items taken from a set of thirty-one items, some method of preselecting a (very much) smaller collection of alternatives must be used anyway. Practical necessity therefore drives one to an indirect method of selection.

One indirect method of feature selection is to refer to authority. In this case the unquestioned leading authorities on the acoustical significance of features are the personnel at the preprocessor manufacturing facility, where the circuitry for extracting the available features was developed. The manufacturer (Threshold Technology, Inc.) most cooperatively suggested a set of fifteen features which, in their judgement, would work well in the LISTEN environment. Since LISTEN is a complex algorithm which had not been thoroughly tested at the time, the manufacturer's suggested set of features must be regarded as an informed opinion rather than a definitive solution to the problem. This opinion is based on extensive testing of many different features. (Presumably in the context of isolated word/phrase recognition, which, while different in many practical respects from connected speech recognition, should nevertheless exhibit similar sensitivity to the utility of a feature for recognition.) The set of features suggested by the manufacturer is the set ultimately selected for use in LISTEN, for reasons described in the following discussion.

An attempt was made to measure objectively the utility of each feature for recognition. The approach used was to posit several different measures of feature "quality", obtain values for these measures and analyze the results. The measures posited were based on plausible judgements about observable characteristics of a feature which carries a large amount of information which would be useful for distinguishing among vocabulary items.

This approach to evaluating features suffers several shortcomings, in spite of its intuitive appeal. Most serious of these shortcomings, perhaps, is the questionable nature of the assumption that features can be evaluated individually. The recognition procedure used in LISTEN is based upon detecting the simultaneous presence, or absence, of several features in the preprocessor output. It is therefore possible that there is no measure of effectiveness of individual features, and only the effectiveness of sets of features can be given concrete meaning. The actual situation is probably intermediate between the inherent extremes. That is, there probably are indicators of individual feature utility such that selecting those fifteen features with highest utility produces an excellent, but not necessarily the best possible, choice.

Another difficulty with evaluating quality measures of features is a practical one. Data must be collected from a particular speaker or speakers, speaking phrases from a particular vocabulary, raising the difficult question as to how valid the results might be for other speakers and other vocabularies. In the VIAS project the available resources allowed examining data collected from a single speaker (LHN), speaking only the digits. (The primary limitation here was labor required to do the analysis in a timely manner, as voice data were available from several other speakers.)

A third difficulty with basing feature selection on evaluation of some intuitively appealing measures of feature quality is that the quality measures themselves are entirely ad hoc, as it is not practical to test the quality measures for the same reasons that it is not practical to evaluate alternative selections of features.

On the positive side, there is a possibility that meaningful individual feature quality measures can be posited and their evaluation may give some clear indication of the utility of at least some features. The quality measure approach was followed in the hope that this would be the case.

Quality Measures. Six measures of individual feature quality were posited and evaluted. One of these (VFO) is defined in terms of the frequency of occurrence of a feature in various vocabulary items. The other five are attempts to measure the amount of reliable "structure" — reliably occurring sequences of feature-present/feature-absent zones — which exist in a large sample of vocalization of a given vocabulary item.

The quality measures were evaluted on a data set extracted by the TTI-500 while the subject (LHN) spoke various vocabulary items in connected combinations. Individual vocabulary items were visually identified within computer printouts of the features detected by the preprocessor. Distributing the segmented connected speech data into example sets for each vocabulary item provided the data base needed for evaluating the quality measures.

In order to evaluate quality measures other than the first, some way to recognize and extract reliably occurring patterns of a feature's history within each vocabulary item was needed. The program GZEC, incorporating the algorithm GENRLIZ, was used for this purpose. (This program and algorithm are part of, and described in connection with, the VDGS, and in previous LCSR project reports.) GZEC was utilized two times for each vocabulary item (in this case just the digits 0-9), first operating on data containing the manufacturer's recommended set of fifteen features (hereafter called the Initial Feature Set), and second operating on data containing only the other sixteen features. GZEC extracted Transition Letter Sets from sixty-six examples of each vocabulary item. These Transition Letter Sets exhibit the pattern with which each feature occurs reliably in the sample of sixty-six vocalizations of each vocabulary item. The feature quality measures (other than the first) are therefore defined in terms of the patterns the feature follows, as revealed in the Transition Letter Sets for each vocabulary item.

Since the Transition Letter Sets obtained for a collection of utterances is dependent upon interaction between features, it cannot be assumed that this method of processing treats all features identically. Attempts to eliminate this potential bias are frustrated by the fact that GZEC can process at most sixteen features in a single run, and there are hundreds of millions of ways to select sixteen features from the available thirty-one.

Definition of Feature Quality Measures. The individual feature quality measures used in this investigation are:

a. Variance of Frequency of Occurrence (VFO). If a feature occurs very frequently in some vocabulary items, about half the time in others, and very infrequently in still others, that feature would be useful for distinguishing among some vocabulary items. The quantity VFO measures the vocabulary item dependent variability of frequency of occurrence of a feature. It is the variance, across vocabulary items, of the average frequency of occurrence of the feature in each vocabulary item. It is determined by the equation:

$$\mathrm{VFO} = \frac{1}{|V|} \sum_{v \in V} (\mu_v - \mu)^2$$

where $|V|$ is the number of vocabulary items

μ_V is the average frequency of occurrence of the features in samples of vocabulary item V

μ is the average of μ_V over all vocabulary items v

b. Frequency of Zero and One (F01). Each feature position in the Transition Letter Sets indicates the reliably occurring pattern of development of that feature in the word. Each Transition Letter Set indicates that, at its corresponding point in the word, the feature is either reliably present (indicated by 1), reliably absent (indicated by 0), or not reliably either present or absent (indicated by a blank). If a feature has a rich and reliable pattern of occurrence and/or non-occurrence in a word, then the number of 0's and 1's for that feature is large compared to the number of blanks. The average fraction of Transition Letter Set occurrences which are zero or one is

$$\mathrm{F01} = \frac{1}{|V|} \sum_{v \in V} \left(\frac{1}{|T|_v} \sum_{i=1}^{T_v} [\#_0\ (T_{i,v}) + \#_1\ (T_{i,v})] \right)$$

where $|V|$ is the number of vocabulary items

$|T|_v$ is the number of Transition Letter Sets found by GZEC for vocabulary item v, and

$\#_k\ (T)$ is 1 if the feature in question has value k (0 or 1) in Transition Letter Set T, and 0 otherwise.

c. **Vocabulary Variance of Frequency (VVF). While a feature with low frequency of required presence or absence (low F01) must have marginal utility for recognition, variability over vocabulary items of the frequency of required presence or absence might indicate high vocabulary item dependence of the feature. Thus VVF is defined to be the variance over vocabulary items of the frequency with which the feature is required to be present or absent. That is, VVF is the variance of F01 determined for each vocabulary item.**

$$\mathrm{VVF} = \frac{1}{|V|} \sum_{v} (\mathrm{F01}_v - \mathrm{FC1})^2$$

where $$\mathrm{F01}_v = \frac{1}{|T|_v} \sum_{i=1}^{|T|_v} [\#_o(T_{i,v}) + \#_1 (T_{i,v})]$$

and other quantities are defined in (b) above.

d. Average Number of Zero or One Zones (ANZ). Each feature tends to vary rather regularly within a word, exhibiting zones where the feature is reliably present or absent, bordered by zones where its occurrence is unpredictable. The number of zones wherein the feature is either reliably present or absent is an indication of structure as found for the vocabulary item by GZEC.

$$\mathrm{ANZ} = \frac{1}{|V|} \sum_{v \in V} Z_v$$

where $|V|$ is the number of vocabulary items, and

Z_v is the number of zones of required presence or absence of the feature, as exhibited in the Transition Letter Sets for vocabulary item V.

e. Average Number of Zero/One Zone Reversals (ANR). As an indicator of the richness or complexity of the reliably occurring pattern of a feature within a word, one can count the number of reversals between zones of required absence and required presence of the feature, ignoring any intervening zones where the feature is not reliably present or absent. The result is

$$\mathrm{ANR} = \frac{1}{|V|} \sum_{v \in V} R_v$$

where $|V|$ is the number of vocabulary items, and

R_v is the number of reversals between zones of required absence and required presence of the feature or vice versa, in the Transition Letter Sets for vocabulary item V.

f. **Mean Log Probability of Acceptance (MLP). A feature which is almost always absent but does reliably occur at some point within a vocabulary item** (or vice-versa) is an effective rejection device for eliminating false recognitions. A measure sensitive to this situation can be obtained by finding p, the frequency with which a feature is present over all vocabulary items, and computing the probability with which a random, uncorrelated sequence of zeros and ones, wherein the ones occur with frequency p, would be accepted by the Transition Letter Sets for that word. MLP is the negative natural logarithm of that probability, and can be computed from

$$\mathrm{MLP} = \frac{-1}{|V|} \sum_{v \in V} [\#_{o,v} \log(1-p_v) + \#_{1,v} \log p_v]$$

where $|V|$ is the number of vocabulary items

p is the average frequency with which the feature occurs in all vocabulary items

Evaluation and Analysis of Feature Quality Measures. Figure 3 shows estimates obtained for the six quality measures described above. Each quality measure is a non-negative number, with higher values suggesting greater utility of the feature for speech recognition purposes.

As described earlier, each of these measures is an ad hoc construction based on an intuitive concept of what characteristics a feature might indicate utility for recognition. If some of the measures evaluated are in fact reliable and accurate indicators of utility for recognition, then it would be expected that significant correlation would appear among those measures. Unfortunately, perusal of the data in Figure 3 shows poor correlation between all pairs of quality measures. This observation is borne out by the data in Figure 4, which shows the coefficient of correlation and coefficient of determination (the square of the coefficient of correlation) between all pairs of quality measures. Reliable pairs of indicators would exhibit a large positive coefficient of correlation and coefficient of determination (both near +1). Since none do, there is at most one reliable and accurate quality indicator among those used.

The lack of consistency among all pairs of suggested quality measures is quite remarkable, in view of the rational and intuitively appealing basis for each of the individual measures. It appears that no two of the measures are reliable and accurate indicators of feature utility. It remains possible, however, that a consensus (if one exists) of the measures may be indicative of feature utility. This possibility was investigated as described in the following paragraphs.

Each of the tentative quality measures establishes an order of preference (mathematically speaking, a partial order) on the set of features. This preference structure is shown in Figure 5. In that figure, a feature in a main

Feature		Quality Measure					
Buffer Location	Mfgr Ident	VFO	F01	VVF	ANZ	ANR	MLP
Initial Feature Set							
A14	MAX_D2	.132	.323	.096	1.5	.3	2.15
A13	MAX_D3	.143	.337	.238	1.5	.3	2.61
A12	MAX_D4	.107	.612	.232	1.8	.2	1.82
A11	MAX_D5	.169	.704	.252	1.4	.1	1.07
A10	MAX_D6	.101	.728	.192	1.7	.5	2.01
A9	MAX_D7	.105	.610	.185	1.6	.5	2.75
A8	MAX_D8	.083	.488	.212	1.8	.7	3.40
A7	MAX_D9	.075	.525	.193	2.1	.9	3.01
B14	SP	.050	.644	.156	2.1	0	0.59
B10*	$\emptyset_x$	.136	.741	.143	1.5	.4	2.34
B9*	UNVLC	.140	.795	.098	1.3	.2	2.14
B5*	S	.145	.795	.146	1.3	.2	1.66
B4	3	.081	.731	.145	1.9	.2	1.51
B3	I	.053	.719	.253	1.5	0	0.59
B1	Λ	.166	.687	.143	2.2	.8	3.38
Complementary Feature Set							
A15	MAX_D 1	.150	.466	.213	2.1	.9	3.18
A6	MAX_D 10	.158	.578	.165	1.7	.2	2.37
A5	PS_2	.139	.673	.315	1.8	.6	3.82
A4	PS_3	.086	.757	.202	1.5	.1	1.31
A3*	3	.118	.542	.243	1.7	.5	2.09
A2	PS_7	.276	.308	.191	1.3	.2	2.01
A1	PS_8	.277	.481	.146	1.6	.5	3.23
A0	S	.135	.634	.258	1.1	0	1.67
B15	$\int$	.101	.627	.239	2.2	.9	4.08
B13*	ε_2	.135	.752	.209	1.8	.3	1.77
B12*	$EG_1 + EG_2$	.127	.368	.118	1.7	.6	2.60
B11*	$n_1 + n_3$	.101	.577	.125	1.5	.2	1.91
B8*	$w_1 + w_2$	.137	.653	.141	1.8	0	0.78
B7	r	.094	.779	.190	1.7	0	0.65
B6	ε_1	.076	.650	.214	1.6	0	0.57
B2	i	.133	.509	.197	1.6	.2	1.65

*Feature same for VIP-100 and TI-500 (LP_4 is not shown).

Figure 3. Estimates of Individual Feature Quality Measures

Indicator Pair	Coefficient of Correlation	Coefficient of Determination
VFO-F01	-.38	.14
VFO-VVF	-,13	.02
VFO-ANZ	-.29	.09
VFO-ANR	.06	.004
VFO-MLP	.31	.10
F01-VVF	.06	.004
F01-ANZ	-.01	≈0
F01-ANR	-.28	.08
F01-MLP	-.36	.13
VVF-ANZ	.04	.002
VVF-ANR	.17	.03
VVF-MLP	.09	.008
ANZ-ANR	.46	.22
ANZ-MLP	.37	.13
ANR-MLP	.68	.46

Figure 4. Coefficients of Correlation and Determination Between Pairs of Feature Quality Measures

VFO	F01	VVF	ANZ	ANR	MLP
A1	B5*	A5	B1*	A7*	B15
A2	B9*	A0	B15	A15	A5
A11*	B7	B3*	A7*	B15	A8*
B1*	A4	A11*	A15	B1*	B1*
A6	B13	A3	B14*	A8*	A1
A15	B10*	B15	B4*	A5	A15
B5*	B4*	A13*	A5	B12	A7*
A13*	A10*	A12*	A8*	A1	A9*
B9*	B3*	B6	A12*	A3	A13*
A5	A11*	A15	B8	A9*	B12
B8	B1*	A8*	B13	A10*	A6
B10*	A5	B13	A3	B10*	B10*
A0	B8	A4	A6	A13*	A14*
B13	B6	B2	A10*	A14*	B9*
B2	B14*	A7*	B7	B13	A3
A14*	A0	A10*	B12	A2	A2
B12	B15	A2	A1	A6	A10*
A3	A12*	B7	A9*	A12*	B11
A12*	A9*	A9*	B2	B2	A12*
A9*	A6	A6	B6	B4*	B13
A10*	B11	B14*	A4	B5*	A0
B11	A3	A1	A13*	B9*	B5*
B15	A7*	B5*	A14*	B11	B2
B7	B2	B4*	B3*	A4	B4*
A4	A8*	B1*	B10*	A11*	A4
A8*	A1	B10*	B11	A0	A11*
B4*	A15	B8	A11*	B3*	B8
B6	B12	B11	A2	B6	B7
A7*	A13*	B12	B5*	B7	B3*
B3*	A14*	B9*	B9*	B8	B14*
B14*	A2	A14*	A0	B14*	B6

Figure 5. Preference Structure Induced on the Set of Features by the Six Quality Measures

vertical column is preferred over any other feature below it in the main column. Features offset to the right are all preferred equally to the feature immediately above in the main column. Features in the Initial Feature set are marked with an asterisk.

A useful concept for dealing with incompatible orders or preference structures is Pareto optimality. In this application, a set of fifteen features is Pereto optimal if there is no other set of fifteen features which is preferable under each of the six preference structures. (A set S is preferable to a set S' if and only if each member of S is not less preferable than any member of S', and some member of S is definitely preferable to some member of S'.) Starting with any set of features, one may derive from it a Pereto optimal set by examining its elements one-by-one to determine if any feature not in the set is uniformly at least as preferred under each quality measure, and definitely preferred under at least one quality measure. If so, that element is replaced by the preferred one, and the process is repeated until no further change takes place.

When this process is applied to the Initial Feature Set, it is found to be very nearly Pareto optimal. For three members of this set of features there is one uniformly preferred feature in the Complementary Set: B15 is the only feature uniformly preferable to A7 and similarly preferable to A8; A5 is the only feature preferable to A9; three features, A1, A5 and A15 are all uniformly preferable to A14. The Initial Feature Set can therefore be made Pareto optimal by replacing A7 or A8 with B15, A9 with A5 and A14 with A1, A5 or A15.

An interesting consistency among these exchanges appears when the acoustical meaning of the features is considered. Each of the features to be replaced (A7 or A8, A9 and A14) is an indicator of high energy at some portion of the spectrum, and the replacing features are mostly (B15, A5, and A1, but not A15) more complex indicators either of specific phonemes or more general spectral characteristics, such as a positive energy slope over a range of frequencies. It is tempting to infer that Pareto optimization, which in some sense represents a consensus of the quality measures, reveals a preference for the more complex features over the more basic spectral energy concentration indicators. Some confidence in this interpretation, and the indications of the Pareto optimization results in general, might be justified if the Complementary Feature Set were found to be far from Pareto optimal. Unfortunately, this is not the case. Carrying out the optimization process for the Complementary Feature Set requires only that B12 be replaced by B1 and B11 be replaced by A9, B1 or B10. The Complementary Feature Set is thus even more nearly Pareto optimal than is the Initial Feature Set. This fact indicates that the six putative quality measures are very incompatible and that any subset of fifteen features is probably almost Pareto optimal.

The dismal failure of the quality measures to give clear indications of differences among feature and, in fact, to demonstrate anything at all, is an indictment of any intuitive approach to evaluating feature utility. Apparently a satisfactory evaluation of feature utility will have to await a more penetrating analysis.

In the absence of any satisfactory indication of relative individual feature utility for recognition, the Initial Feature Set was retained for use in the remainder of the VIAS project. One virtue of this selection is that the speech data gathered using this set of features, and results obtained with them, extend the data set and results learned in other related projects (such as the Laboratory Version AIC Training System), which use the Initial Feature Set.

FEATURE SET VALIDATION (Task 1d). Selection of the Initial Feature Set for use in the subsequent analyses in this project was validated by monitoring the performance of the entire LISTEN speech processing system operating with that selected set of features. The process of extracting speech characteristics for two speakers (LHN and JEP) was monitored especially carefully to detect any indication of individual feature peculiarity. Transition Letter Sets were extracted as usual from ninety-six examples of the eleven word LCSR vocabulary, using the GZEC program. Each feature clearly contributes to the recognizability of at least some vocabulary items, and most features display regularity in most vocabulary items for both speakers. The Transition Letter Sets obtained by GZEC are shown in Figures 1 and 2. Loop Letter Sets generated for the same speech data also failed to reveal any anomalous characteristic of any individual feature. The Loop Letter Sets indicated that the Transition Letter Sets almost completely characterize the TTI-500 output for each vocabulary item, as they did for the VIP-100 output. That is, Loop Letter Set states are quite infrequently entered, most words being recognized through a sequence of transitions from one Transition Letter Set state to the next.

The remainder of the voice data analysis process leading to the data base needed for real-time recognition is not easily related to individual feature characteristics. These processes include collecting data about the timing of transition and loop sounds (states), violation and artifact (false alarm) rates, etc. However, these were monitored and no peculiarities attributable to, or suggestive of, individual feature anomalies were detected.

Although no specific anomalies were noted in the process of extracting voice reference data from speech samples for these two speakers, the recognition accuracy which LISTEN exhibited for them was significantly inferior to that obtained for MWG using the VIP-100. As described in connection with Example Set generation, the poor performance for JEP can be attributed to the method of generating individual vocabulary items, but MWG's and LHN's voice data were processed in functionally identical ways. It remains ambiguous, therefore, whethe. the difference in recognition performance between these two speakers is due to speaker peculiarities or speech preprocessor differences, and if the latter, whether a different selection of features might lead to better recognition accuracy. Unfortunately, project resources did not permit resolution of this ambiguity.

SUMMARY OF TECHNOLOGY TRANSFER TASK RESULTS. Practical considerations forced an indirect approach to choosing a set of features for use with LISTEN from those available from the newer model speech preprocessor. An Initial Features

Set was constructed following the preprocessor manufacturer's recommendations. Six measures of individual feature utility were posited, evaluated and the results analyzed. The six measures were found to be pairwise incompatible to a high degree. The Initial Feature Set was adopted for use in the remainder of the study in the absence of any rationale for selecting another set, because this extends the accumulated data and experience based on that set of features.

In a qualitative sense, the previously developed LISTEN technology was successfully transferred to the new preprocessor in all phases of the LISTEN operation, including voice data collection, voice data analysis and reference data generation, and real-time voice recognition, in the sense that no qualitative change to LISTEN was required to achieve recognition. However, inferior recognition performance for LHN, whose voice data were processed in essentially the same way as MWG's, leaves it unclear as to whether LISTEN can obtain similar performance with the two preprocessors. On a word basis (counting all insertions, deletions and substitutions as errors) 95% recognition was obtained for MWG and 89% for LHN, using Test data, without speaker feedback. This marginal difference in performance could presumably be due to either speaker or preprocessor differences.

CONTRIBUTION OF EACH INFORMATION SOURCE TO RECOGNITION

As described in Reference 1, the LISTEN speech recognition system has two major subdivisions, implemented in programs MEX and MINT. MEX detects the presence of segments of speech preprocessor output which exhibit the structural characteristics of individual vocabulary items, and notified MINT of these potential word recognitions. MEX also, in this process, notes the presence of certain non-fatal structural peculiarities (if any) of each potential recognition. MINT then processes these data to distinguish between real recognitions and artifactual ones. In doing so, MINT uses information of various other kinds. Each of these information sources is discussed separately in the following paragraphs.

CONTRIBUTION OF STRUCTURAL INFORMATION IN MEX. Structural data are used in two ways in the LISTEN recognition procedure, as the description of MEX and MINT just given shows. The expected structure of individual vocabulary items is first used in MEX to detect the potential presence of that item in the incoming stream. The first use of structural information is thus an initial detection function; indicators of its contribution are the frequency with which vocalizations of words are not detected, and the frequency with which artifacts are generated. These data are presented in Figure 6 for Test data.

Speaker/Preprocessor	Number of Vocabulary Items Spoken	Number of Vocabulary Items not Detected by MEX/Percentage	Number of Artifactual Recognitions
MWG/VIP-100	1049	7/0.7%	2512
LHN/TTI-500	1054	21/2.0%	1269

Figure 6. Missed and Artifactual Recognitions in MEX Output

The difference in MEX failure and artifact production rates, between the two speaker/preprocessor combinations, is quite remarkable. The artifact production rate for LHN is half that for MWG, at the cost of three times the MEX rejection rate. The detection of the potential presence of a word in the speech signal is primarily dependent upon the combined discrimination capabilities of the preprocessor features and the Transition Letter Sets. Visual and quantitative comparison of the Transition Letter Sets for these two speakers fails to reveal any substantive difference. (For example, both speakers average 9.5 Transition Letter Sets per vocabulary item.) If there were significant differences in the ariculatory habits of the two speakers, for example in enunciation precision, presumably the difference would be evident as structural differences in their respective Transition Letter Sets. As there are no apparent differences, it seems likely that the contrasting MEX failure and artifact production rates are characteristic of the preprocessors, or at least of the sets of features LISTEN accepts from the two preprocessors, and not due to differences between the two speakers.

CONTRIBUTION OF OTHER INFORMATION SOURCES IN MINT. In the process of detecting, by use of structural information, the potential occurrence of a vocabulary item in the speech data stream, MEX also computes two measures of how typical the time duration of various detected feature combinations are. MINT thus receives from MEX notification of the occurrence of a potential recognition of a particular type, start and end times of the potential recognition, an indication of any detected structural peculiarities, and two indicators of temporal peculiarity. Using the start and end times of the potential recognition, MINT (in principle, at least) builds and operates upon a directed graph representing the utterance. This directed graph consists of a Start and an End node, together with one additional node for each potential recognition. A pair of nodes is joined by a directed edge if and only if the start and end times of the events are compatible with one node representing the event immediately preceding the event represented by the other node. MINT then computes the path through this directed graph, moving backwards from End to Start, seeking the best explanation of what has been observed about the utterance. In the process of doing this computation, MINT adds to the structural violation and intraword timing data supplied by MEX, data about the a priori probability that a potential recognition is real versus artifact, about its coincidence in time with other potential recognitions, and about the interword timing. All of these data are expressed numerically as a scaling constant (-64) times the natural logarithm of the likelihood ratio for the occurrence of what was actually observed. That is, the i^{th} information source is summarized as a value

$$\Delta Q_i \cong -64 \, \ell n \frac{\text{Prob (observation/real)}}{\text{Prob (observation/artifact)}}$$

Those information sources relating to individual potential recognitions produce ΔQ values associated with nodes, and the interword timing data produce ΔQ values associated with edges of the directed graph.

The ΔQ values, attached by MINT to nodes and edges of the graph of potential recognitions, are estimates of the scaled log likelihood ratios based on statistical models of each information source. The parameters of these statistical models are estimated from speech data during the voice data generation process. Validity of these statistical models and estimation procedures is examined in Tasks 5a and 5b, described later in this section. In this task, attention is directed to determining how effectively each information source, as represented by its associated ΔQ values, contributes to the recognition procedure.

As shown in Reference 1, under suitable assumptions, the Bayes optimal solution to the problem of deciding which path through the graph is best reduces to the problem of finding the path with minimum sum of ΔQ values on nodes and edges. Evaluating an information source's contribution to recognition thus reduces to determining how effectively the ΔQ values help establish the correct path through the graph as the one with minimum total cost. Although MINT considers all possible paths through the graph of the utterance, correct identification of the spoken words depends decisively on the ΔQ values attached to two particular paths through the graph (when they both exist): the lowest cost path of those which gives the correct answer, and the lowest cost path of those which give any incorrect answer. If these two paths exist, the correct answer is found precisely when the total cost of the former is less than the total cost of the latter. The effectiveness of the ith information source, in establishing a correct path as the chosen one, is thus indicated by the difference between the sums of the ΔQ values along the best of the incorrect paths and the best of the correct paths. Subtracting the latter from the former gives a value which, when positive, indicates that the information source in question is a productive contributor to selecting the correct path but which, when negative, indicates that the information source is counterproductive.

The first measure used to evaluate the contribution of the ith information source is, for the reasons just given, defined to be

$$M_i = \underset{\substack{\text{best}\\ \text{incorrect}\\ \text{path}}}{(\Sigma\Delta Q_i)} - \underset{\substack{\text{best}\\ \text{correct}\\ \text{path}}}{(\Sigma\Delta Q_i)}$$

The measure of information source contribution just defined cannot be applied if the graph of the utterance does not contain at least one path giving a correct result and at least one path giving an incorrect result. Although there are several different possible reasons for such a situation arising, only one has been observed to arise commonly in practice. That is the failure of MEX to detect a word actually spoken and inform MINT of its existence as a potential recognition. Failures of this type occur only when the word as spoken does not have expected structural characteristics; i.e., when the word exhibits extensive structural violation. These cases are much in the minority.

The measure M gives a value to the contribution of each information source towards correct recognition in each utterance. To summarize the utility of the information source ver many utterances requires some approach to dealing with the collection of M values for each utterance. One approach, adopted here, is to present a graph of the cumulative distribution of observed M values.

The PASS program STATSUM computes the best correct and best incorrect path through the graph of each utterance, and also the contribution of each information source to the cost difference between these two paths, i.e., the M value defined above.

Figures 7 and 8 show the M distributions for each information source, for MWG and LHN, respectively. From these graphs one can obtain at a glance such indicators as the fraction of cases wherein the information source was counterproductive (i.e., the fraction of cases where M is negative) and such qualitative features as evidence of peculiar clusters of cases.

Since M can be interpreted as an estimate computed in MINT the logarithm of the likelihood ratio for the correct path being in fact correct, M values can be translated into odds that the correct path is in fact correct. For example, an M value of 147.4 corresponds to an estimate in MINT that, according to that particular information source, the odds are 10-to-1 that the correct path, rather than the best incorrect one, is in fact correct. Odds values are indicated in Figures 7 and 8.

Another interesting indication of the relative value of an information source is the frequency with which it is the most "productive" of all the sources considered, in the sense of differentiating most strongly (and correçtly) between the best correct and best incorrect explanations of an utterance as indicated by a most positive M value. The complementary notion is the infrequency with which the information source is not the most counterproductive one (i.e., does not have the most negative M value). An information source which is essentially random and which takes on large values would frequently be the most productive and also frequently be the most counterproductive, as these terms are defined above. Therefore, both these indications of information source quality should be considered simultaneously.

Using the data produced by STATSUM, it is possible to compute the fraction of cases in which each particular information source is the most productive and the fraction of cases in which it is not the most counterproductive. Figure 9 shows these figures for MWG's and LHN's test data, in the form of two-dimensional plots. The same data are given in tabular form in Figure 10. As can clearly be seen in Figure 9, there is a definite consistency in the productivity of each information source for both speakers, with the single exception of the association information source. If one uses as a measure of the utility of an information source the sum of the frequencies with which it is most productive and not least productive, the following ranking (best to worst) of information sources holds for both speakers:

a. Interword Timing

b. Violation Category

c. Intraword Timing (QT)

d. A priori and Intraword Timing (QL)

with Association somewhere below Violation Category.

Another interesting single valued measure of the contribution of each information source can be obtained by computing the information contained in the distribution of M regarding the selection of the correct path. To apply the theory of information to this situation, one can model it as follows. Let the two paths contending for choice (the best of the correct and the best of the incorrect) be labeled A and B in the order they are discovered in MINT. As this is entirely random labeling, the correct path has equal probability of being path A or path B. MINT computes the total ΔQ along the two paths and chooses the path with minimum value. The difference in path ΔQ values (say, path A minus path B) will then be distributed as a random variable equal to S times M, where S is a random variable with probability one half of being either +1 or -1, and M is the difference in ΔQ values for the incorrect path minus the correct path. The product SM is the value available in MINT, which may be regarded as a signal received. The message sent is equivalent to designation of which path (A or B) is the correct one, or equivalently, whether the value

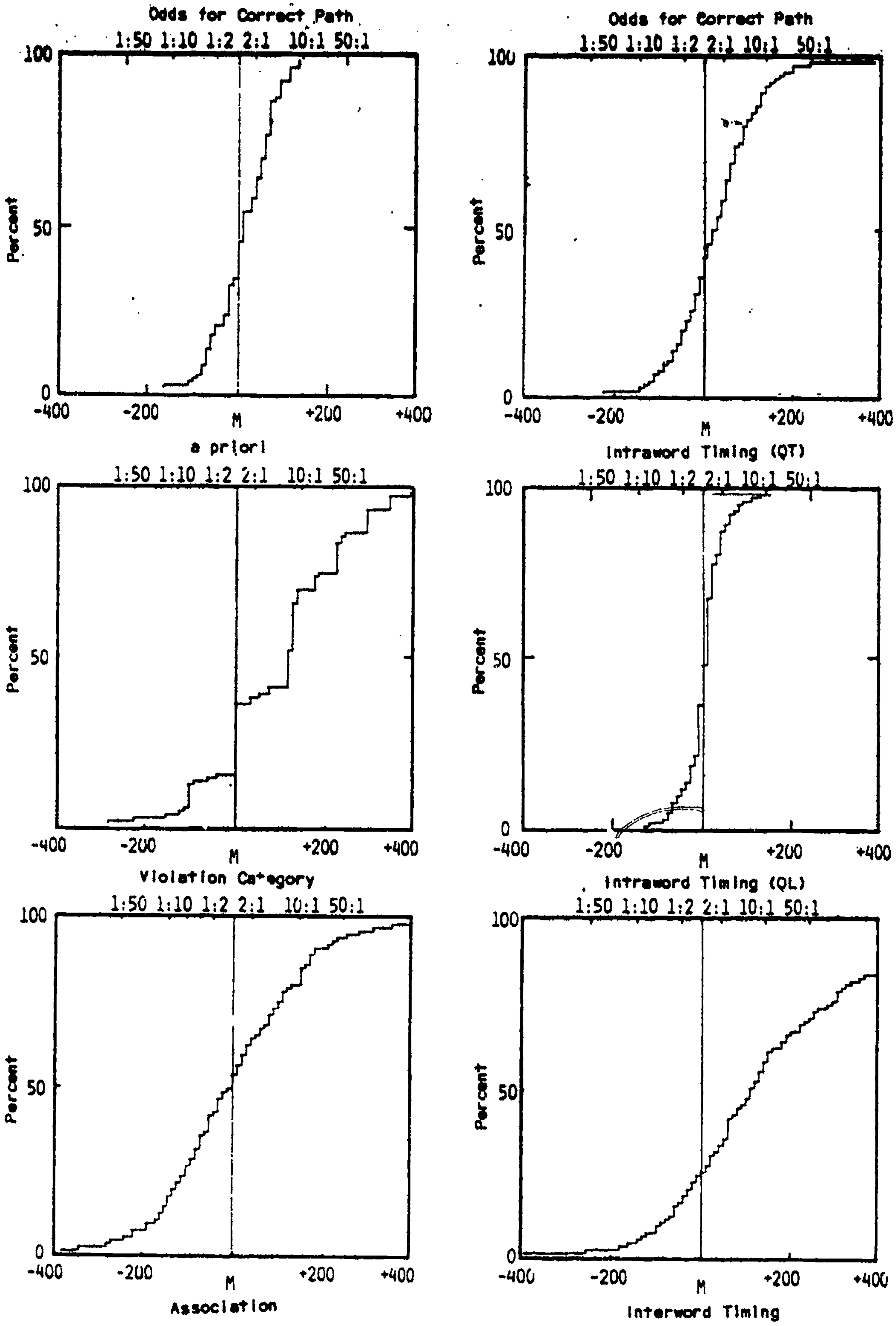
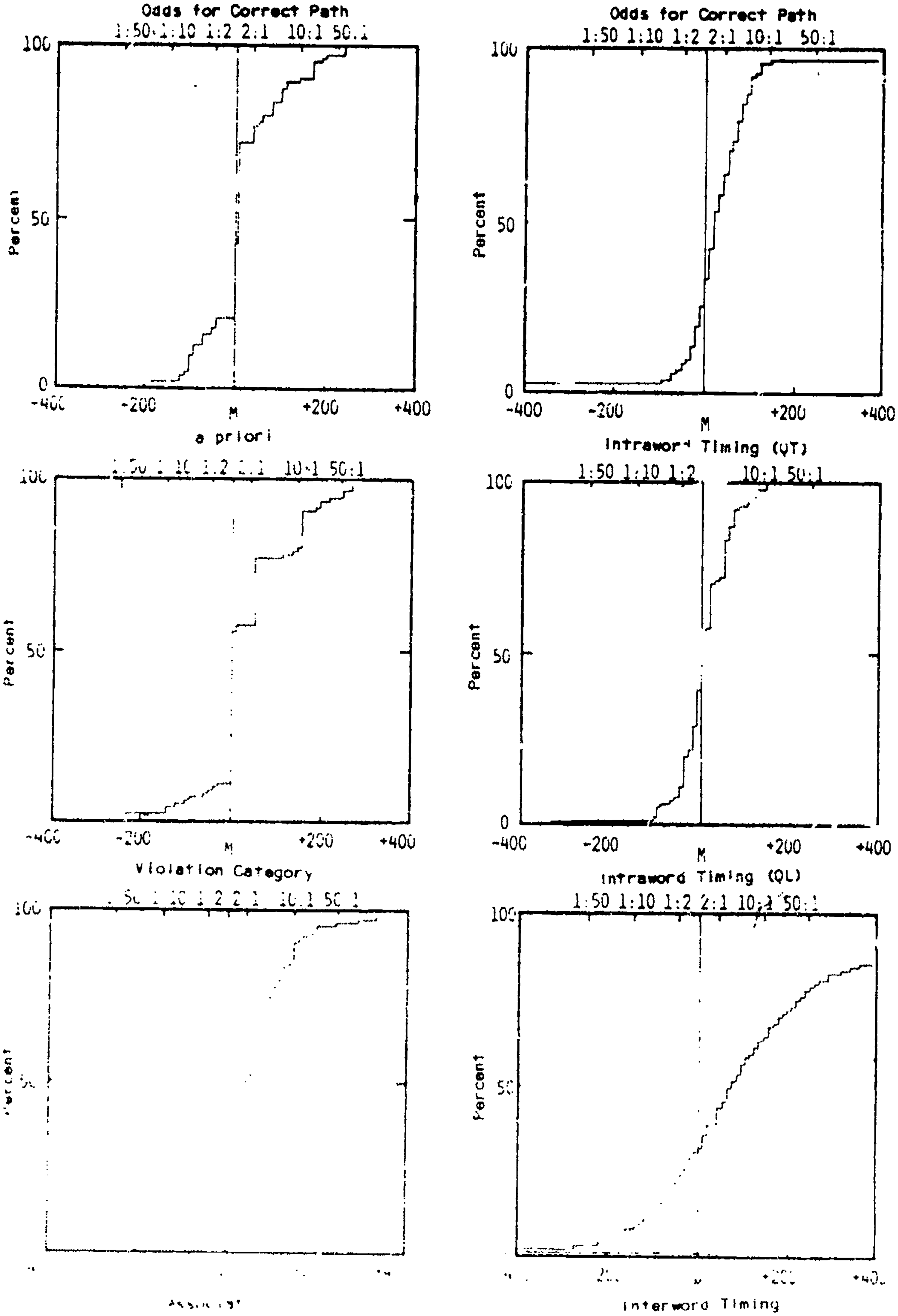


Figure 7. Cumulative Distribution of M for Various Information Sources Used in MINT. Speaker MWG



[illegible] Information [illegible]
[illegible] M[illegible] [illegible]

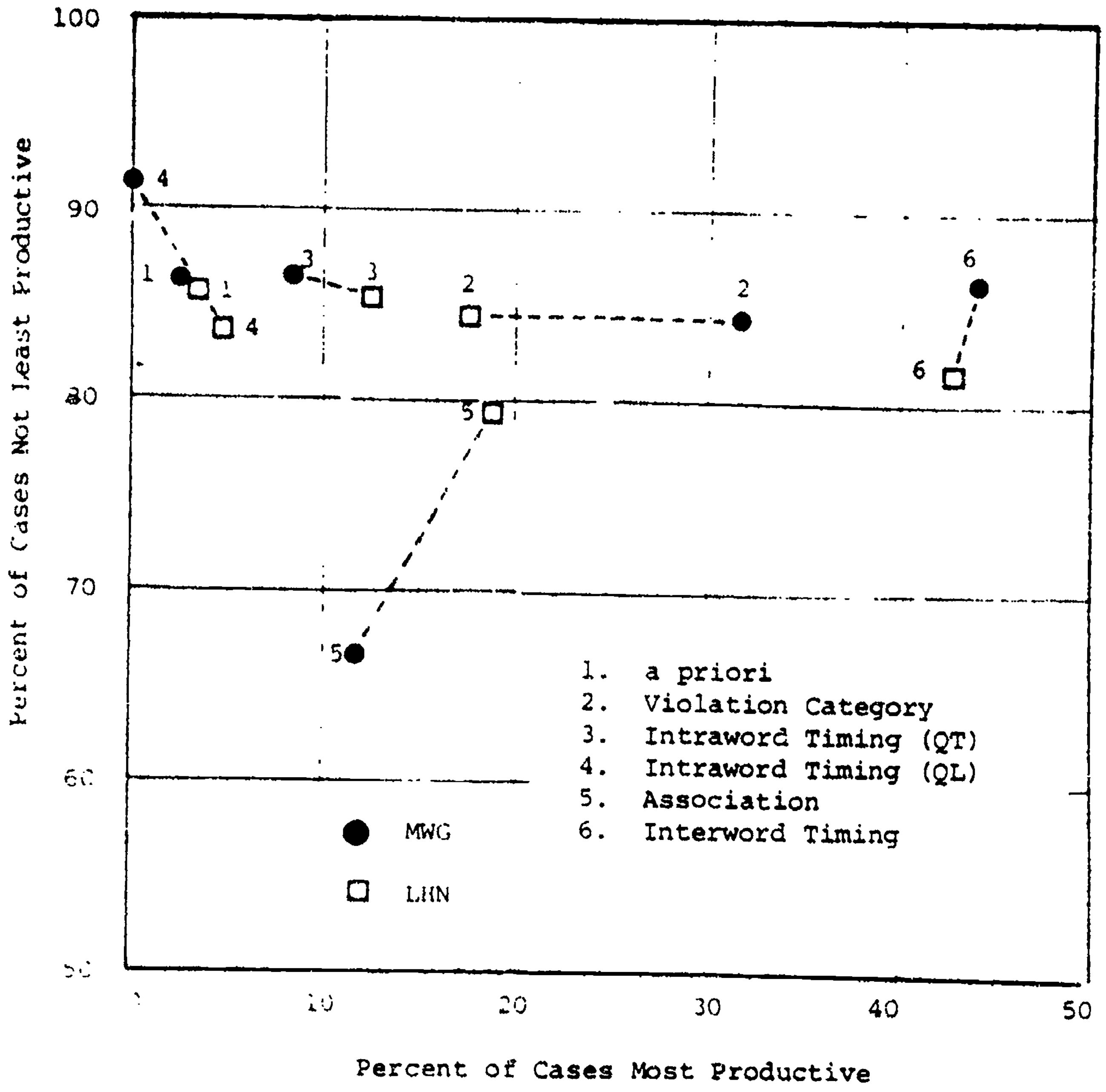


Figure 1. Frequency with which Various Information Sources Are Most Productive and Not Least Productive

Information Source	Speaker and Preprocessor	% of Time Correct (Equivocal)	% Time Best	% Time Not Worst	Information Content (bits)
A priori	MWG/VIP-100	66	3.5	85	.15
	LHN/TTI-500	57	3.3	86	.15
Violation Category	MWG/VIP-100	85 (20)	32	84	.29
	LHN/TTI-500	90 (45)	18	84	.24
Association	MWG/VIP-100	47	12	67	.06
	LHN/TTI-500	62	19	79	.11
Intraword Timing (QT)	MWG/VIP-100	59	9.1	87	.07
	LHN/TTI-500	67	12.5	85	.13
Intraword Timing (QL)	MWG/VIP-100	51	0	91	.05
	LHN/TTI-500	53	4.8	84	.15
Interword Timing	MWG/VIP-100	75	44	86	.32
	LHN/TTI-500	69	43	82	.21

Figure 10. Summary of Indicators of the Contribution of Each Information Source to Recognition

of S is +1 or -1. The information content of the signal about the message sent according to Information Theory, is the entropy of the random variable SM minus the entropy of the conditional random variable SM given S. This is a value lying between zero and one bit of information (one bit being exactly enough information to decide perfectly between the two alternatives). If the M value were always positive, for instance, one could always recognize the correct path as the one with least ΔQ value. The information content in that case is found to be one. If M is distributed symmetrically about zero, its information content is zero.

The information content of an information source, defined above, has several interesting properties. One of them is that it establishes an upper bound on how successfully an information source can be used to select the right path, i.e., to recognize what was spoken, regardless of the algorithm used to effect the recognition.

The information content of each information source has been estimated from the observed distribution of M values. The results are given in Figure 10. These data tend to corroborate the ranking given to each of the information sources earlier.

The fraction of time that an information source gives a correct, i.e., productive, indication of the right path can be read from the cumulative distribution of M values. A positive M value indicates a correct indication, and a negative M value an incorrect one. These data are also summarized in Figure 10 for each information source, and lend further evidence that the information source ranking given earlier is correct. (Since there are only eight violation categories, and violations are relatively rare, it often happens that the two paths do not have potential recognitions which differ in violation category. The M value in that case is zero, and the information source is equivocal. The frequency with which this occurs is also given in Figure 10.)

ANALYSIS OF RECOGNITION ERRORS

Two aspects of recognition error analysis covered here are the automatic clasification of errors by programs in the PASS and relating recognition errors to information sources. The related analyses are discussed below.

AUTOMATIC CLASSIFICATION OF RECOGNITION ERRORS. In connected speech, many possible explanations for the observed speech data are usually generated in an attempt to recognize what was actually spoken. When a wrong explanation is selected, the cause may be related to a large number of factors. This is especially true in an algorithm like MINT which considers the entire complex of potential recognitions and all plausible explanations for the entire utterance. Classification of errors might at first seem like a hopeless task, as the process can apparently go wrong in very many ways. However, when the recognition system works even moderately well, most errors are found to belong to a small collection of types. Simple deletions, insertions and one-for-one substitutions, for example, comprise the majority of all erors. So classification, and its automation, is not a hopeless goal.

A useful dichotomy of recognition failures distinguishes between those cases where there does not exist a path through the graph of the utterance which yields the correct string of vocabulary items, and those cases where there does exist such a path. The former will be called "structural failures."

Structural failures are generally of two types. One, treated earlier, results from MEX's failure to detect the potential presence of a word actually spoken. The other type occurs when the correct potential recognitions are present in the graph of the utterance, but MINT fails to consider a path through them. This can occur only when the interword timing is so anomalous as to exceed limits (set in MINT) on the time between potential recognitions to be considered potential predecessors.

The PASS program BIGMINT recognizes structural failures and provides data whereby the type of failure involved may easily be determined.

Misrecognitions involving a source of error other than structural failure occur because some incorrect path through the directed graph of the potential recognition has lower total cost than any correct path. By considering only the best of the correct and the best of the incorrect paths, the locus of the difficulty becomes apparent because even in utterances of several words, the best of the correct and the best of the incorrect paths usually have much in common, the difference existing only at a small portion of the utterance.

As an example of the simplification obtained by considering only the best of the correct and best of the incorrect paths, consider the following. The phrase "015." occurs in Test data for MWG. This utterance was misrecognized, as there were five paths through the graph of the utterance with lower cost than that of the correct path. These five paths corresponded to 201557, 20155, 015.7, 01557 and 0155, the last one having least cost. Examining the correct path (015.) and the best of the incorrect paths (0155) on a node-by-node basis shows that they entail the same first three nodes (not obvious from the vocabulary items), differing only in the last node. The node-by-node analysis shows that this is a case of simple substitution, and the four other incorrect paths with costs less than the correct path are not informative, being present only because of the anomalous properties of the final "." and the end of the utterance - anomalies already indicated by the comparison of best correct and best incorrect path. This simplification is typical to the point of being universal.

Comparison of the best correct and best incorrect path becomes impossible if there is no correct path or no incorrect path. But when at least one correct and one incorrect path through the utterance exist, the utterance can be further categorized as entailing either a single or multiple branches at which the two paths differ. The concept is illustrated in Figure 11.

When the difference between the best paths is the insertion or deletion of contiguous words, the path difference is interpreted as a single branch case, as in Figure 11(b).

The distinction between single branch and multiple branch categories is useful because the single branch group is amenable to further subdivision and because the multiple branch case is so rare. (It has not been observed to occur.)

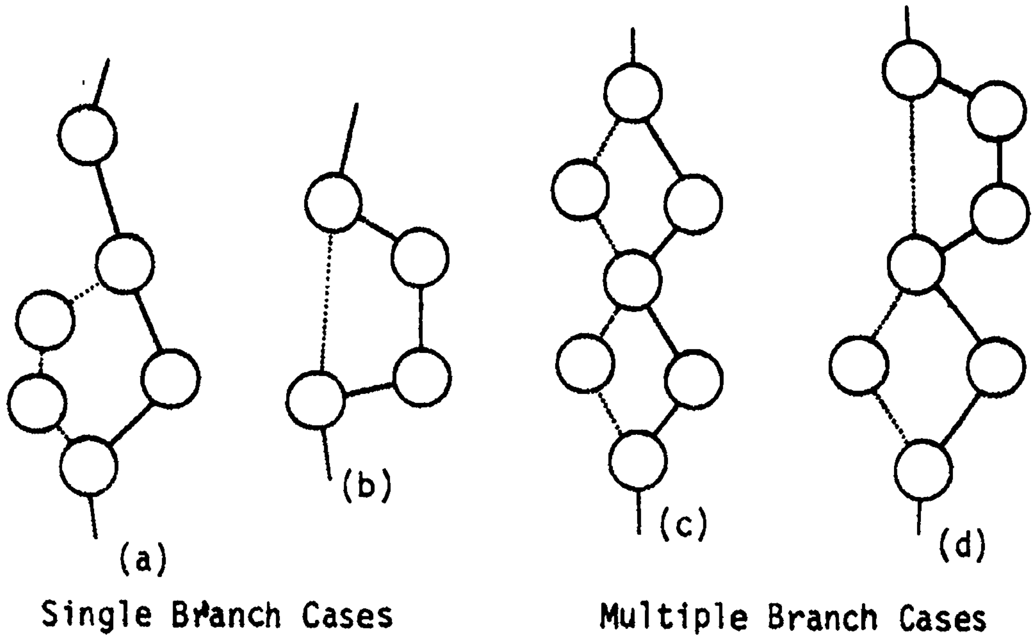


Figure 11. Illustrating Single and Multiple Branch Differences Between the Best Correct (Solid) and Best Incorrect (Dotted) Paths Through the Graph of an Utterance

Classifying utterances on the basis of the best correct and best incorrect paths thus gives rise to four categories of cases:

Category 0	No best correct path exists (structural failure)
Category 1	A single branch distinguishes the best correct and best incorrect paths
Category 2	Multiple branches distinguish best correct and best inc rrect paths
Category 3	No best incorrect path exists

Among the Category 1 cases one may further distinguish cases on the basis of the number of nodes in each part of the differentiating branch. Writing the number of nodes (words) in the correct branch on the right and the nodes in the incorrect branch on the left, a type (0,1) utterance is one in which the best incorrect path is formed by deleting one word in the correct utterance. If the best incorrect path has, in fact, lower associated cost than the best correct path, a simple deletion occurs. Similarly, a type (1,1) utterance is one in which the error, or potential error, is a simple substitution. A type (1,0) utterance is potentially a simple insertion, and a type (2,3) utterance would potentially be a more complex type of substitution.

The PASS Program STATSUM classifies all utterances (both those correctly and those incorrectly recognized) according to Category and Type as defined above, and the program SSPLOT prints various data about each classified group. These data can be used to classify any set of utterances, including the subset of utterances with errors, as SSPLOT indicates which of the classified utterances were misrecognized. Classification of all utterances in this uniform way is useful in that it shows both how typical various types of contention arise in LISTEN, and the relative success MINT has in resolving each type of contention.

The results of classifying test data for MWG and LHN in this way are shown in Figure 12.

The simpler forms of contention are found to be most common in LISTEN. MEX's failure to spot the potential presence of a word, and simple insertion, deletion or substitution of a simple word cover the vast majority of cases examined.

Correct recognition most frequently entails the resolution of which of two alternative words to choose; i.e., resolution of a simple one-for-one substitution problem. Furthermore, this is a most difficult problem to resolve, as indicated by the relatively small fraction of these cases resolved correctly. One probable reason for the difficulty in resolving substitution of like numbers of words (type (1,1) and (2,2) controversies) is that the strongest information source, interword timing, is relatively ineffectual in these cases.

MISRECOGNITIONS VIS-A-VIS INFORMATION SOURCES. The data produced by STATSUM, comparing the best incorrect and best correct paths, permits detailed examination of the course of each misrecognition. Quite often one peculiarity of a troublesome word stands out in a misrecognized utterance, but the specific nature of the peculiarity varies from utterance to utterance. While it is easy to identify the most counterproductive information source in individual utterances, it is not reasonable to summarize these results for misrecognition cases only. An information source may correlate highly with both correct and incorrect recognitions, if it has a random component large enough to dominate all other information sources. To maintain a balanced view of an information source, then, it is important to consider its influence on correct recognitions as well as on misrecognitions. This was done in the analysis of the contribution of each information source, summarized in the earlier Figures 6 through 10.

Another indication of the association of errors with information sources can be obtained by counting the number of correctly and incorrectly recognized utterances, categorized by which was the most productive and which the least productive information source. The PASS program STATSUM provides data from which these counts can easily be accumulated. Results obtained in that way are presented in Figure 13.

	MWG Test Data			LHN Test Data		
	All Utterances	Errors Only	% Resolved Correctly	All Utterances	Errors Only	% Resolved Correctly
Category 0 (Structural Failure)						
MEX failed to detect	7	7	0	21	21	0
Anomalous Interword Timing	0	0	-	16	16	0
Category 1						
Type (0,1)	45	1	98	64	3	95
(1,0)	112	9	92	7	3	57
(1,1)	155	28	82	208	49	76
(1,2)	1	0	100	5	3	40
(2,0)	1	0	100	0	0	--
(2,1)	3	0	100	2	1	50
(2,2)	5	3	40	4	2	50
Total	322	41	87	290	61	79
Category 2	0	0	-	0	0	--
Category 3	0	0	-	0	0	--

Figure 12. Classification of all Utterances and of Erroneously Recognized Utterances, by Category and Type

Worst Information Source

		AP	VC	QT	QL	AS	TG	Totals
Best Information Source	AP		1c 2i		2c	1c 3i	1c 1i	5c 6i
	VC	13c 2i		12c 1i	12c	35c 4i	20c 3i	92c 10i
	QT	3c	5c		2c	6c 6i	6c 2i	22c 8i
	QL						1c	1c
	AS	4c	11c	7c	3c		8c 5i	33c 5i
	TG	27c	28c 2i	22c	8c	43c 10i		128c 12i
	Totals	47c 2i	45c 4i	41c 1i	27c	85c 23i	36c 11i	

Speaker MWG

Worst Information Source

		AP	VC	QT	QL	AS	TG	Totals
Best Information Source	AP		1i		1c	2i	3c 2i	4c 5i
	VC	10c 3i		4c	4c 3i	11c 4i	8c 4i	37c 14i
	QT	2i	2c 1i		4c	7c 10i	2c 8i	15c 21i
	QL		2c	2c		4c 2i	1c 3i	9c 5i
	AS	2c	12c 1i	5c	11c 2i		18c 4i	48c 7i
	TG	23c 1i	24c 2i	29c 3i	22c	16c 3i		114c 10i
	Totals	35c 6i	40c 5i	40c 3i	42c 5i	38c 22i	32c 21i	

Speaker LHN

Legend:		
	*c	Number of Correctly Recognized Utterances
	*i	Number of Incorrectly Recognized Utterances
	AP	a priori
	VC	Violation Category
	QT	Intraword Timing (QT)
	QL	Intraword Timing (QL)
	AS	Association
	TG	Interword Timing

Figure 13. Counts of Utterances Correctly and Incorrectly Recognized, Categorized by Most Productive (Best) and Least Productive (Worst) Information Source. Test data, Category 1, Types (0,1), (1,0) and (1,1).

CRITICAL EXAMINATION OF INFORMATION SOURCE MODELS

In the decision theoretic model of the problem solved by MINT, each information source is considered to provide one component of a complex observation of the characteristics of the utterance. Solution of the problem then rests on estimating the probability that the particular observed value would arise, given various hypotheses about what was actually said. In the MINT implementation of this solution, the observed values must be used as a basis for estimating the logarithm of the likelihood ratio; i.e., the logarithm of the ratio of the conditional probability that the observed value would occur, given that the potential recognition is a real one, to the conditional probability that the observed value would occur, given the potential recognition is an artifact. (ΔQ_i as defined earlier.)

The mechanism for converting an observed value to an estimate of the log likelihood ratio entails a statistical model; specifically, a pair of conditional distributions of the observable values, given they are descriptions of either real or artifactual recognitions. These statistical models contain distribution parameters which are estimated from Interim Test data, using procedures appropriate to the nature of the data and the statistical models. Recognition accuracy and theoretical soundness of the MINT algorithm both require that these statistical models and parameters must be reasonably descriptive of the actual nature of speech data.

Each information source presents its own difficulties for statistical modelling, but three issues can be identified which are of interest in assessing the validity of each model:

a. The independent variables must be properly identified.

b. If a distribution shape has been assumed, it must fairly describe the actual shape

c. The model must describe statistical characteristics which do generalize from Interim Test data to new speech data.

a priori MODEL. The decision theoretic model of the problem solved in MINT requires knowledge of the a priori probability that a particular hypothesis - in this context a string of vocabulary items which potentially may have been said - will arise. ıhis a priori probability is the probability unconditioned by any observation about the acoustic data as received by the preprocessor or operated upon by MEX, except that the graph of the utterance admits of the string; i.e., contains a path corresponding to the hypothesis.

It is assumed that the probability that a hypothetical path is in fact the correct one, without consideration of any details of the individual potential recognitions comprising the path, or of their mutual temporal relationship (beyond that required to make them constitute a path through the graph) depends only on the vocabulary items in the path. It is further assumed that the a priori probability of correctness of the entire path is the product of probabilities associated with each vocabulary item in the path. Finally, it is assumed that the probability that a particular recognition of a particular vocabulary item in a path is in fact real can be estimated from the relative frequency of occurrence of that vocabulary item as a real-vice-artifactual potential recognition in a large body of speech data.

Several links in this chain of assumptions are difficult or impossible to justify on theoretical grounds, or even to test. In fact, the assumptions are rationalizations for the way the a prior' contribution to total cost is actually computed in MINT, which was chos‹ because it is plausible and computable at small cost in data storage and processing burden. However, the evaluation of the a priori information source presented earlier shows that this procedure results in a cost contribution which is productive more often than it is counterproductive. The whole chain of assumptions is thus justified in that it leads to a useful result. Two features of the a priori statistical model which are amenable to test and verification are its dependence upon vocabulary item, and stability of the relative frequency of real and artifactual potential recognition for each vocabulary item type. To verify these aspects of the model, the relative frequency of occurrence of real and artifactual potential recognitions for each vocabulary item are compared for Interim Test data and Test data in Figure 14.

As Figure 14 shows, there is considerable variation in the relative rate of occurrence of artifactual recognition for various vocabulary items, justifying the use of vocabulary item as an independent variable. More precisely, it is the <u>form</u> of the vocabulary item which is important and which is used as the independent variable, in the sense that some vocabulary items exist in an initial form and a non-initial form, and different a priori statistics are stored for the two forms.

These data also show the stability of the artifact production rates, indicating that rates estimated from Interim Test data remain valid for Test data, thus presumably for all new speech data. Of course, artifact production is dependent upon vocabulary content and frequency of occurrence of various vocabulary items in the corpus of spoken material. Since Test and Interim test data have identical vocabularies and incidence of vocabulary items, the generalization from Interim Test data to Test data is justified. However, if LISTEN were to be used with a set of utterances wherein each item did not occur a substantially equal fraction of the time, new a priori statistics should be derived from artifact occurrence rates.

Data used in the analysis of the a priori statistical model are gathered using the PASS program STATSUM and printed using the DOGLEG option in LICVAT.

VIOLATION CATEGORY MODEL. In the process of detecting the potential presence of a vocabulary item in the speech stream, MEX notes several types of deviations of the speech from the structure expected of that vocabulary item. Eight types of structural violation are recognized, and they are described in detail in Reference 1. Each type of structural violation is assigned a violation category number, ranging from 1 through 8. Violation category 0 indicates that no structural violation was detected by MEX.

Violation category (0 through 8) is regarded in MINT as an observed characteristic of the potential recognition, and the probability of occurrence of a given violation category is modeled as depending only upon violation category and whether the recognition is real or artifact. Dependence upon vocabulary item is suppressed, primarily because very few examples of some

Vocabulary Item and Form	MWG Interim Test Data	MWG Test Data	LHN Interim Test Data	LHN Test Data
0	1.2	1.2	0.11	0.15
1	1.4	1.1	2.0	1.9
2 (Initial)	6.1	6.2	1.9	1.5
2 (Non-Initial)	2.2	2.4	0.79	0.76
3	4.5	5.2	*	*
3 (Initial)	*	*	0.10	0.03
3 (Non-Initial)	*	*	0.23	0.21
4	2.9	2.9	2.0	1.7
5	1.8	2.0	0.052	0.063
6	1.0	1.0	0.26	0.27
7	2.5	2.4	0.44	0.53
8	3.3	3.7	4.5	4.1
9	2.4	2.7	0.89	0.72
•	0.21	0.34	1.7	1.4

(*Vocabulary item does 1ot exist in this form for this speaker)

Figure 14. Artifact Production Rates. (The number of artifactual recognitions divided by the number of times the vocabulary item was spoken.)

violation categories are observed for some vocabulary items, making it impossible to estimate reliably the rate of occurrence of these violations for real recognitions.

Vocabulary Item Dependence for Artifactual Recognitions. The low rate of occurrence of violations for real recognitions makes it impractical to estimate its vocabulary item dependence with a reasonable sample size. However, artifactual recognitions exhibit violations much more frequently, and the vocabulary item dependence of their frequency can be estimated. It might, therefore, be both practical and useful to use a model of violation occurrences which treats violation as independent of vocabulary item for real recognition, and dependent upon vocabulary item for artifactual recognitions.

To examine this potential improvement of the violation category model, the variation of the frequency of occurrence of artifactual violation categories with vocabulary item was evaluated, as shown in Figure 15. To prepare that figure, the conditional probability that a given violation category would occur, given that the recognition is artifactual and of a given vocabulary item and form, was estimated using the frequency of that occurrence. The maximum and minimum values of the probabilities estimated in that way, over all vocabulary items, are shown in the figure. The average probability of occurrence of each violation category (for all artifactual recognitions), found by ignoring the vocabulary item dependence, is also shown there.

The data in Figure 15, collected using the DOGLEG option in PASS program LICVAT, show that for many violation categories there is a significant vocabulary item dependence in the frequency of occurrence. Therefore, extensions of this model to include vocabulary item as an independent variable has definite potential to increase the effectiveness of this information source.

Stability. The stability of the rate of occurrence of violation categories was evaluated by comparing the frequency of occurrence of violations in Interim Test data with their frequency in Test data. The results, also collected using the DOGLEG option in LICVAT, are shown in Figure 16. The data show that violation occurrence rates can be estimated safely using Interim Test data, for both real and artifactual recognitions.

INTRA-WORD TIMING MODEL. During the recognition process, MEX notes the time spent in each state of the recognition automaton. A measure of how typical the loop state durations are is accumulated as a linear combination of the time spent in each loop state. The resulting value is denoted QL, and is treated in MINT as an observation associated with the potential recognition. QL is a non-negative number. The coefficients of the linear form used in computing QL are obtained from Training data, and the computational procedure (forming the linear combination) is based on a model of the joint distribution of the individual loop state durations, as described in Reference 1. MINT itself uses a model of the distribution of QL values which is quite independent of the model upon which the computation of QL is based.

In MINT it is assumed that QL is distributed exponentially over positive values, with a mass concentration at zero, in a way which depends upon the vocabulary item type and form, and whether the potential recognition is real or artifactual. The parameters of the distribution (the probability that the

		Violation Category								
Speaker	Range	0	1	2	3	4	5	6	7	8
MWG	Min. Avg. Max.	.018 .17 .371	.015 .074 .29	0 .061 .19	.13 .24 .51	.004 .031 .101	.057 .13 .26	0 .002 .020	.015 .22 .59	.019 .069 .17
LHN	Min. Avg. Max.	0 .38 .67	0 .053 .12	0 .044 .087	0 .23 .43	0 .010 .043	0 .084 .28	0 .013 .26	0 .14 .79	0 .052 .29

Figure 15. The Range of the Conditional Probability of Occurance of Violation Categories for Artifactual Recognition Over Vocabulary Items. (Test data used in both cases.)

Real Recognitions

	Violation Category								
Speech Data	0	1	2	3	4	5	6	7	8
MWG Interim Test MWG Test	.88 .88	.052 .033	.016 .017	.041 .049	.002 .007	.003 .004	.001 0	.003 .005	.002 .004
LHN Interim Test LHN Test	.86 .87	.055 .046	.005 .011	.052 .049	.006 .007	.011 .004	.001 .002	.005 .006	.001 .001

Artifactual Recognitions

	Violation Category								
Speech Data	0	1	2	3	4	5	6	7	8
MWG Interim Test MWG Test	.16 .17	.066 .074	.053 .061	.26 .24	.030 .031	.13 .13	.003 .002	.23 .22	.072 .069
LHN Interim Test LHN Test	.40 .38	.052 .053	.046 .044	.21 .23	.018 .010	.077 .084	.017 .013	.13 .14	.048 .052

Figure 16. Rate of Occurrence of Violation Categoies for Real Recognitions and Artifactual Recognitions

value will be zero, and the parameter of the exponential portion of the distribution) are estimated from the distribution of QL values observed over Interim Test data.

If QL is in fact distributed in the modified exponential manner assumed, the computation of the log likelihood ratio performed in MINT is accurate. It is therefore of interest to determine the validity of this assumption.

As the mass concentration at zero and the parameter of the exponential part of the QL distribution are observed to be vocabulary item dependent, it is desirable to normalize observed QL distributions with respect to these parameters in order to avoid detailed consideration of two dozen distributions for each speaker. For this reason, the QL distributions have been linearized as described in the following paragraph.

A large set of independent samples of a random variable, distributed as QL is assumed to be distributed, can be converted to a set of numbers which are approximately uniformly distributed in the interval (0,1). To do this, first put the QL values in increasing order and assign running index i = 1....N to these values. For each i, replace the ith QL value (QL_i) by

$$f_i = \begin{cases} \frac{i}{n_o} \rho_o & \text{if } QL_i = 0 \\ 1-(1-\rho_o)e^{-\lambda QL_i} & \text{if } QL_i > 0 \end{cases}$$

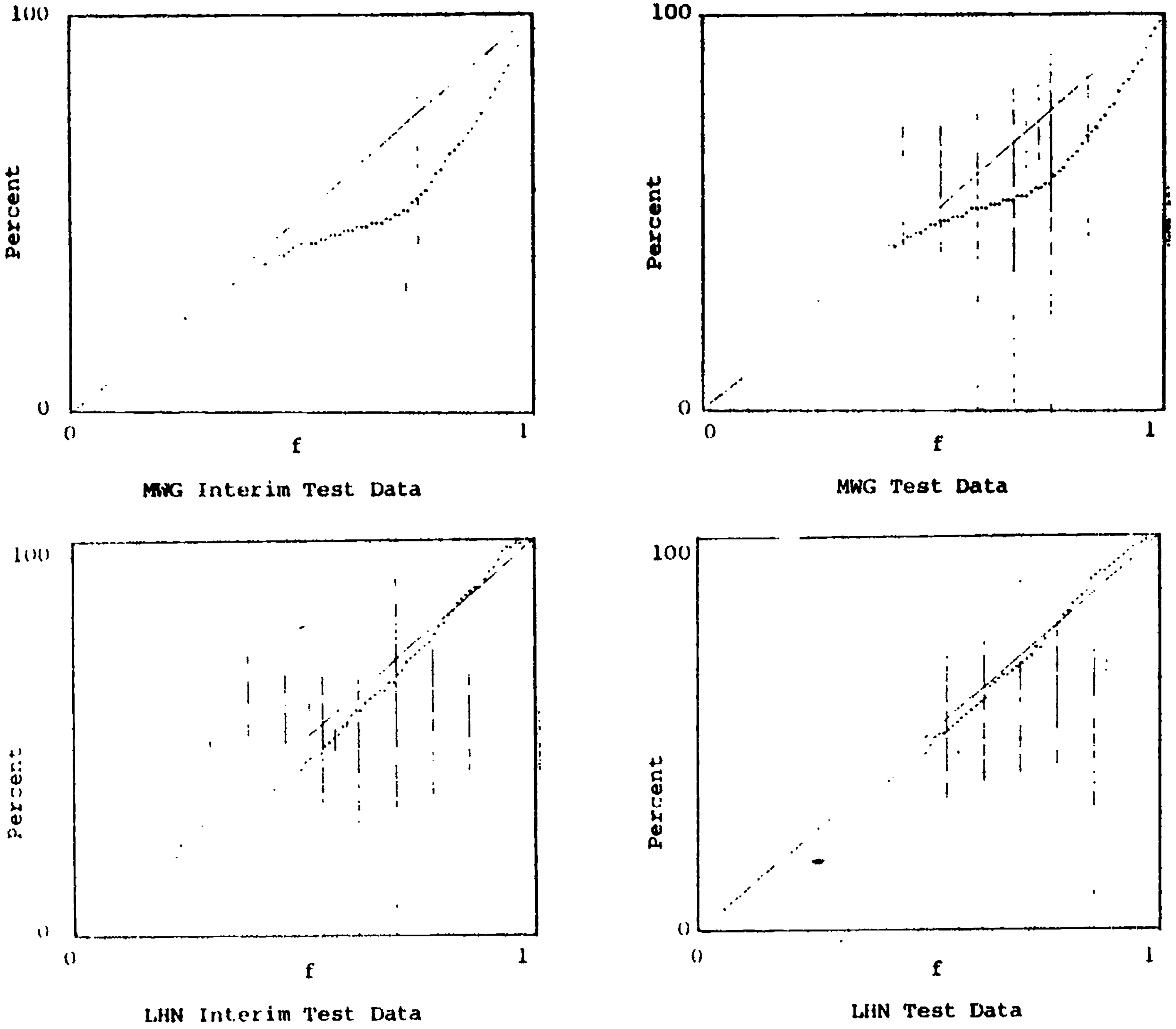
where

ρ_o is the probability that QL = 0

λ is the parameter of the exponential portion of the QL distribution

If ρ_o and λ are in fact the correct parameters of the QL distribution, and if QL has the assumed distribution shape, the resulting set of numbers approach a uniform distribution on (0,1) for large N. Correctness of the assumed distribution shape, and of the parameters ρ_o and λ can then be checked by plotting the cumulative distribution of the f_i values. If the distribution shape and parameters are correct, a straight line will result. More importantly, sets of QL values for different vocabulary items and for real and artifactual recognitions can be converted to sets of f values using the parameters appropriate to each set, and the sets of f values can be merged. The resulting large set of dat? will be uniformly distributed on (0,1) if the model and the parameters for individual vocabulary items and types of recognition are correct. A single graph thus checks the model and parameter validity for the whole vocabulary.

The PASS program STATSUM performs the conversion of QL values to f values just described, and the QLPLOT function in LICVAT generates a computer graph of the cumulative distribution of the merged f value sets. Results obtained in this way are presented in Figures 17 and 18.



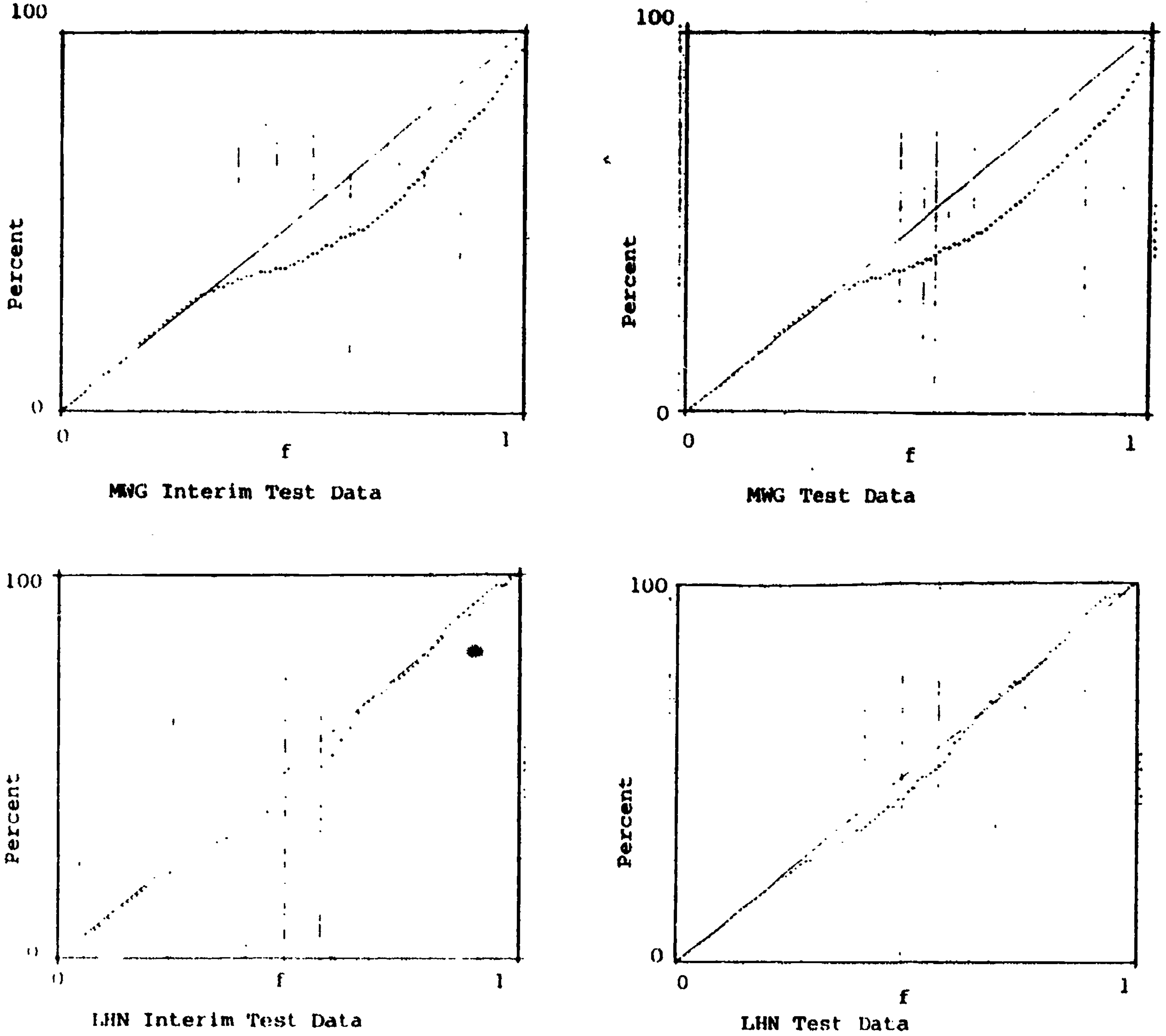
MWG Interim Test Data

MWG Test Data

LHN Interim Test Data

LHN Test Data

Figure 17. Computer Generated Plots of the Cumulative Distribution of f. Real Recognitions



MWG Interim Test Data

MWG Test Data

LHN Interim Test Data

LHN Test Data

Figure 18. Computer Generated Plots of the Cumulative Distribution of f. Artifactual Recognitions.

The graph of the cumulative distribution for f deviates significantly from a straight line for speaker MWG, but is quite reasonably straight for speaker LHN. This is probably due to different procedures used to generate the exponential parameters (λ) for each vocabulary item for the two speakers. For MWG, the λ values were estimated by visually comparing computer plotted cumulative QL distributions with exponential curves of known parameter. The distortion noted in the upper right hand portion of MWG's f distributions are what would be expected if there were a systematic bias towards estimating too high a value for λ. The fact that the lower left portions of those curves tend to be straight lines coincident with the graph diagonal indicates that the ρ_o values are correct. They were estimated as the fraction of observed zero values and were thus not subject to human error as were the λ estimates. In contrast, both λ and ρ_o estimates were derived objectively for LHN's voice, using programs in the VDGS.

The difference just noted between the two speakers' data suggests that the QL information source could be improved for MWG by re-estimating the λ parameters for each vocabulary item, using the unbiased mathematical procedure.

These graphs indicate that the assumed exponential shape fits the distribution of non-zero QL values quite well. If the data were distributed in some other way, with the parameter λ chosen to obtain best fit to the data, the curves would have an ogival shape rising above and falling below the graph diagonal in the upper right hand portion of the graph. The graphs also indicate that the parameters obtained from Interim Test data are descriptive of other speech data, as indicated by the similarity of the curves for Interim Test and Test data. Thus the QL statistical model appears stable.

ASSOCIATION MODEL. In an effort to exploit the fact that speaking certain vocabulary items may have a tendency to cause artifactual recognition of another vocabulary item, MINT detects and uses the temporal association of potential recognitions. If there is significant asymmetry in the rates of artifact production (for example, if speaking "five" usually causes artifactual recognition of "nine," while speaking "nine" seldom produces artifactual recognition of "five") association may carry information useful for recognition.

A set of associated vocabulary items and forms is ascribed to each potential recognition for this purpose. A vocabulary item is associated with a given potential recognition if there is another potential recognition of that vocabulary item type which overlaps sufficiently in time. The required amount of overlap (called the association criterion) is determined as described in Reference 2. Only the existence or non-existence of associated recognitions of each vocabulary item is noted, not their number.

The probability that a potential recognition will have an associated potential recognition of given vocabulary type is assumed to depend upon both vocabulary items and forms, and whether the former recognition is real or artifactual.

The PASS program STATSUM tallies the number of times each vocabulary item and form is found to be associated with real and artifactual recognitions of each vocabulary item and form and the LICVAT option DOGLEG prints these data. The probability that a real (or artifactual) recognition of given vocabulary item and form will have an associated recognition of given vocabulary item and form can be estimated directly from these tallies. The association data can contribute to correct recognition when the probabilities for real and artifactual recognition differ significantly. The natural measure of this potential is the likelihood ratio.

The assumed dependence upon vocabulary item of the associated recognition is shown to be factual by the data in Figure 19. These data show the estimated probability that various vocabulary items and forms will be associated with real and artifactual recognitions of the word "five," as determined for MWG test data. The likelihood ratio is seen to vary widely from unity. However, if one averages over vocabulary items, it is found that both real and artifactual fives have the same probability (.21) of having an associated recognition of unspecified type, resulting in a likelihood ratio of one, and no information for distinguishing real from artifactual recognitions. Similar results can be demonstrated for vocabulary items other than "five." Including vocabulary item dependence in the association model is, therefore, necessary in order to extract the available information.

The data of Figure 19 also show that association of a potential recognition of "five" with another recognition of any vocabulary item other than "nine" yields information useful in distinguishing real from artifactual recognitions. The exception is unfortunate, as "five"/"nine" discrimination is difficult.

The stability of association statistics can be demonstrated by comparing association frequencies observed for Interim test and Test data. These frequencies are shown in Figure 20 for LHN's enunciations of "point." The frequency observed in Test data is plotted against the frequency observed in Interim Test data to facilitate comparison.

INTERWORD TIMING MODELS. MINT uses three different models related to the relative time of occurrence of potential recognitions within an utterance. These three models treat the delay between the start of the utterance (sound detected by the preprocessor) and the beginning of the first recognition, the gap or overlap between successive words of the utteranee, and the delay between the recognition of the last word of the utterance and the cessation of sound.

Initial Delay Model. The delay between the beginning of the utterance and the start time of the recognition of the first word in the utterance is assumed to be distributed exponentially over positive values, with a mass concentration at zero. (This distribution was suggested by examining many cases during LISTEN's development.) The probability of a zero value and the parameter of the exponential portion of the distribution are assumed to be dependent upon vocabulary item and whether the recognition is really the first word spoken in the utterance or not. (Thus for the initial delay model, recognition of the second word actually spoken is an artifactual recognition of the first word spoken.)

Variation of the distribution parameters with vocabulary item, and stability of these statistics, are revealed by comparing estimates of the parameters derived from Interim Test data with estimates taken from Test data. Data for computing these estimates are provided by the PASS program STATSUM, and printed by the GAP DATA option in LICVAT.

	Vocabulary Item and Form											
	0	1	2 Initial	2 Non-initial	3	4	5	6	7	8	9	Point
Real "Five"	.033	.033	.10	.11	.49	.53	0	.13	.43	.11	.54	.055
Artifactual "Five"	.005	.44	.035	.13	.10	.17	.040	.025	.12	.41	.58	.49
Likelihood Ratio	6	.08	2.9	.85	5	3	0	5	3.6	.3	1.	.1

Figure 19. Estimated Probabilities and Likelihood Ratios that Potential Recognitions of Various Types Will Be Associated with Real and Artifactual Recognitions of the Word "Five." MWG Test Data.

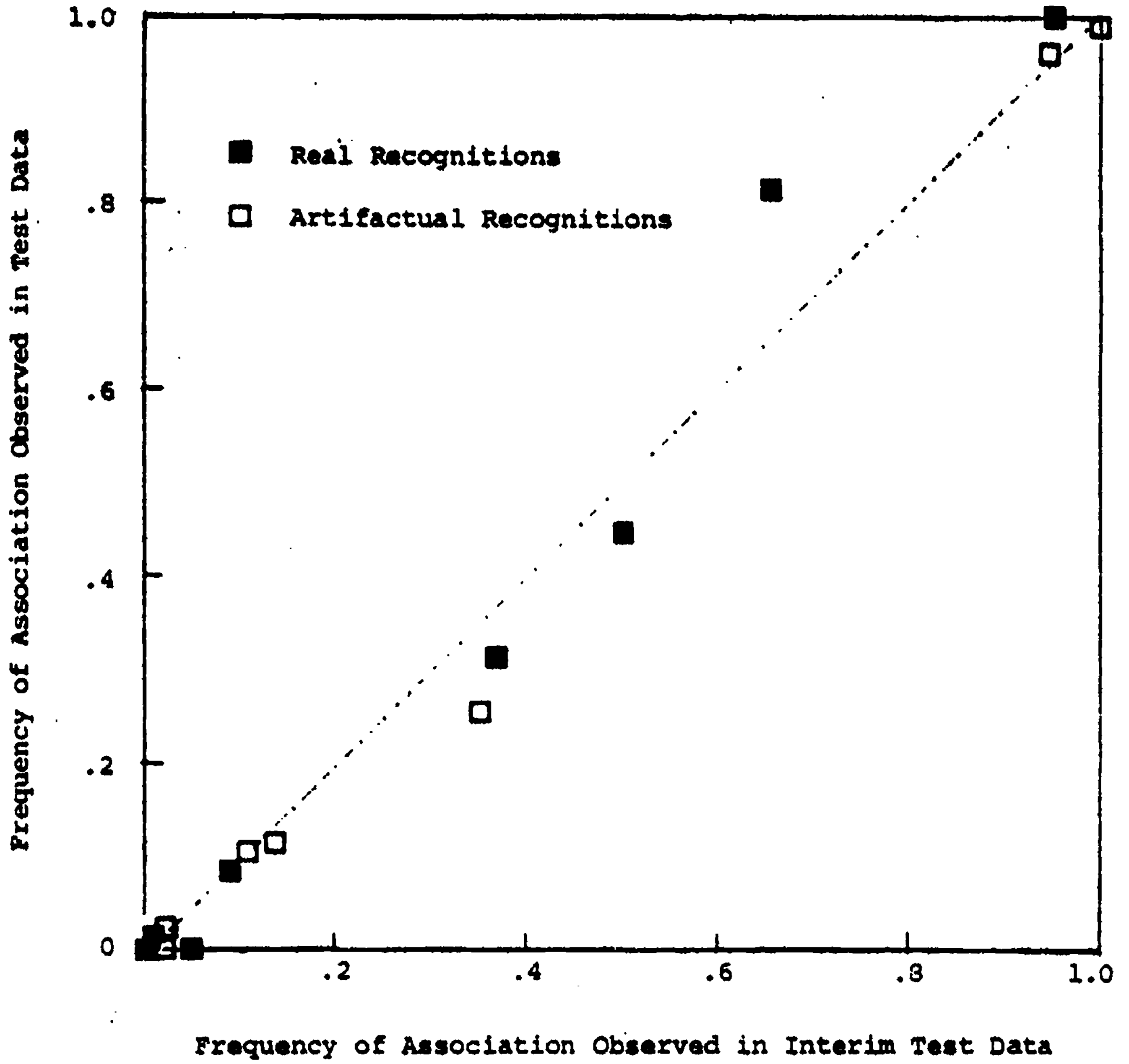


Figure 20. Comparison of the Frequency with which Various Vocabulary Items Are Associated with Recognition of the Word "Point" in Interim Test and Test Data for Speaker LHN.

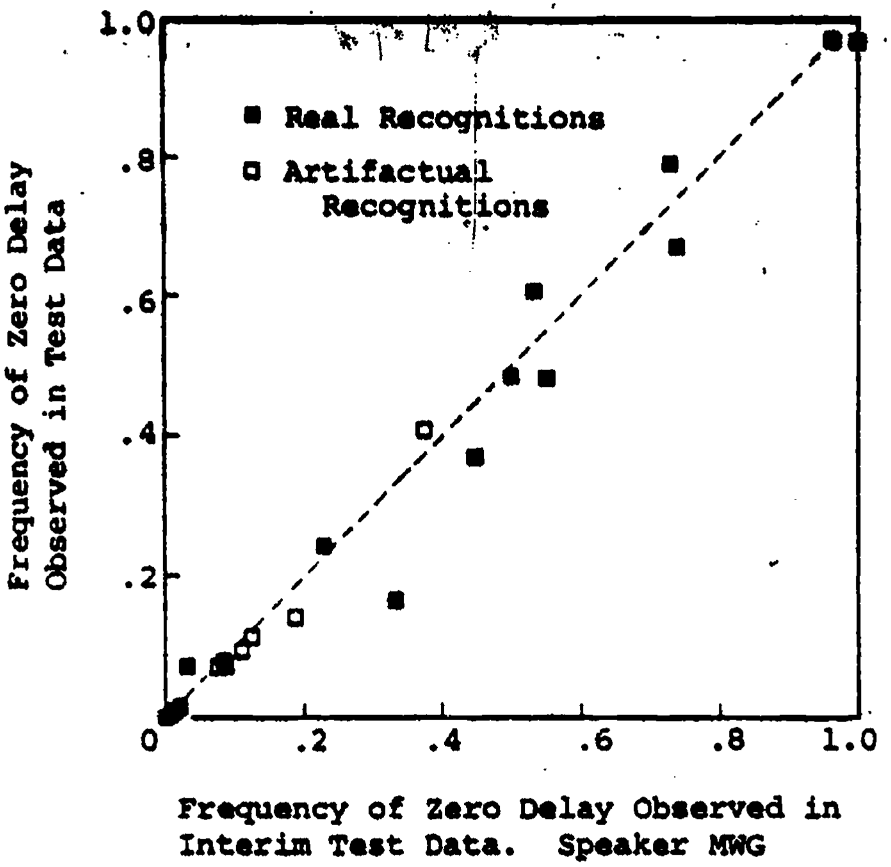
Figure 21 shows the frequency with which zero initial delay was observed for all vocabulary items, in Interim Test data and Test data. The wide variation of these ratios for various vocabulary items indicates the importance of using vocabulary item as an independent variable. This figure also shows that the variation in rate of occurrence between vocabulary items is greater than the variation from Interim Test to Test data, further validating the dependence upon vocabulary item, and showing the stability of the statistics. The wide disparity in frequency of zero delay observed for real and artifactual recognition, and hence the utility of this information source, is also apparent in this figure.

Figure 22 shows the mean of the non-zero initial delay observed in Interim Test and Test data for all vocabulary items. The reciprocal of this value is an unbiased estimator of the exponential distribution parameter. The time unit is one "count," the period of the interrupt signal from the preprocessor, which is approximately two milliseconds. These data indicate several interesting characteristics of the non-zero initial delay distributions.

First, non-zero initial delays are very much larger for artifactual than for real recognitions. The only exceptions to this rule are vocabulary items of initial form; for those vocabulary items, the non-zero initial delays for real and artifactual recognitions are comparable. (This is because potential recognition of the initial form of a vocabulary item is only allowed by MEX to start in the first fifty or so milliseconds of the utterance.) If the distribution of non-zero initial delays is in fact exponential, this indicates that artifact non-zero initial delays are distributed essentially uniformly in the interval where there is any reasonable probability of a delay being due to a real recognition.

Second, among non-initial artifactual vocabulary items, the variation of mean non-zero initial delay with vocabulary item is not a large fraction of the average value, and comparable to the variability between Interim Test and Test data. Combining this fact with the first observation, it appears that the initial delay model could be simplified by assuming non-initial delays for artifactual recognitions are distributed uniformly over the region of interest, with a density which is independent of vocabulary items. From a computational point of view, however, it turns out to be simpler to retain the assumption that the distribution is exponential rather than uniform, but with a distribution parameter which is independent of vocabulary item.

Third, the stability of the non-zero initial delay distribution for real recognition and artifactual recognition of initial form is suspect. This is almost certainly a problem of sample size, as several vocabulary items have high probability of zero initial delay, leading to very few cases of non-zero delay from which to estimate the mean. For example, in the corpus of utterances used in this project, each data set (Training, Interim Test and Test) contains thirty occurrences of each vocabulary item in the initial position (including the six cases where the item is spoken in isolation). If the probability of zero initial delay is 0.8, the expected non-zero delay sample size is six. An extended study in this area might reveal an appropriate simplification of this portion of the initial delay model as well.



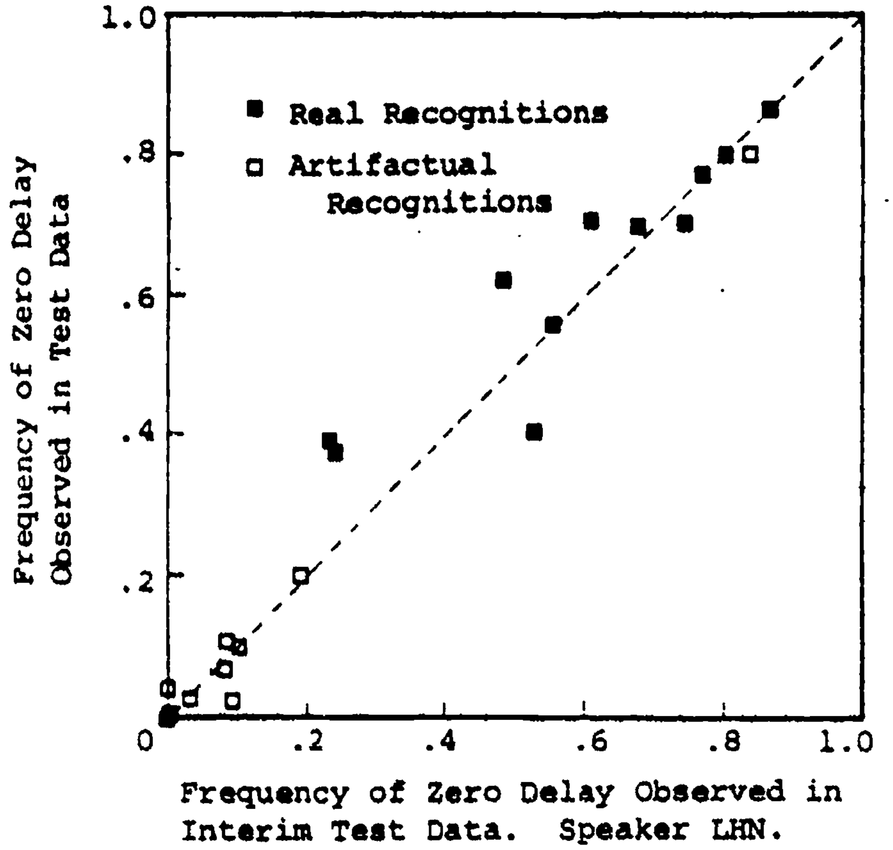


Figure 21. Graphs Showing Frequency of Zero Initial Delay.

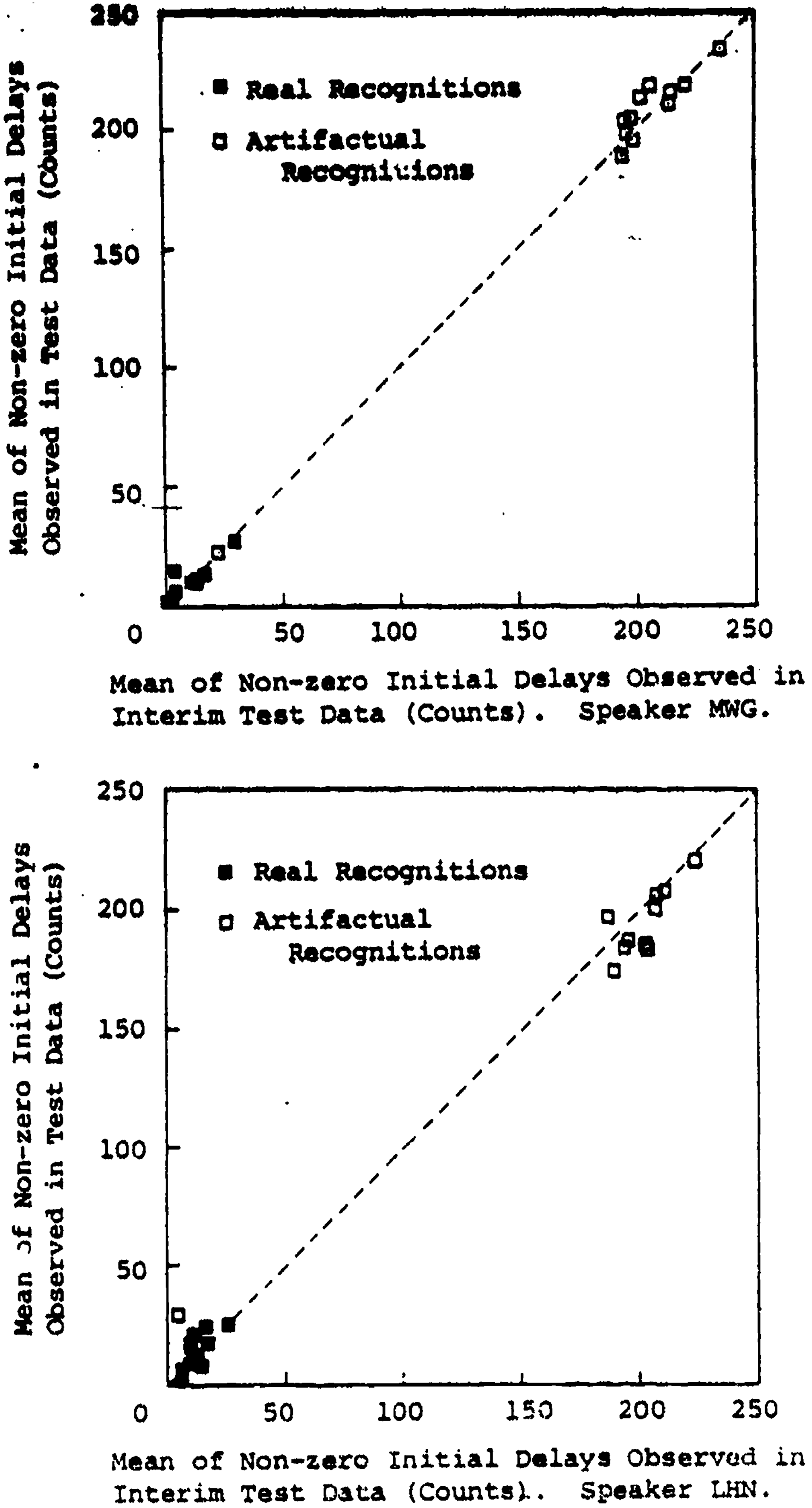


Figure 22. Graphs Showing Mean of Non-zero Initial Delay.

Final Delay Model. The interval between the time of recognition of the last word spoken in an utterance and the preprocessor's detection of the cessation of speech is assumed to be distributed exponentially, with distribution parameter depending upon vocabulary items and whether or not a potential recognition is really the last word spoken. (Thus "artifactual last words" include all artifactual recognitions and all real recognitions of words other than the last.) The PASS program STATSUM accumulates end delay values and averages those data for each vocabulary item and recognition type and the GAP DATA option of LICVAT prints the results. These data are shown in Figure 23 for all vocabulary items. (The reciprocal of the average delay is an unbiased estimator of the exponential distribution parameter.) Two different scale factors have been used in this figure to increase visibility of certain features of the data. The unit of time used is one "count", about two milliseconds.

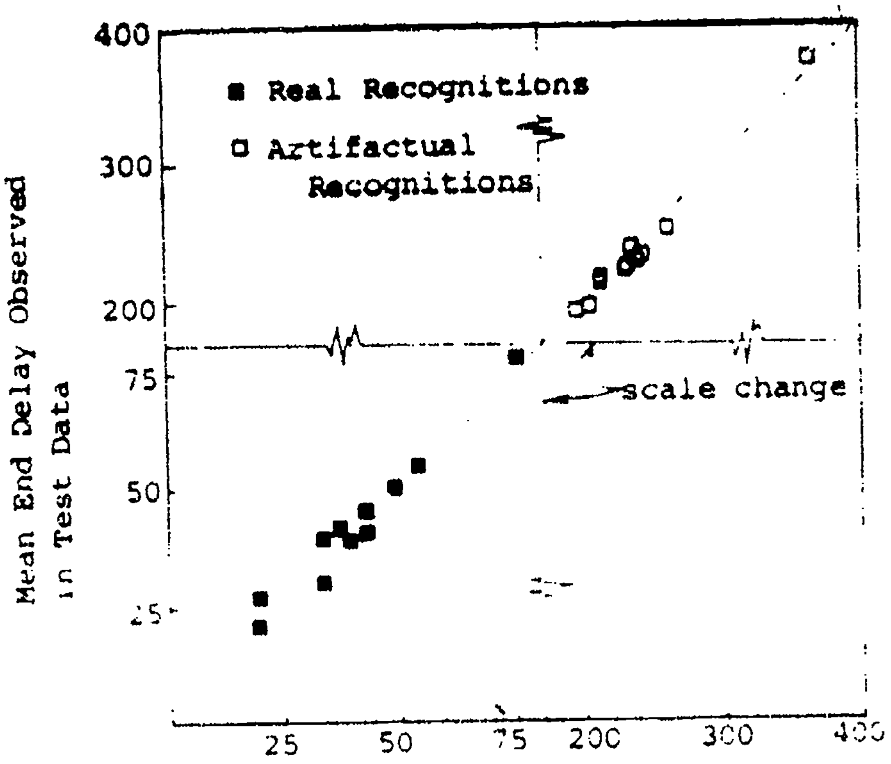
These data show that the final delay has significant vocabulary item dependence, and that the variation with vocabulary items is considerably larger than the variation from Interim Test data to Test data, for both real and artifactual recognitions. Therefore, unlike the initial delay model, the final delay model cannot be simplified by suppressing vocabulary item dependence without sacrificing information.

Interword Gap Model. The time interval between the end (recognition time) of one potential recognition and the beginning (start time) of another is assumed to be distributed in a symmetric limited exponential manner. That is, the probability density, as a function of the interword gap g is assumed to be of the form:

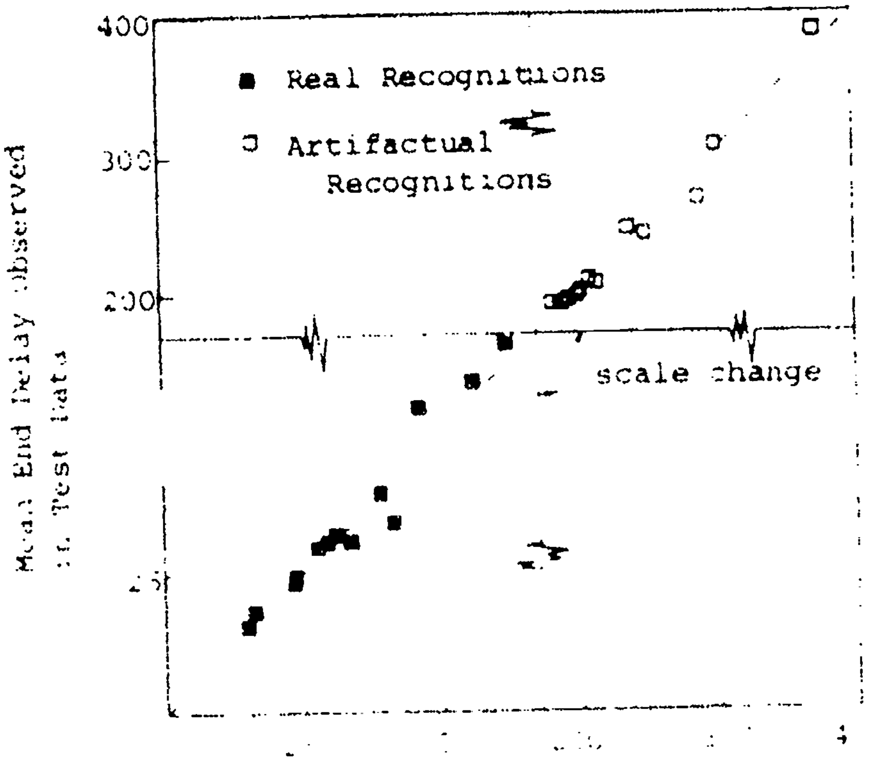
$$p(g) = \begin{cases} \dfrac{1}{4d} & \text{if } |g-\mu| \leq d \\[2ex] \dfrac{1}{4d}\, e^{1-\left|\frac{g-\mu}{d}\right|} & \text{if } |g-\mu| > d \end{cases}$$

where μ and d are parameters of the distribution. These parameters are assumed to depend upon the vocabulary item and form of the two potential recognitions, and on whether they are really recognitions of contiguous spoken words taken in correct order, or otherwise. The time interval between two potential recognitions is thus considered artifactual if the first is treated in MINT as a potential predecessor of the second, but they are not both real recognitions of contiguously spoken words.

With a dozen vocabulary items, this model requires considering a gross of vocabulary item pairs. Since each Magic Number Set of (55) utterances contains each sequential pair of vocabulary items exactly once, ("point-point" was excluded). Training, Interim Test and Test data sets contain six examples of each interword gap distinguished by the model. Statistical sample size is thus a serious problem in estimating the distribution parameters μ and d for each pair of vocabulary items.



Mean End Delay Observed in Interim Test Data. Speaker MWG.



Mean End Delay Observed in Interim Test Data. Speaker [illegible]

The small number of available interword gap samples also makes it difficult to validate treating vocabulary items as an independent variable in the interword gap model. Some justification for considering vocabulary items in modelling real interword gaps can be taken from the fact that data tend to have certain trends which would be expected on a phonological basis. For example, in the "six-six" case, one expects overlap due to the identical terminal and initial sound of the words involved. Similarly, one expects overlap of word pairs which share a stop, such as "eight-two." Word pairs which entail dissimilar sounds at their juncture, such as "seven-point" are expected to have larger than average interword gaps. The observed mean values of interword gaps, subjectively at least, seem to exhibit many of these anticipated tendencies, as is demonstrated in Figure 24. These data were obtained by the VDGS program GAPSTER, for speaker MWG's Interim Test data.

It is much less likely that vocabulary item dependence should be considered in the distribution of artifact interword gaps, since the phonological argument cannot be applied to artifactual recognitions or non-contiguous real recognitions. Little would probably be lost by simplifying the model by suppressing this dependence, but it is impossible to demonstrate that as fact with available data.

In an attempt to evaluate the stability of interword gap statistics and validity of the assumed distribution shape, the following procedure was used to normalize interword gap data. A derived random variable, f, can be computed from the observed random gap values, g, using the known parameters of the distribution. If f is related to g by

$$f(g) = \int_{-\infty}^{g} p(x)\,dx$$

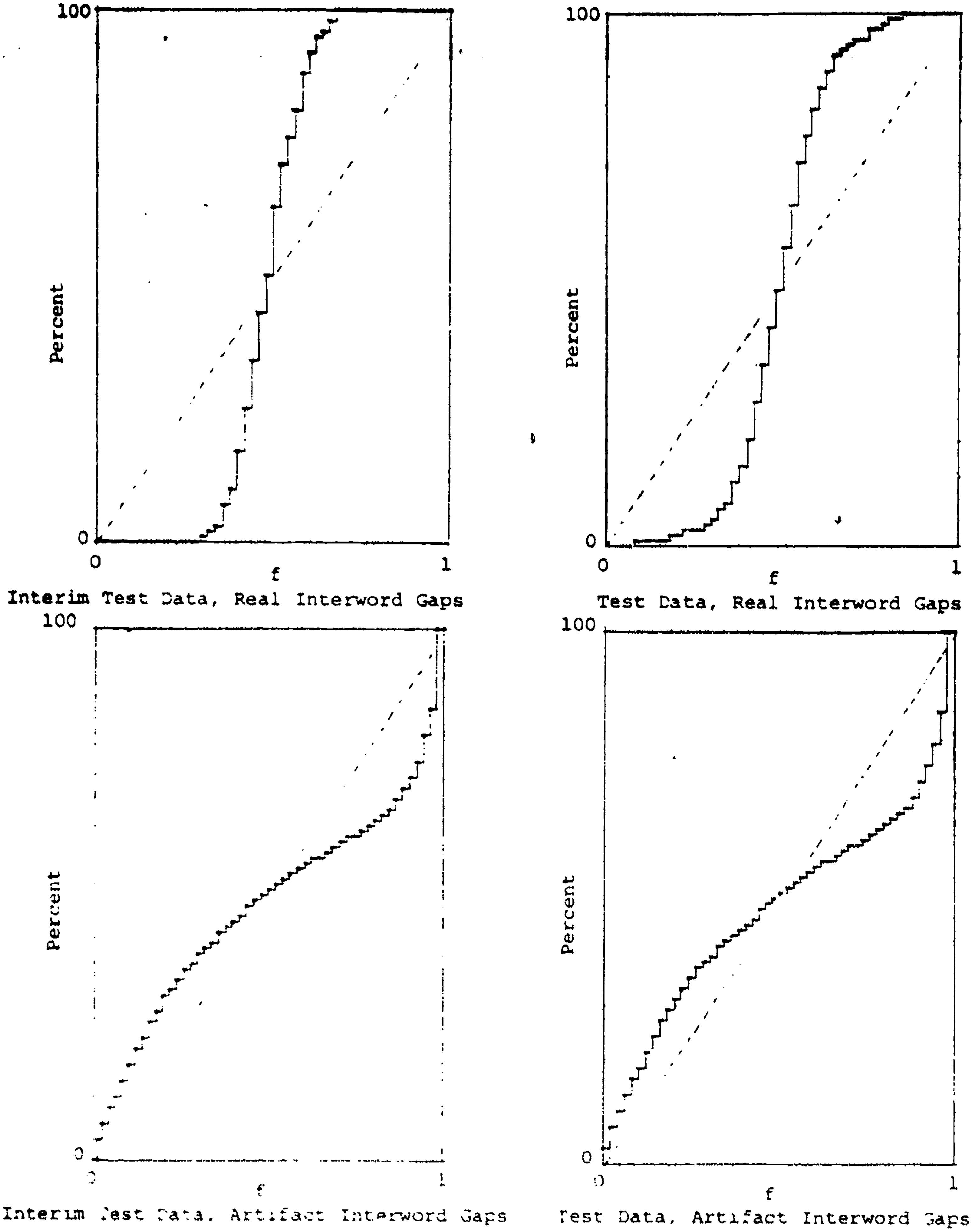
where p(x) is the probability density assumed for the gap data, f will be uniformly distributed on (0,1) provided the assumed distribution is correct. By using distribution parameters [illegible] and [illegible] appropriate to the instance of g, all f values thus found can be merged and their cumulative distribution plotted. If the assumed distribution shape and parameters are correct, a straight line will result.

The VDGS program STATSUM computes the normalizing function f and the [illegible] of the program LICVAT generates a computer plot of the cumulative distribution [illegible] taken from the computer plots are presented in Figures [illegible] features of these data are discussed in the following [illegible]

[illegible] data show that f is not uniformly distributed [illegible] the distribution parameters used in computing f are taken [illegible] data. The shape of the curves can be explained by the fact [illegible] parameters [illegible] operates [illegible] parameters [illegible] when small numbers

		Following Word										
		0	1	2	3	4	5	6	7	8	9	•
	0	36	27	37	40	72	72	49	40	61	36	56
	1	3	12	1	27	41	56	19	12	20	13	33
	2	9	47	22	20	66	68	17	37	42	35	60
	3	56	54	42	60	80	86	60	53	63	75	68
	4	25	11	20	36	39	58	37	17	8	13	24
Leading Word	5	-11	25	19	16	21	30	- 2	- 1	9	22	34
	6	- 3	35	29	23	28	46	- 2	- 8	19	46	27
	7	29	25	21	21	41	54	21	29	-27	- 5	40
	8	13	23	-17	23	21	40	29	7	12	38	24
	9	28	30	26	44	39	67	33	31	23	38	41
	.	23	18	20	23	35	50	19	19	18	33	--

Figure 24. Mean Interval Between End of Recognition of One Word and Beginning of Recognition of Succeeding Word. Speaker MWG



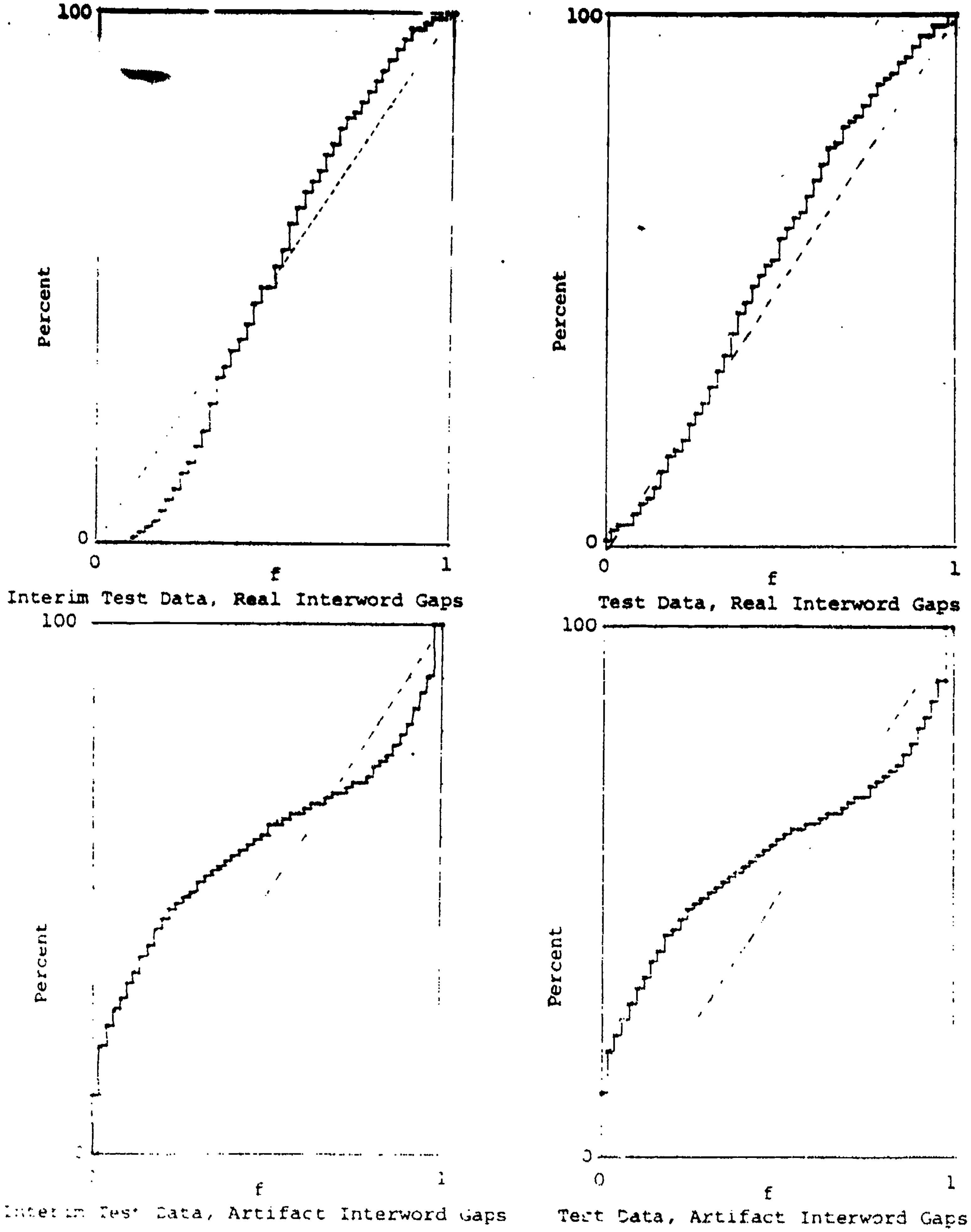
Interim Test Data, Real Interword Gaps

Test Data, Real Interword Gaps

Interim Test Data, Artifact Interword Gaps

Test Data, Artifact Interword Gaps

Figure 25. Cumulative Distribution of the Interword Gap, Normalizing Variable f. Speaker MWG



Interim Test Data, Real Interword Gaps

Test Data, Real Interword Gaps

Interim Test Data, Artifact Interword Gaps

Test Data, Artifact Interword Gaps

Figure 26. Cumulative Distribution of the Interword Gap, Normalizing Variable f. Speaker LHN

of samples are available. As a result, the parameters for several vocabulary item pairs describe a distribution which is wider than the data indicate, leading to fewer than expected f values near zero and one. Another factor which may be contributing to the paucity of real gap cases at the extremes would be that the distribution shape has too much weight in the exponential portions, vice the uniform portion. (The adopted distribution has twenty-five percent of its mass in each exponential segment.) No reasonable explanation is available for the greater deviation of the f distribution from linearity observed for speaker MWG than for speaker LHN.

The distribution of real gaps is very stable with respect to speech sample, as can be seen by comparing the f distribution for real gaps shown for Interim Test and Test data. This stability, and the similarity of the distributions obtained for the two speakers, suggests that substantial improvement in the modelling of real gaps can be obtained by reducing the small sample protection bias toward large d values, and perhaps changing the assumed distribution shape by reducing the mass in the exponential portions.

The f distribution graphs for artifact gaps show a deviation from linearity which is the reverse of that observed for real gaps. In each case, more than the expected number of f values are found near zero and one, and fewer near middle values. This is the result to be expected when the gaps are actually distributed more or less uniformly over a broad interval, including values where the density is modelled as decreasing exponentially. It is a clear indication that the assumed distribution shape is not appropriate for artifact gaps. As the distribution width (indicated by the parameter d) is much greater for artifacts than for real gaps, a superior model for artifact gaps would result from assuming that artifact gaps are uniformly distributed over an interval containing almost all real gaps. The almost linear portion of the f distribution near middle f corresponds to time values covering the region of interest for real gaps, so this linearity indicates that the locally uniform assumption is a good one.

Stability of the gap statistics is also indicated by the similarity of the artifact f distribution for Interim Test and Test data. This is another indication that improvement in the artifact gap model may significantly improve use of the gap information source.

This analysis reveals a tendency to underestimate real gap densities, and overestimate artifact gap densities, at middle f values. This results in a considerable underestimation of the likelihood ratio, and too little cost advantage being assigned for gaps observed in this region. For extreme f values, the density of real gaps is overestimated and the density of artifact gaps is underestimated, leading to overestimation of the likelihood ratio. As a result, extremely short or long gaps are not penalized by high cost to the extent they should be. The net effect of these model inadequacies is to underemphasize the gap information source by assigning costs which partially mask the true significance of typical and atypical gaps alike. This is a very interesting result in view of the fact that gap data are an important part of the interword timing information source, and this information source has been found to be the most productive information source used in LISTEN. Improvement of the gap model would then seem to offer significant potential for improving LISTEN's performance.

SECTION V

SUMMARY OF RESULTS AND CONCLUSIONS

SUMMARY OF RESULTS

The VIAS project is a continuation of the NAVTRAEQUIPCEN's exploratory development program for automated speech technology. It has contributed to that program by developing a working system suitable for laboratory concept development in the area of limited connected speech recognition which is readily modified for research purposes. This system permits the variation of parameters and evaluation and analysis of effects upon recognition results. Consideration has been given to increasing the number of speakers, automating the process of reference pattern creation, expanding vocabulary size, and transferring technology to a new preprocessor, all within the context of real-time recognition.

Specific results achieved by this project are summarized below.

TRANSFER OF TECHNOLOGY. The real-time-connected speech recognition system LISTEN has been modified to operate successfully with a new model of speech preprocessor.

EXTENSION TO NEW SPEAKERS. It has been demonstrated that LISTEN can achieve connected speech recognition accuracies in excess of ninety percent (word basis) for a new speaker.

EXAMPLE SET GENERATION. The importance of the method of generating sets of individual vocabulary items used in creating voice reference data has been demonstrated.

VOICE DATA GENERATION SYSTEM (VDGS). A unified body of computer programs for generating voice reference data has been developed. These programs automate the voice reference data creation process to the full extent practicable at this time. These programs exist in two forms: as an almost autonomous sequence of programs requiring an absolute minimum of human intervention, and as a collection of individual programs which can be exercised independently for research purposes. A detailed users manual has been provided for both versions of this system of programming.

PERFORMANCE ANALYSIS SUBSYSTEM (PASS). A useful, powerful, and convenient set of programs has been developed and exercised for analyzing the overall performance and many technical details of LISTEN's operation. A users manual also has been provided for using these programs.

VOCABULARY EXPANSION. The number of vocabulary items which can be accommodated by various VDGS programs has been increased toward the desired goal of thirty. The goal has been reached for several of those programs, and no fundamental barrier exists to reaching it for all programs.

ANALYSES OF LISTEN PERFORMANCE. Programs of the PASS have been used to analyse the significance of the several information sources LISTEN uses to obtain recognition. It has been found that these information sources vary considerably in their utility for recognition. Methods of automatically classifying and analyzing recognition errors have been developed and used. Among the many findings, it has been shown that most recognition errors result from failure to correctly select the correct alternative in a simple substitution decision. The statistical models used to represent the information sources have been examined critically with a variety of results. While the models have generally been shown to be effective, several specific modifications to simplify data collection or improve model fidelity (and recognition accuracy) have been suggested.

CONCLUSIONS

Results obtained in the VIAS project support four conclusions of general interest, as discussed in the following paragraphs.

MAGNITUDE OF THE VOICE REFERENCE DATA GENERATION BURDEN. Producing the VDGS was a major task, due to the number and complexity of the procedures used to produce voice reference data needed by the LISTEN real-time recognition programs. Using the VDGS to produce voice reference data for new speakers also requires a considerable amount of computer time and labor. These facts have made clear the important role that reference data generation requirements may have in determining the practicality of applying a connected speech recognition capability in a training environment.

LISTEN was developed with primary emphasis on real-time operation and exploitation of all information which might be present in the preprocessor output, and essentially no concern with the voice reference data production burden. Now that much has been learned about the nature of the information present in the preprocessor output, the opportunity exists to reformulate the recognition and reference data extraction processes in a way which will maintain or improve recognition performance while minimizing the reference data production burden.

INFORMATION IN THE PREPROCESSOR OUTPUT. Analyses performed using the PASS programs have verified the presence, and elucidated the nature, of information sources in the preprocessor output. Models of those sources posited during LISTEN's development have been validated to varying degrees, but the validity of the models is secondary in significance to the fact that those information sources have been isolated and demonstrated by objective means to be present and to have utility for recognizing connected speech.

THE ANALYTIC APPROACH. Equally significant is the fact that the approach used in this project to evalute LISTEN's performance has led to analytic procedures which reveal the character and relative value of different sources of information in a preprocessor's output. This approach is data intensive and costly in terms of computer processing requirements for developing and exercising the analysis programs, but it is the only approach which yields concrete knowledge about the information sources. As demonstrated in this report, this concrete knowledge of the information sources can clearly indicate improvements in the recognition of reference data generation procedures.

POTENTIAL. The analyses described in this report indicate only the potential of the PASS programs for providing data which yield insight into the nature of LISTEN, the speech preprocessor and the voice data generation problem. These programs can support the extension of the reported analyses as well as many new avenues of investigation. It is indicative of the power of these programs that even the relatively modest analysis effort which could be mounted within the resource limitations of this project has yielded recommended modification to improve LISTEN's recognition accuracy.

REFERENCE LIST

1. Use of Computer Speech Understanding in Training: A Preliminary Investigation of a Limited Speech Recognition Capability; Technical Report NAVTRAEQUIPCEN 74-C-0048-2, Logicon, Inc.; June, 1977

2. LISTEN: A System for Recognition Connected Speech Over Small, Fixed Vocabularies, in Real Time; Report NAVTRAEQUIPCEN 77-C-0096-1, Logicon, Inc.; April, 1978

3. Speech Understanding in Air Intercept Controller Training System Design. Technical Report NAVTRAEQUIPCEN 78-C-0044-1, Logicon, Inc., January 1979.

APPENDIX A

VOICE DATA GENERATION SYSTEM USERS MANUAL

A.1 GENERAL

The object of this appendix is to describe in some detail the use of the Voice Data Generation System (VDGS). Specifically discussed will be what is involved in the process which begins with the extraction of voice samples from the speaker and ends with the creation of a MIND file. The individual programs of VDGS also will be described. The main body of this appendix describes two methods of using the VDGS programs to prepare the MIND file which is necessary for the operation of LISTEN.

The operating environment for which the VDGS software has been prepared is one in which there is available a Data General S-130 minicomputer running under RDOS with a Threshold TTI-500 voice preprocessor and standard peripheral devices. The executable files for each of the individual VDGS routines are intended to function on the S-130. However, the VDGS software also will operate on a Nova 3 minicomputer, provided all routines are recompiled and all programs reloaded.

Program descriptions for VDGS routines are presented in A.8.

File descriptions for VDGS user-created files are presented in A.9.

Data files and compile and load macros are tabulated in A.10.

A.2 THE TWO METHODS OF VDGS

Before LISTEN can perform limited continuous speech recognition of a given speaker's voice it is necessary to construct a MIND file. A MIND file is a file containing the concentrated statistical essence of a voice, and many routines (twenty-four) are required to create it. We will describe two different methods for using these twenty-four routines to create the MIND file. One method, which we will refer to as the chain method, is to use one routine and two command macros to execute all twenty-four routines with operator intervention required at only one point. The other approach, which we will call the step-through method, requires an operator to execute each program separately and engage interactively with the programs. The chain method has some limitations which will be described later, but it essentially runs by itself. The step-through method is more flexible, but it requires relatively extensive operator input. Whichever method is chosen, it must be followed through to the creation of the MIND file. In the sequel we will describe both methods for using the LCSR statistical preprocessing package.

A.3 THE VDGS CHAIN

INTRODUCTION TO CHAINMIND

The approach to using the VDGS software which is simplest, in the sense of requiring the least input from the operator, is embodied in

CHAINMIND - the VDGS chain. CHAINMIND has three parts: (1) the extraction and compression of voice samples, (2) the creation of example spaces and transition letter sets, and then (3) all the rest of statistics gathering and statistical processing, including the building of the MIND file.

The first part of CHAINMIND is accomplished by the program EXTRACT which prompts the user to speak, extracts raw voice data from the TTI-500 preprocessor and compresses the data to a form usable by the remaining VDGS routines. The operation of EXTRACT is described later and some further comments about its use are included in A.4.

The second part of CHAINMIND is GENTL, a small chain consisting of the programs ESG and GZEC. ESG creates eleven example spaces, one for each of eleven vocabulary items; GZEC creates transition letter sets, also one for each item. The operation of GENTL is also described in detail later.

The third part of CHAINMIND is MAKEMIND, a chain of the remaining 21 VDGS programs.

USE OF CHAINMIND

To begin CHAINMIND, first use EXTRACT to create compressed data files for all utterances in eighteen magic numbers sets, MNSETA through MNSETR. This can be done over a period of time at the user's convenience. Probably not more than six magic number sets at the most (310 utterances) should be spoken at a sitting.

After all eighteen magic number sets of utterances have been spoken, the chain GENTL can be run. To do this make sure that the files ZESG.SV and ZGZEC.SV are on the speaker's directory (this directory should have a three letter name, and should hold all the compressed data files) as well as the data files PFILE and WIZ.ST and the command file GENTL. Having done that, type @GENTL@, and the example spaces ES$XXX$** and temporary transition letter set files TRLS**.TM and TRIX**.TM will be created.

When GENTL is finished, the user must intervene to pick the best transition letter set for each item. The procedure for choosing the best transition letter sets is described later when the program RESCUE is discussed. When the best transition letter sets have been determined and their "RESCUE indices" found, the user should create a file called REDEEM with the editor. The user then should enter into the REDEEM file the eleven RESCUE indices, in order from the first vocabulary item to the eleventh, one per line, in I2 format.

Once the file REDEEM has been created, the third section of CHAINMIND can be run. Again, all the MAKEMIND executable files must exist on the speaker's subdirectory, together with all the compressed data files, the magic number set files, and the file REDEEM. Then the user must create a file called WHERE with a single entry of the form "disk unit: subdirectory" indicating where the speaker's counter data files will reside - e.g. DP2:USG. To continue, type @MAKEMIND@, and (after 20-25 hours) the MIND.VD file is created.

In summary, the voice technician supervising the operation of CHAINMIND should proceed like this:

a. Make sure all the CHAINMIND routines and data files exist on the speaker's subdirectory. (See Table A1)

b. Run EXTRACT to create the compressed data files

c. Run @GENTL@ to create example spaces and transition letter sets

d. Pick the best transition letter sets and create the REDEEM file. Also create the WHERE file.

e. Run @MAKEMIND@ to execute the remaining VDGS routines and create the MIND file.

SOME COMMENTS AND CAVEATS

The CHAINMIND method of VDGS processing has some rigidities and limitations which must be pointed out.

a. CHAINMIND is limited to the use of eleven machines and does not allow the option of creating universal machines for the special handling of initial words in an utterance.

b. The magic number sets to be used for training, interim test, and test data are fixed in CHAINMIND. The sets MNSETA through MNSETF are used for training data, MNSETG through MNSETL for interim test data, and MNSETM through MNSETR for final test data.

c. Some examples in the ESG-created example spaces may be too long for processing by GZEC and LOOPER, and these examples will be ignored.

d. There is no way for the user to intervene and remove special bad cases in the counter data file CDAT.RV created by REVEXA and REVEX. This mostly has the effect of increasing the false alarm rate later on in the other programs.

e. Perhaps most significantly, the CHAINMIND method has a poorer facility for recovering from abnormal or error situations than the step-by-step approach. This means that there are abnormal situations with which CHAINMIND cannot cope and will crash.

A.4 THE VDGS STAND-ALONE VERSION

The other method of using the VDGS programs is a step-by-step interactive procedure wherein the user executes each program in turn, responds to its prompts, and examines its output as necessary. A list of the VDGS programs in the order in which they are to be executed appears in Table A1. A description of this step-by-step approach follows.

This approach has some obvious advantages over the CHAINMIND approach. First of all, the programs can be run individually in relatively

TABLE A1. VDGS Programs in Order of Execution

1. EXTRACT
2. ESG
3. GZEC
4. RESCUE
5. SIGH
6. LOOPER
7. REVEXA
8. RVDIT
9. COVERT
10. INVERT
11. CROAK
12. REVEX
13. RVDIT
14. CROAK
15. ADDER
16. AVRAJ
17. CRAP
18. GAPSTER
19. SORTRA
20. SORTRB
21. GAPSTER
22. MUTE
23. GLOVE
24. TAILOR
25. BUILDER
26. DEALER
27. PHEW

NOTE: Three routines, RVDIT, CROAK, and GAPSTER, are run at two different stages.

small blocks of computer time and do not require a 20-hour block as does MAKEMIND. Secondly, error situations and abnormal conditions are much more easily responded to than in the CHAINMIND approach. If an error occurs in the step-by-step apprpach, one need only back up a step or two and restart. Thirdly, as will be seen in the sequel, the step-by-step aporoach offers a degree of flexibility not available in CHAINMIND.

With these observations in mind, let us consider the VDGS programs in their order of execution.

EXTRACT

The process of voice data extraction begins with the collection of voice samples by the program EXTRACT. For each utterance EXTRACT creates a compressed data file (-.CD), and an optional raw data file (-.RD), all the while maintaining a listing file (EXTOUT.LS) if desired. It is the set of compressed data files that is used in the remainder of the voice data generation procedure.

Probably the simplest way to collect voice data samples is to proceed as follows: Take a disk which has been formatted and initialized and which is substantially empty. Create a subdirectory with a three-letter-long name, and copy onto this subdirectory the set of magic number sets MNSET* which are to be used, as well as the file EXTRACT.SV. This subdirectory will then hold all the -.CD files and any -.RD and listing files created by EXTRACT. The separate description of the program EXTRACT tells how to proceed from here. But some additional comments are in order.

1. The room where the voice extraction is to be done should, of course, be kept as quiet as possible to avoid excessive noise in the voice signal. The volume adjustment should be set so that the meter registers about 0.8 when the word "five" is spoken. The microphone headset should be adjusted so that it is comfortable and so that the microphone itself is about 4 cm. from the speaker's mouth.

2. The number of voice samples taken during LCSR and VIAS work amounted to eighteen magic number sets worth of utterances - six designated training data, six designated interim test data, and six designated test data. It is probably a good idea to limit the speaker to three magic number sets at a sitting to avoid degradation in the voice samples due to speaker fatigue or boredom. For the step-through operation of the VDGS routines, it is possible to use fewer than six magic number sets apiece for training, interim test, and test data; but we still recommend that altogether eighteen magic number sets worth of data be used.

3. Each magic number set run, if desired, creates a listing file (EXTOUT.LS) which holds formatted versions of all compressed data files (and raw data files if those also are being saved). If listing files are desired for more than one magic number set, the file EXTOUT.LS should be renamed at the end of each magic number set run. This listing file is enormously large when both raw and compressed data are saved. It is [illegible] [illegible] and printing it consumes a great deal of time and

paper. This file was saved and printed for one magic number set at the beginning of VIAS, as part of a check that the TTI-500 preserves the same features that were identified in the VIP-100.

d. Utterances which are misspoken should be deleted in the sense that the corresponding -.CD and -.RD files should be deleted. Then the utterances should be respoken using the alternate mode of extraction permitted by EXTRACT when no prompting file is used.

At the completion of voice data collection for a speaker, we recommend that any raw data (-.RD) files created by EXTRACT be moved to a separate disk or to tape. They are not needed in the rest of the VDGS processing, and they take up a great deal of space on the disk. Also, any listing files created by EXTRACT should be handled in the same way.

ESG

Once voice data has been collected and compressed by EXTRACT, the next task of the VDGS is to create example spaces for use by GZEC and LOOPER. The program ESG is responsible for the creation of example spaces. The separate program description for ESG explains basically how to operate it, but perhaps a few suggestions are in order.

The example space name must begin with "ES", but the rest of the name is open to the user. We suggest that the example space names have the format

```
ES$XYZ$nm
```

where XYZ are the initials of the speaker and nm is the vocabulary item number (so, for example, Ulysses S. Grant's example space for item 4 would be ES$USG$04.)

Also, the user must create a prompting file containing the names of all compressed data files to be used in building the example spaces. A version of this prompting file, called PFILE, is delivered with CHAINMIND; but this prompting file assumes that the training data came from magic number sets MNSETA through MNSETF, and consequently that the example spaces are to be built from compressed data files A, B, C, D, E -.CD. Should the user want a different prompting file, the CLI command BUILD should be used. For example, to build a file of names of compressed data files corresponding to magic number sets MNSETG and MNSETK, type

```
BUILD prompting file name G-.CD K-.CD.
```

Then the user must use the editor to insert carriage returns between each entry and to eliminate carets.

Besides the prompting file, the routine ESG also requires W17.ST, a file of length and stretch factors. This is a canonical file to be used for all speakers, and it is delivered in the software package.

In its step-by-step form ESG runs once for each vocabulary item. The length of time for each run should be about 20 minutes.

Once ESG is run and the output examined, it may be the case that ESG has marked some words as too long for further processing. The user has two options at this point: (1) continue and let the program GZEC and LOOPER ignore the words which are too long, or (2) use the example space editor ESDIT to modify the word lengths. For an explanation of the operation of ESDIT, see the separate program descriptions. If the user chooses to bypass ESDIT and let GZEC and LOOPER ignore some examples, then the data base used to construct transition and loop letter sets will be reduced to the extent of the number of ignored words.

In lieu of using ESG to generate example spaces automatically, one could also use the program GWIZ to facilitate hand-marking the training data and the program MEND to create example spaces using the data produced by hand marking. For descriptions of these programs, see the VDGS auxiliary programs.

GZEC

Once example spaces have been created, the VDGS is ready to generate transition and loop letter sets. Since the collection of transition letter sets is the single most critical item in the VDGS data base, it is mandatory that each transition letter set be generated correctly. The program GZEC, embodying the critical algorithm GENRLIZ, generates the transition letter sets. For an explanation of GZEC, see the separate program descriptions. GZEC runs once for each example space (and consequently for eleven vocabulary items should be run eleven times). Each run of GZEC takes about 20 minutes. Since some of the questions the user will be asked by GZEC are not entirely self-explanatory, we make some suggestions for responses below.

a. The listing file should be printer and not disk, to conserve disk space

b. Do not change the value of SDCOEFF

c. Do not use the cost-weight factors

d. There is no existing machine to be generalized

The operation of GZEC does not produce a single collection of transition letter sets to be used. Rather, GZEC keeps a history of the transition letter sets formed at each stage of its operation. It is up to the user - using the routine RESCUE - to pick out the best transition letter set for each vocabulary item.

Since it may happen that a particularly bad exmple of an utterance occurs in the example space, or that a bad "cutting" of a vocabulary item within an utterance has occurred, any collection of transition letter sets may be resurrected as long as the temporary files TRLS**.TM and TRIX**.TM still exist. This "resurrection" is done using the program RESCUE which asks for the "RESCUE INDEX." The "RESCUE INDEX" corresponds to that number on the cost graph produced by GZEC indicating the desired set of transition letter sets.

The correct RESCUE index for each vocabulary item is determined by looking at the GZEC printout. Normally the last transition letter set formed by GZEC is the right one. In this case, look at the cost graph at the end of the particular GZEC run, and determine the last "machine number" corresponding to the column of utterance names on the left hand side.

If it should happen that the last transition letter set formed by GZEC differs significantly from the next-to-last or next-to-next-to-last, then an earlier machine number should be chosen. In this case "differ significantly" means that the final transition letter set was formed by dropping three or more transition letters from the preceding transition letter set. In a rare case the last transition letter set will be "significantly different". Still rarer, the last two transition letter sets will be significantly different. Once these machine numbers have been chosen by the user for each vocabulary item, we're ready to run RESCUE, pluck out the transition letter sets corresponding to those machine numbers, and set up the transition letter set files to be used for the rest of VDGS processing.

RESCUE

To run RESCUE, follow the instructions in the separate program description. In RESCUE, the term "rescue index" means the same thing that "machine number" does in GZEC printout. RESCUE must be run once for each vocabulary item. Its execution time in minimal.

We recommend that the user not delete the files TRLS**.TM and TRIX**.TM when given this option by RESCUE. Should the transition letter sets created by RESCUE be accidentally deleted or become inaccessible, they can be re-created if the temporary files TRLS**.TM and TRIX**.TM still exist. Otherwise, it would be necessary to run GZEC all over again.

SIGH

The program SIGH is run next. It checks the transition letter sets one-by-one to see if their length exceeds 13. If so, the length is reduced by omitting the letters with most "?" featues. If not, SIGH simply passes on to the next item.

LOOPER

Once transition letter sets are created, rescued, and checked, the loop letter sets can be found by the program LOOPER. LOOPER must be run once for each example space and so must be run once per vocabulary item - eleven times for eleven items. The run time for a single LOOPER run in this environment is about 30 minutes. A complete description of the operation of LOOPER is included in the program descriptions.

REVEXA

The real statistical data collection process begins with REVEXA after the transition and loop letter sets have been created. The purpose of REVEXA is to collect counter data statistics, i.e., statistics of time of residence of an incoming letter from an utterance in a transition or loop letter set. To this end, REVEXA must be run over all training data - that is, over all magic number sets used to generate the training data. The explanation of how to run REVEXA is included in the program descriptions. Recommended responses to some of the prompts are given below:

a. The mode of data acquisition should be 3. The magic number sets to be used here should be the ones designated for training data.

b. All optional printing should be done. A great deal of information about REVEXA and the recognition process in general is contained in these printouts.

c. When, on the second and succeeding runs of REVEXA, the user is asked if the CDAT.RV and CIDX.RV files are to be deleted, the answer should be "no". In this case the program will continue to append to the old files, and this is what is needed.

d. Later on we will explain why the user might choose to run one or two initial machines. If the user is doing so, he must enter the vocabulary item number for each initial machine and also a stop time (in TTI-500 time count units) for that initial machine.

e. For REVEXA, the user should request that only the machines in the utterance should be used.

A large file of counter data statistics is created by running REVEXA over the six magic number sets constituting training data. For some of the utterances processed by REVEXA, the subroutine MINIMINT (which mimics the operation of the MINT part of LISTEN) cannot come to a conclusion. In that case the user has two options: (1) ignore the misses and run RVDIT with no counter data record modifications, or (2) list the record numbers of all items occurring in a MINIMINT failure, and create a file RVCARDS in the format described in A.9., with record number entries for all the records to be flagged as "real." If the misses are simply ignored, the number of artifacts generated in subsequent routines will be somewhat larger; and the distinction, between real recognitions and artifacts, blurs in proportion to the number of items ignored.

The run time of REVEXA is abcut thirty minutes per magic number set.

RVDIT

The program RVDIT creates individual counter data records for each vocabulary item and, if desired, deletes from consideration all records specified in the file RVCARDS. The use of RVDIT is further explained in the program descriptions. Run time for RVDIT is about twenty minutes.

COVERT

The routine COVERT is run next. Its principal function is to create the covariance matrix for each machine. The details of its operation and use are explained further in the program descriptions.

INVERT

The routine INVERT is the next step. It inverts the covariance matrices created by COVERT. Its use is also explained later. In running this routine the user has the option of computing and printing the eigenvalues of the covariance matrices. The actual eigenvalues are not used later in the processing, so they may or may not be computed at the user's discretion.

CROAK

The last routine which operates using training data is CROAK. The program CROAK is an eclectic routine which performs all manner of statistical computations and prints plots of δ and μ distributions. A discussion of the operator's interaction with CROAK appears in the program descriptions of this appendix. CROAK must be run twice over each vocabulary item - once to generate statistics about real recognitions, and once to generate statistics about artifacts. So, for the first CROAK run, the user should:

a. Answer "2" to the question about modes

b. Save the probability statistics

c. Set starting machine number = 00 and end machine number equal to the last machine used (10, 11, or 12)

Then CROAK runs about thirty minutes. For the second CROAK run, the user should:

a. Answer "4" to the question about modes

b. Enter starting and end machine numbers as before

Than CROAK runs again for about thirty minutes.

SOME CLEANUP

At this point the user should do some disk cleanup. He can and should delete the following files: CDAT-.RV, CIDX.RV, QDAT-.RR, QDAT-.AF, MUDT-.RR, MUDT-.AF, RVX.ST, and RVCARDS.

REVEX

Now we begin to use the interim test data. The general description of the use and functions of the program REVEX is contained in the program descriptions, but there are a few suggestions we should make.

a. The mode of data acquisition should be 3. The magic number sets to be used here should be the ones designated for interim test data.

b. All optional printing should be done.

c. When, on the second and succeeding runs of REVEX, the user is asked if the CDAT.RV and CIDX.RV files are to be deleted, the answer should be "no". Then REVEX will continue and append to the old files.

d. If the user is running one or two initial machines, he must, at the appropriate prompt, enter the vocabulary item number for each initial machine and also a stop time (in TTI-500 counts) for each initial machine.

e. The user should request that all machines, not just the ones in the utterance, be run. This is important because this is the point at which data about artifacts is gathered.

With REVEX, as with REVEXA, a large file of counter data statistics is created by running the program over the six magic number sets of interim test data. Also, some of the utterances will not be recognized correctly by the MINIMINT subportion of REVEX. In these cases the user once again has the choice of ignoring the MINIMINT failures and proceeding, or of creating the file RVCARDS of records to be flagged as "real." If this option is chosen, all record numbers corresponding to real recognitions should be entered in RVCARDS.

RVDIT

Run RVDIT just as before on REVEX output files CDAT.RV and CIDX.RV.

CROAK

Run CROAK just as before.

ADDER

The next VDGS routine to be run is ADDER. This program builds a table of transition and loop letter set violations for each utterance processed by REVEX. A complete description of ADDER is given in the program descriptions, and the only additional suggestion to be made is that the output should not be directed to disk since disk space is probably sparse at this point.

AVRAJ

The program AVRAJ is the next step in the process. AVRAJ computes and prints the average word length for all vocabulary items. The routine should be run as described below.

CRAP

The critical association parameters are determined by CRAP. CRAP should be run as indicated in the program descriptions. As usual, the listing file should be directed to the printer and not to the disk.

GAPSTER

The program GAPSTER is primarily responsible for creating the gap matrix and the QASM matrix needed in the MIND file. The operation of GAPSTER is described in the sequel in the program descriptions, but a few comments about user inputs to GAPSTER are suggested below.

a. The critical association parameter entered should be 1.0.

b. The real standard deviation spread factor for the gap matrix also should be entered as 1.0.

c. The quartile and mean calculations are optional and not used in later processing.

d. Disk file output should not be chosen.

SORTRA

Run SORTRA to sort the file GAPMAX.

SORTRB

Run SORTRB to sort the file CONGAP.

GAPSTER

Delete the files QASM.DT, GAP.DT, and GAPMAX, and re-run GAPSTER as before.

MUTE

Run MUTE to compute the L-counter parameters MDLA** for each machine.

GLOVE

Run GLOVE to do the curve fitting for CROAK-generated δ-distributions

TAILOR

Run TAILOR to compute the T-counter parameters MDTA** for each machine.

BUILDER

Run BUILDER to create the machine data file.

DEALER

The program DEALER creates the MIND file. For the most part, its operation is described in the program descriptions of this appendix. However, there are a couple of required operator inputs that are not completely self-explanatory. In the first place, the "revision number for this job" should be entered as "*0" for all speakers. Secondly, when there are one or more universal machines being run, the program DEALER will ask for vocabulary identification number and end time for each machine, and these must be supplied by the user. For example, if machine 11 is a universal machine for vocabulary item 2, an appropriate response might be "2,25" when vocabulary item identification and end time are asked for.

PHEW

Then the process of the VDGS is completed by PHEW which finishes the building of the MIND file. The only response required of the operator here is the entry of the total number of vocabulary items used.

A.5 THE AUXILIARY PROGRAMS

The auxiliary programs delivered with the VDGS are: GASP, ESDIT, ESGDIT, MEND, and GWIZ. Here we describe the function of these auxiliary routines and indicate how they add to the flexibility of the VDGS.

The program GASP has the simple function of printing the transition letter sets after the program RESCUE has been run, or printing the merged transition and loop letter sets after the program LOOPER has been run. Using GASP the user can see, and group together on hardcopy for future reference, the transition letter sets for each vocabulary item (with accompanying loop letter sets, if desired).

The programs ESDIT and ESGDIT are both concerned with the editing of example spaces. ESDIT allows the user to change individual start or stop times in the example space using either an ESG or a GZEC print. The reason that ESDIT is sometimes used is that the individual start/stop times in the example spaces are sometimes bad - either the time duration for the word is too long, or the word has been "cut out" from the utterance in a less than satisfactory way. This "bad cutting" can arise from either an anomaly in the automatic example space generator ESG, or a human error if manual hand-marking is done using GWIZ and MEND.

In any case, if the user wishes to modify the individual start/stop times in an example space, just what numbers are entered depends upon what program's output is being used. If an ESG printout is being used, simply enter the new start and stop times when the program requests them. If a GZEC printout is used, the situation is a little more complicated. If the old beginning stop time is T_b and the user wishes to change this to T_b, enter $T_b - T_b + 1$ as "new starting record". So, if the beginning record number is correct as is, enter 1. To change the end time from T_e to T_e, enter T_e - (T_b + total number of records in word) + 1 as "new ending record".

The program ESGDIT is designed to operate on an existing example space file to produce a new example space file in which all utterances beginning with the vocabulary item specified are omitted. The operator of ESGDIT is explained fully in the program descriptions. Just why ESGDIT is used is explained below where universal and initial machines are discussed.

The routines GWIZ and MEND are programs to be employed when manual "hand-cutting" of utterances into individual words is to be used as a step in the creation of example spaces. The technique of hand-cutting is described later. The general procedure for the semi-manual creation of example spaces is as follows:

a. Create a file GWIZ.CD which holds the names of all compressed data files corresponding to training data. This file should hold one file name per line, left justified. A relatively painless way of constructing this file is to use the BUILD command at the CLI level as was previously described for the program ESG.

b. Run GWIZ, following the instruction given in the separate program description.

c. Hand cut the GWIZ printouts, noting start and end times of each word within each utterance.

d. Create a file MEND.WD holding all this data from hand-cutting (the format of this file is described in the description of MEND).

e. Run MEND to create the example spaces.

A.6 HAND-CUTTING THE DATA

The process of hand-cutting data to separate words within an utterance is as much of an art as a science and is best learned by doing. However, there are some rules of thumb and general guidelines that the speech technician might wish to consider.

a. The program GWIZ, itself, indicates locations of words with utterances on its printout. These are quite helpful but generally are not refined enough to be used as more than guidelines.

b. Whoever handcuts data must become extremely familiar with the hard copy presentation of an utterance and the variety of patterns associated with each particular item. It is a good idea to start with utterances consisting of a single word and to compare those with utterances where that word is only a part. The important thing is to be able to distinguish words visually and locate the interword boundaries. Because of co-articulation effects, it is important to allow overlap of word boundaries; rarely, in the context of hand-cutting, should the utterances be divided into non-overlapping segments. The idea here is that the transition letter set maker, GZEC, will discern the important structure within the utterance, and that one should not attempt to make too fine or toc subtle distinctions during the hand-marking process.

c. The geometrical configurations of the vocabulary items are a significant aid in marking the data. The "shapes" of words should be learned well before much hand-marking is done.

d. Sibilants and fricatives are a big help in marking the data. The "s", "x", and "th" sounds, when identified, make the demarking procedure much easier.

e. The relative letter counts indicated as the GWIZ printout help pick out longish sounds, e.g., "...ee" in "three", ""...oo", in "two", etc.

f. Features 16-18 in GWIZ printout are generally set for fricatives, e.g., "s" in "six"; features 15-18 are often set in the "x" of "six".

g. Relatively long stops occur preceding "two", "point", and "three", when these items occur in the middle of an utterance.

h. The vocabulary item "eight" is short and hard to pick out. For this case, attention is best paid to relative time counts of two or three different basic sound groups.

i. Fluid vowel sounds are sometimes very hard to distinguish, and often vary considerably from sample to sample.

A.7 ON UNIVERSAL AND NON-INITIAL MACHINES

For some speakers a given vocabulary item can apparently vary significantly, depending on whether it is the initial word in an utterance. If the difference between initial and non-initial voicings of a word are significant enough, then a recognition process which does not distinguish initial from non-initial will not work very well. The VDGS has some facility for dealing with this problem, at least to a limited extent.

When REVEXA is run, one has, for the first time, some indication of how well the transition letter sets are performing for the standard eleven vocabulary items. If the recognition of initial digits is noticeably bad for one or two items, the speech technician has the option of creating "universal" and "non-initial" machines for these items, with separate transition and loop letter sets. Concretely, this means that the following steps must be carried out (to be specific here, we assume that the vocabulary items "two" and "three" for Ulysses S. Grant require both non-initial and universal machines):

a. Rename ES$USG$02 as ES$USG11

b. Rename ES$USG$03 as ES$USG12

c. Run ESGDIT, entering "2" as vocabulary item, example space name ES$USG$11, and new example space name as ES$USG$02.

d. Run ESGDIT, entering "3" as vocabulary item, example space name ES$USG$12, and new example space name as ES$USG$03.

e. Rename MC02.TL as MC11.TL

f. Rename MC03.TL as MC12.TL

g. Rename MC02.LP as MC11.LP

h. Rename MC03.LP as MC12.LP

i. Delete TRLS02.TM, TRLS03.TM, TRIX02.TM, and TRIX03.TM

j. Run GZEC for the new example spaces ES$USG$11 and ES$USG$12

k. Run LOOPER for these new example spaces

l. Rerun REVEXA.

A.8 PROGRAM DESCRIPTIONS FOR VDGS ROUTINES

Program descriptions for VDGS routines follow:

1. EXTRACT

Title: EXTRACT.SV

Purpose:

The purpose of EXTRACT is fourfold:

a. Prompt the speaker to voice an utterance.

b. Save the TTI-500 features generated by that utterance on a disk file.

c. Compress the features for LCSR processing.

d. Provide hardcopy printouts of both raw feature data and compressed data.

Printout:

The TTI-500 detects 32 features every 2 msec. One of these features (LP_4) signals a long pause: the speech sample is complete. The software backs-up 50 TTI-500 samples (a set of 32 features), and continues going back through the samples. When feature 26 (UVNLC), 28 ($n_1 + n_3$) or 29 ($EG_1 + EG_2$) is found, the search terminates. This collection of features we call the "raw-data."

The data extraction program, at the user's option, will print this raw data in the standard space/asterisk format. The printout is consistent with TTI conventions: feature 1 (MAX_D1) is at the left, feature 32 (LP_4) on the right.

A letter is the subset of features 17-31. Associated with each letter is a count of the number of times that letter occurred in the raw data, interrupted by not more than one occurrence of any other letters. (Such single count letters are always ignored.) This collection of letters and counts we call the "compressed data."

The data extraction program reduces the raw data and prints the resulting compressed data.

User Dialog:

EXTRACT

(Ensure that any subdirectories to be used are initialized. It is recommended that each user utilize a personal subdirectory so that multiple copies of data files for the same digit string can be kept. An identifier and instructions appear:)

```
DATA EXTRACTION PROGRAM--ECLIPSE RDOS REV 6.23

STRIKE CNTRL-A TO EXIT FROM PROGRAM
```

(The user is requested to enter his name and an identifying comment of up to 80 characters for this run. This information, together with the date, is printed as a header on all printouts of the program.

Next, the user is queried to determine if he wishes to use a predefined prompting file. If so, he enters the file name. The program will verify that the name and file exist, but no other special checks are made.

Following queries to determine if the user wishes the printout of raw data and/or compressed data, the program prepares to accept speech data. If no prompting file is named, the user is commanded:)

SPEAK!!

(The TTI-500 is activated and the program "listens" until the LP_4 feature is detected. The TTI-500 is then de-activated and the user is requested to)

ENTER COMMENT LINE:

(Up to forty characters may be entered, then)

ENTER RAW DATA FILE NAME:
ENTER COMPRESSED DATA FILE NAME:

(If a file name which is already used is entered, the user is told of the condition and requested to redefine the file. A more serious error (e.g., directory not initialized) will cause an abnormal return to CLI.

The following convention is recommended for use in naming the raw data and compressed data files: five characters, dot, "RD" or "CD". Of the five characters, the first is the number set identifier (A-K) or "X" if no number set is used. The following four characters represent the number spoken, with N meaning "null," "P" meaning point. The .RD and .CD extensions refer to "Raw Data" and "Compressed Data."

Once the files are named, the program proceeds to print the compressed data and raw data (if the user requested it) and to write the data onto the disk files. The program then goes back to listening signaled by the SPEAK!! command. Note that unless spooling is disabled, the line printer may still be active (very noisy!) when SPEAK!! is offered. The user should be careful not to turn on the microphone until after the printing is complete.

If the user had named a prompting file, the request to SPEAK!! will be replaced by)

SAY: number

(Where the number is retrieved from the prompting file. When the TTI-500 "hears" something, assumed to be the prompted string, the notification)

OK

(is given. The file names are automatically retrieved from the prompting file (the convention noted above is used) and the printout and disk writing is performed. If the files already exist, the user is requested to intervene and name files on-line for the data. The user should note these problems and resolve them following the data extraction session. When the prompting file is exhausted, the warning)

NO MORE PROMPTS IN file name

(is given and the user is informed that the program will)

GO BACK TO START!

Input File:

MNSET-, the number set file

Output Files:

-.CD, the compressed data files.
-.RD, the raw data files.

Error Messages:

Only the standard RDOS file error messages are applicable to EXTRACT.

2. ESG

Title: ESG.SV

Purpose:

ESG builds an example space for a specified vocabulary item from specified compressed data files. Inputs to ESG include a list of the compressed data files which contain this item, and the canonical set of length and stretch factors. ESG determines what portion of the utterance is most likely to contain that item, and writes the file name and starting and ending records to the example space file.

Printout:

ESG provides a printout which describes the example space file entries. A printout of the automatically selected portion of the utterance is also provided under certain conditions when ESG determines that hand marking of the data may be required. This occurs whenever the selected portion of the utterance is too long for GENRLIZ to accommodate, and when a doublet occurs. If modifications are required, the example space file can be edited using ESDIT.

User Dialog:

```
ESG

ENTER THE EXAMPLE SPACE FILE NAME
(THE FIRST 2 CHARACTERS MUST BE 'ES'):
```

(This is the designated file name of the example space file which is to be generated.)

```
FILE            ALREADY EXISTS.
MAY I DELETE IT (Y/N)?
```

(If the example space file already exists, the user can choose to continue by deleting the existing file or terminate.
If he chose to terminate, the CRT displays)

```
    STOP- EXAMPLE SPACE FILE ALREADY EXISTS.
```

(Otherwise the dialog continues)

```
ENTER A BRIEF DESCRIPTION OF THE FILE:

ENTER THE 2-DIGIT VOCABULARY ITEM # (00-10):

ENTER THE PROMPTING FILE NAME:
```

(The prompting file contains the file names of the compressed data file names used in generating the example space file.)

```
ENTER THE 3-LETTER SUBDIRECTORY NAME:

COMPLETED PROCESSING VOCABULARY ITEM #:
USING PROMPTING FILE:       AND SUBDIRECTORY
DO YOU WISH TO CONTINUE PROCESSING ON THIS VOCABULARY ITEM (Y/N)?
```

(If the user wishes to continue processing, the program requests another input of a prompting file name and a subdirectory name. The program continues building the example space file on the same vocabulary item using the newly specified prompting file and subdirectory. Otherwise the program terminates.)

```
STOP
```

Input Files:

WIZ.ST, the file of length and stretch factors
Prompting file (a user-supplied filename - e.g. PFILE) with the file names of the compressed data files to be used.
Specified compressed data files

Output Files:

Example space file for the specified vocabulary item.

Error Messages:

```
INVALID VOCABULARY ITEM # ENTRY
```

(Another input is requested.)

```
STOP-FILE WIZ.ST DOES NOT EXIST
```

(The program terminates without this input file.)

```
CKST--FILE DOES NOT EXIST:
```

(If the prompting file does not exist, the program terminates. If a specified compressed data file does not exist, the program continues with the next specified compressed data file.)

```
CKST---UNKNOWN ERROR:                FILE:
```

3. GZEC

Title: GZEC.SV

Purpose:

G7EC finds the set of transition letter sets for a specified vocabulary item, using GENRLIZ.

Printout:

GZEC provides three types of printout. The first describes the development of the set of transition letter sets. In the compressed data printout, the header shows the feature number. Below this, in the "FREQ" column, the letter itself, delimited by the ";" symbol, is printed. The features set in a letter are shown by the symbol "*", blanks indicate the feature was not set. In the transition letter set printout, the "*" means the feature must be set, a blank means the feature must not be set, "?" indicates indifference, and "N" shows where a modification to accommodate this utterance occurred. The mapping of the transition letter sets into the utterance is shown by printing the particular set next to the first letter in the utterance which is contained in that set which occurs after a letter in the previous set. The "NUMBER OF TRANSITION LETTER SET" column shows the relation of the current sets to the initial or seed set of transition letter sets.

The second printout shows the cost to modify the transition letter sets, the current mean cost and standard deviation for each example encountered. The plot of these values is useful for detecting bad examples. The rescue index is used to retrieve any particular set of transition letter sets for further use.

The third printout shows the cost to modify particular sets of transition letter sets ("MACHINE NUMBER" on the printout) to accommodate a particular example. A cost of 0.0 indicates that no modification was required. A "*" marks the birth of a new machine, that is, it shows that previous machine was modified to accommodute the example.

User Dialog:

```
GZEC

ENTER NAME OF EXAMPLE SPACE FILE:

ENTER NAME OF SUBDIRECTORY WHERE
 TEMPORARY FILES ARE TO RESIDE:

ENTER TWO DIGIT VOCABULARY ITEM NUMBER:

(If temporary files already exist, the system queries)
```

```
FILES ALREADY EXIST:  TR**--.TM
 MAY I DELETE THEM? (Y OR N):
```

(A "N" response causes system to STOP. Rename .TM files before resuming processing.)

```
ENTER LISTING FILE
 (P = > $LPT, D = > DISK):
```

(If the "D" option is selected, the listing file name is constructed from the example space file name with the .LS extension. If this listing file already exists, the system queries)

```
MAY I DELETE filename?  (Y OR N):
```

(A "N" response causes the system to STOP. Rename .LS file before resuming processing)

```
PRESENT VALUE OF SDCOEF IS XX.X
 DO YOU WANT TO CHANGE SDCOEF?  (Y OR N):
```

(This is the value which controls modification of the transition letter sets. The modification is allowed if the cost is ≤ the mean cost + SDCOEF standard deviations. If "Y" is entered, the system responds.)

```
ENTER SDCOEF:
```

```
DO YOU WANT TO USE THE COST WEIGHT FACTORS?  (Y OR N)
 (IF NOT, ALL WEIGHTS ARE 1.0.  ENTER 'N' FOR HAND MARKED DATA):
```

(Weighting factors are used to reduce the contribution of extraneous end data to the final set of transition letter sets.

```
IS THERE AN EXISTING MACHINE
 WHICH IS TO BE GENERALIZED?  (Y OR N):
```

(This option allows an existing machine to be generalized to accommodate new examples. A "Y" response causes the system to prompt:)

```
ENTER FILE NAME (SUBDIR:NAME):
```

(If the "N" response was given to the former question, the system begins the search for a good starting point in the data. If the value of SDCOEF is sufficiently small, etc., the first pass through the example space may not yield a starting set of transition letter sets which satisfy the conditions. In this case the system notifies the user with)

```
PAUSE NO INITIAL MACHINE FOUND.  SHALL I TRY AGAIN?
```

(Strike any key to continue.)

```
STOP, ALL DONE!
```

Input Files:

ES -, the example space file
-.CD, the compressed data files
MC**.TL, optional set of transition letter sets which is to be generalized

Output Files:

$LPT or ES-.LS, the listing file
TRLS**.TM, all intermediate sets of transition letter sets
TRIX**.TM, index file into TRLS**.TM
COSTF.TM, temporary file of costs, deleted after graph printouts
FNFF.TM, temporary communication file between GZEC and PRNT7.

Error Conditions:

```
***WARNING:  WORD TOO LONG***
  FILE: filename, NLETR: XX
```

(The system protects itself against overfilling its arrays by verifying that the utterance to be processed is not too long. The examples which are too long must be edited using ESDIT. Processing continues to the next file.)

```
CKST -- FILE DOES NOT EXIST:  filename
```

(If a file is given in the example space which cannot be found at processing time, the error is noted on the printer and processing continues.)

```
CKST -- UNKNOWN ERROR:  XX          FILE:  filename
```

(This error indicates that although the file was found, it cannot be accessed for some reason which CKST is unable to remedy. Refer to the RDOS manual for a description of error codes and file status codes. Again this error is noted on the printer and processing continues.)

```
GWRD -- UNKNOWN ERROR:  XX          FILE:  filename
```

(If GWRD is unable to read the compressed data file, it prints this error and takes the error return. The existence of the file is not in question when this error is detected, but rather some other file data error has occurred. The most likely cause is an illegal starting or ending record specified for the compressed data file resulting from an error introduced in editing the example space.)

4. RESCUE

Title: RESCUE.SV

Purpose:

RESCUE retrieves a desired set of transition letter sets from a temporary file and writes it into a machine file.

Since the final set of transition letter sets for a vocabulary item may not be the best one due to the inclusion of bad examples, all the unique transition letter sets and the information to access them are saved in temporary files.

RESCUE prints the desired set of transition letter sets if requested. RESCUE deletes the temporary files if requested.

Printout:

A printout of the set of transition letter sets is provided.

User Dialog:

RESCUE

ENTER THE 3-LETTER SUBDIRECTORY NAME:

ENTER THE 2-DIGIT VOCABULARY ITEM # (00-29):

ENTER THE RESCUE INDEX:

(This is the desired set of transition letter sets as determined from the cost graph produced by GZEC.)

IS THE MACHINE TO BE PRINTED (Y/N)?

(If requested, the desired set of transition letter sets is printed in addition to being written into the machine file.)

CREATED FILE

(The machine file name is displayed.)

ARE THE FILES AND
TO BE DELETED (Y/N)?

(The user is queried whether the temporary files are to be deleted.

The temporary file TRLS**.TM saves all unique sets of transition letter sets for vocabulary item**.

The temporary file TRIX**.TM contains the starting record numbers of the transition letter sets in TRLS**.TM and the number of transition letter sets in each set for vocabulary item**.

After the temporary files are deleted, if so requested, the following message appears.)

DELETED FILES AND

STOP PROCESSING COMPLETE

Input Files:

TRLS**.TM, the temporary file of sets of transition letter sets for vocabulary item**.

TRIX**.TM, the temporary file of starting record numbers and the number of transition letter sets for the sets of transition letter sets stored in TRLS**.TM for vocabulary item**.

Output Files:

MC**.TL, the machine file of transition letter sets for vocabulary item**.

Error Messages:

CKST - FILE DOES NOT EXIST:

(If any of the input files do not exist for the vocabulary item, this message is output and the program terminates.)

CKST -- UNKNOWN ERROR: FILE:

FILE ALREADY EXISTS:

(If the machine file already exists for the vocabulary item, the program terminates.)

STOP ON ERROR

(The program terminates for any of the above errors.)

STOP - NO SUCH MACHINE

(The specified machine number does not exist. The program terminates.)

5. SIGH

Title: SIGH.SV

Purpose:

SIGH checks the transition letter set files MC**.TL created by RESCUE to determine if the number of transition letters in each MC**.TL file is less than thirteen. If the number is less than thirteen, nothing is done; if this number is greater than thirteen, the transition letters with the greatest number of "?" features are deleted until the remaining number of transition letters is smaller than thirteen.

Printout:

None

User Dialog:

SIGH

ENTER 2-DIGIT STARTING MACHINE NUMBER

ENTER 2-DIGIT END MACHINE NUMBER

(SIGH checks each transition letter set in order. If a transition letter set need not be reduced, the message)

TRANSITION LETTER SET FOR THIS ITEM OK

(appears on the CRT. If a transition letter set is reduced, the message)

TRANSITION LETTER SET FOR THIS ITEM REDUCED

(appears.)

Input Files:

MC**.TL,	transition letter set for item**

Output Files:

MC**.TL,	reduced or unmodified transition letter set for item**
MC**.XY,	non-reduced transition letter set for item**

Error Messages:

INVALID ENTRY - illegal machine number entered

FILE OPEN ERROR - could not open MC**.TL file.

6. LOOPER

Title: LOOPER.SV

Purpose:

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

Printout:

In LOOPER each example from an example space is printed and next to it the sets of transition and loop letter sets. The loop letter set printout is identical in format to the transition letter set format, except .hat the words "EMPTY SET" appear to describe this condition (impossible in transition letter sets). The letter sets are identified in the far right- hand column. "T1" means transition letter set 1, "L2" means loop letter set 2, and so on. In some cases, empty loop letter sets are not shown because the transition letter set printout takes precedence.

If the example has more than one start point, this printout is repeated for the subsequent cases.

The final set of loop letter sets which accommodates at least one start point in each utterance in the example space is also printed.

User Dialog:

LOOPER

ENTER NAME OF EXAMPLE SPACE FILE:

ENTER 2-DIGIT VOCABULARY ITEM (00-10):

ENTER NAME OF SUBDIRECTORY
WHERE SET OF LOOP LETTER SETS IS TO RESIDE
(MUST BE 3 CHARACTERS):

ENTER DESCRIPTION OF THIS RUN:

(The description entered here is printed in the header of the LOOPER listing.)

STOP LOOPER IS FINISHED

Input Files:

ES-, the example space file
MC**.TL, the set of transition letter sets for this vocabulary item
-.CD, the compressed data files specified by the example space.

Output Files:

MC**.LP, the set of loop letter sets for this vocabulary item
LP**.TM, a temporary file of sets of loop letter sets for the different starting points in the examples. This file is deleted after processing is complete.

Error Conditions:

LOOP LETTER SETS ALREADY EXIST FILE: filename

(This fatal error results when the specified loop letter sets already exist. Delete or rename the specified file before restarting LOOPER.)

STOP LOOPER MUST HAVE A SET OF TRANSITION LETTER SETS

(Loop letter sets cannot be generated without the corresponding transition letter sets (MC**.TL).)

Other file data error conditions are identical to the CKST and GWRD errors described in the GZEC discussion.

7. REVEXA

Title: REVEXA.SV

Purpose:

REVEXA is version A of the revised research machine exerciser. Essentially, it is a stripped-down version of REVEX whose purpose is to collect counter data. In contrast to REVEX, however, REVEXA does not allow any utterance with a transition or loop letter set violation to proceed to recognition. For a more complete description of the operation of REVEXA, see the program description for REVEX.

Printout:

This is the same as in REVEX.

User Dialog:

This is the same as in REVEX. Here, to speed execution, the question:

DO YOU WANT TO USE ONLY THE MACHINES IN THE UTTERANCE? (Y/N)

should be answered "Y", since the other machines would only contribute artifacts, and, at this point, data about artifacts are not used.

Input Files:

MNSET* - the magic number sets to be used by REVEXA

-.CD - compressed data files

MC**.TL - transition letter set for item **

MC**.LP - loop letter set for item **

8. RVDIT

Title: RVDIT.SV

Purpose:

RVDIT is the counter data file editor. RVDIT creates from the counter data file for a speaker (<SUB>: CDAT.RV) counter data files for each of the machine numbers (<SUB>: CDAT**.RV where ** is the machine number).

The user can specify which counter data records are to be flagged as "real" in the new files to be created by inputting the file <SUB>: RVCARDS.

Printout: None

User Dialog:

RVDIT

ENTER THE 3-LETTER SUBDIRECTORY NAME:

IS THERE A MACHINE 12 (Y/N)?

ARE THERE ANY COUNTER RECORDS TO BE MODIFIED (Y/N)?

(If the user does not wish to keep all of the counter data records, then file RVCARDS must exist.)

```
WARNING -- FILES  subdirectory:CDAT**.RV WILL BE DELETED
  FOR ALL MACHINE NUMBERS** (00-11).
  DO YOU WISH TO CONTINUE (Y/N)?
```

(If specified, the program terminates with STOP PROCESSING.

Otherwise, the specified files are deleted and the program continues.)

STOP ALL DONE

Input Files:

CDAT.RV the counter data file

CIDX.RV the counter index file

RVCARDS contains the record numbers of the counter data records to be flagged as "real" in the new files to be created.

Output Files:

CDAT**.RV the counter data files for machine number **.

Error Messages:

CKST--FILE DOES NOT EXIST:

(If any of the input files do not exist, this message is output and the program terminates.)

CKST--UNKNOWN ERROR:

STOP ON ERROR

(The program terminates for any of the above errors.)

9. COVERT

Title: COVERT.SV

Purpose:

COVERT computes the covariance matrix, median, delta lower, delta upper of the counters for each specified machine. It also calculates the coefficients of correlation for the non-diagonal upper triangular elements of the covariance matrix.

Printout:

For each specified machine, COVERT prints for each counter position the selected counter equation, the ordered C-values, median, delta lower, and delta upper. It prints the calculated covariance matrix and the coefficients of correlation for the covariance matrix.

User Dialog:

COVERT

ENTER THE 3-LETTER SUBDIRECTORY NAME:

ENTER THE 2-DIGIT STARTING MACHINE NUMBER (00-15):

ENTER THE 2-DIGIT END MACHINE NUMBER (00-15):

(COVERT creates the covariance matrix files for the machines in ascending order, beginning with the starting machine number and finishing with the end machine number.

The current limits on the machine numbers are 0-15).

CREATED FILE:

(The covariance matrix file name for the machine is displayed).

STOP-ALL DONE FOLKS!

Input Files:

CDAT**.RV, the counter data file for machine **

Output Files:

CM**, the covariance matrix file for machine **.
COV.ST, the counter data statistics file. A record of counter data statistics (medians, delta uppers, delta lowers, equation flags) is written for each machine processed by COVERT.

Error Messages:

INVALID ENTRY

(Invalid starting or end machine numbers were entered. Another input is requested.)

CKST--FILE DOES NOT EXIST:

(If the input file does not exist for the machine being processed, this message is output and the processing is skipped for this machine.)

CKST -- UNKNOWN ERROR: FILE:

FILE ALREADY EXISTS:

(If the covariance matrix file already exists for the machine being processed, this message is output and the processing is skipped for this machine).

10. INVERT

Title: INVERT.SV

Purpose:

INVERT calculates the inverted covariance matrix for each specified machine. It also computes the eigenvalues of the covariance matrix for each specified machine if so desired.

Printout:

INVERT prints the eigenvalues of the covariance matrix when the option to compute the eigenvalues is chosen.

User Dialog:

```
INVERT

ENTER THE 2-DIGIT STARTING MACHINE NUMBER (00-29):

ENTER THE 2-DIGIT ENDING MACHINE NUMBER (00-29):

PRINT EIGENVALUES OF COVARIANCE MATRIX (1=YES, 0=NO):
```

(INVERT creates the inverted covariance matrix files for the machines in ascending order, beginning with the starting machine number and finishing with the end machine number.)

The current limits on the machine numbers are 0-29.

Input Files:

CM**, the covariance matrix file for machine **.

Output Files:

INCM**, the inverted matrix file for machine **.

Error Messages:

INVALID ENTRY

(Invalid starting or end machine numbers were entered. Another input is requested.)

FILE OPEN ERROR

(The covariance matrix file cannot be opened.)

11. CROAK

Title: CROAK.SV

Purpose:

CROAK calculates the delta and mu values for each counter data record of a machine number and then orders and prints the delta and mu values. CROAK also creates the file RVX.ST of statistics used by REVEX.

Printout:

For each vocabulary item, CROAK prints out the inverted covariance matrix for that item, its determinant, and the delta and mu values in unordered and in sorted form with their computed mean and standard deviation. CROAK also plots the cumulative distributions of the delta and mu values.

User Dialog:

CROAK

ENTER THE 3-LETTER SUBDIRECTORY NAME:

(Then the program requests the user to enter the data extraction mode:)

ENTER THE DATA STATISTICS EXTRACTION MODE

1 (Mode 1 - REAL RECOGNITIONS WITH VIOLATIONS)

2 (Mode 2 - ALL REAL RECOGNITIONS)

3 (Mode 3 - REAL RECOGNITIONS WITHOUT VIOLATIONS)

4 (Mode 4 - ARTIFACTS ONLY)

(If the user enters anything but 1, 2, 3, or 4, the message "INVALID ENTRY" appears, and the user is asked again for a mode number.)

(If any mode but 4 is chosen, the user is asked if he wishes to save the CROAK-generated statistics:)

DO YOU WISH TO SAVE THE PROBABILITY STATISTICS? (Y/N)

(Then the program requests starting and ending machine numbers:)

"ENTER 2-DIGIT STARTING MACHINE # (00-15)"

"ENTER 2-DIGIT END MACHINE # (00-15)"

(If an illegal entry is made, the message "INVALID ENTRY" appears and the user is asked again for starting and end machine numbers.)

When the delta values have been computed and sorted, the message

DELTA VALUES ARE STORED IN FILE______appears, and when the mu values have been computed and sorted, the message

MU VALUES ARE STORED IN FILE_______appears.

The message

STOP- ALL DONE FOLKS!

appears when the processing is complete.

Input Files:

INCM**	-	inverted covariance matrix file for item **
CDAT**.RV	-	counter data file for item **
CIDX.RV	-	index file for the counter data file
COV.ST	-	counter data statistics file

Output Files:

QDAT**.RR	-	file of delta values for reals
QDAT**.AF	-	file of delta values for artifacts
MUDT**.RR	-	file of mu values for reals
MUDT**.AF	-	file of mu values for artifacts
RVX.ST	-	statistics file for REVEX

Error Messages:

DELTA FILE FOR THIS ITEM ALREADY EXISTS:

MU FILE ALREADY EXISTS FOR THIS ITEM

(Either QDAT** or MUDT** already exists and should be deleted or renamed before invoking CROAK again.)

PROBLEM CREATING ________

CROAK was not able to create the named file.)

CKST -- FILE DOES NOT EXIST

CKST -- UNKNOWN ERROR

12. REVEX

Title: REVEX.SV

Purpose:

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

Printout:

REVEX provides three types of printout. The first provides a detailed history of the progress of each copy of each active machine. It is read as follows. The machine number is shown in the heading. Machine 0 is that constructed for the word zero. Machine 10 is that which recognizes "point" and is shown as "P" in printout 3. When two separate machines exist for a single vocabulary item, their histories are combined under the one column for that machine. The initial or universal machine start is marked by a "Z", while the n n-initial version starts are marked with the typical "S".

The stage of each machine is shown for each letter in the utterance (the letter number appears on the left and can be correlated with the utterance printout given in printout 3). The stage is described by a single number or letter, as shown in Table A2. (If the numerical stage exceeds 9, only the units digit is printed.) Thus the progress of a copy of a machine can be traced by simply following the specified print column. Note that when the number of copies of a machine exceed the space available, data for copies in the next print column is shifted to the right to allow data for all copies to be printed.

Special symbols are used in addition to the stage descriptors. Their meaning is shown in Table A3.

Finally, a line appears at the end indicating which machine copies in the final stage were forced to recognition at the end of the utterance.

Printout 2 lists relevant data about the recognitions, including the loop letter violations. The "start order" values are used to correlate these data with printout 3. The universal machine descriptors are user selectable. Her "2U" distinguishes that machine from the non-initial version ("2").

Printout 3 gives the utterance and maps the recognitions onto it in order of start time. The characters refer to the machine, as discussed for printout 1. The "-" symbol marks the time the machine spent in its last stage.

TABLE A2. Stage and State Descriptors Used in REVEX Type 1 Printout

Stage	State	
	Transition	Loop
1	1	A
2	2	B
.	.	.
.	.	.
.	.	.
10	0	J
11	1	K
.	.	.
.	.	.
.	.	.

TABLE A3. Meaning of Special Symbols in REVEX Type 1 printout

Symbol Category	Symbol	Meaning
Machine start	S	The violation-free start of a particular copy of a machine. Not used for universal machines. Always stage 1.
	Z	The violation-free start of a particular copy of the universal version of a machine. Always stage 1.
	$	Machine copy start on a transition letter violation. Can occur in stage 1 only for the first letter of the utterance. Thereafter, the stage is one greater than parent's stage.
Parent copy	@	Marks the parent copy of the "$" to the left of this copy. Indicates parent is in T state.
	:	Parent in L state.
	=	Parent in the L state with an acceptable violation this letter.
	/	Parent dropped due to excessive loop violations.

TABLE A3. Meaning of Special Symbols in REVEX Type 1 printout (Cont)

Symbol Category	Symbol	Meaning
Violations	$	Transition letter violation within acceptable limits such that a new copy (the offspring) was started.
	-	L state, letter not in L (i.e., an L violation).
Dropped copies	X	Copy dropped due to excessive L violations.
	/	Copy dropped due to excessive L violations after having sired an offspring.
	+	Copy not selected for advancement to the next stage because a) a better copy was advanced; or b) a copy in the next stage was better than this copy.
	\	Copy dropped because a copy advancing to or created in this stage is a better copy.
	*	Copy dropped after recognition.
Final Stage	?	Copy in final stage delay, awaiting recognition.
Recognition	*	Recognition

User Dialog:

```
REVEX

ENTER NAME OF SUBDIRECTORY
 WHERE DATA FILES RESIDE
 (MUST BE 3 CHARACTERS):

ENTER MODE OF DATA ACQUISITION
 (MODE 1 - ESG FORMAT FILE
  MODE 2 - LIVE UTTERANCE
  MODE 3 - MAGIC NUMBER SET
  MODE 4 - INDIVIDUAL -.CD FILES):
```

(The live utterance option will be implemented in a subsequent phase. If mode 1 is selected, the system responds.)

```
ENTER NAME OF EXAMPLE SPACE:
```

(If mode 3 is selected, the system requests,)

```
ENTER NAME OF MAGIC NUMBER SET:
```

(The system searches for and reads the machines, and explains the brief pause to the user)

```
READING INDIVIDUAL MACHINES . . .
```

(User dialog continues after machines are found)

```
EXTRA PRINTOUT TO $LPT? (Y OR N):
```

(This option allows MINIMINT printout to be directed to a disk file RV.LS if "N" is entered. If this listing file already exists, the system asks)

```
LISTING FILE EXISTS
 MAY I DELETE IT? (Y OR N):
```

(The "N" response causes the system to open the existing listing file for appending.)

```
DO YOU WANT THE LONG STATE PRINTOUT? (Y OR N):
```

(A "N" response causes the printout described as type 1 to be suppressed.)

```
DO YOU WANT THE MACHINE OVERLAP PRINTOUT? (Y OR N):
```

(A "N" response causes the type 3 printout to be suppressed.)

```
FILES EXIST, FILES: SUB:CIDX.RV SUB:CDAT.RV
 MAY I DELETE THEM? (Y OR N):
```

(This message is output if counter data files already exist on the specified subdirectory. A "N" response causes the system to append the new data to the existing files, and the message is output,)

```
APPENDING TO EXISTING FILES
```

(When a machine is found for a vocabulary item above 10, the system asks,)

```
MACHINE ** FOUND.  IT IS ASSUMED TO BE AN INITIAL MACHINE.
 ENTER VOCABULARY ITEM TO WHICH IT CORRESPONDS (0-10):
```

(The system then requests a descriptor for use in type 2 and 3 printouts,)

```
ENTER 2 CHARACTER DESCRIPTOR FOR MACHINE
 (E.G. '2U'):
```

(Finally, the starting time for the non-initial version is requested,)

```
ENTER TIME ** SHOULD STOP AND XX BEGIN:
```

(The system offers the option of activating only those machines which are actually in the utterance. To use all machines, enter "N")

```
DO YOU WANT TO USE ONLY THE MACHINES IN THE UTTERANCE? (Y OR N):
```

(A pause for system initialization follows, and then the system requests,)

```
ENTER DESCRIPTION OF THIS RUN:
```

(The user may enter a 40 letter descriptive string which is printed in the header.)

```
ENTER NAME OF UTTERANCE FILE
 (OR '*' TO TERMINATE):
```

(This request is made for every utterance if mode 4 was selected above. Otherwise no further user inputs are required.)

```
STOP REVEX IS FINISHED
```

Input Files:

MC**.TL, the sets of transition letter sets for the vocabulary items of interest.
MC**.LP, the sets of loop letter sets
-.CD, the compressed data files.
ES- or MNSET*, example space or number set files if either form of entry was selected.
COV.ST, counter data statistics
INCM**, inverted covariance matrix files
RVX.ST, statistics file created by CROAK

Output Files:

CIDX.RV, the index file into CDAT.RV in which the data for each utterance are kept.
CDAT.RV, the set of counter data, start and end times and loop violations for each machine which goes to recognition.

Error Messages:

```
ILLEGAL MODE

ENTER MODE......
```

(This message is printed when the mode is not in the range 1 $\leq$ mode $\leq$ 4.)

```
****WARNING:  -.RV FILES ARE NOT COMPATIBLE
CURRENT BYTES:  XX, OLD BYTES:  XX
STOP
```

(This message occurs when an attempt is made to append to counter data files created under a different revision of REVEX. The differing record sizes make it impossible to append. The old counter files must be deleted or renamed.)

```
Filename DOES NOT EXIST
```

(This message appears when a particular machine is not found. REVEX assumes this machine is not to be used and continues. REVEX can operate with 1 to 13 machines in its present configuration.)

```
NO SPACE IS AVAILABLE TO INSERT A MACHINE COPY FOR
 VOCABULARY ITEM #:  XX
```

(This message is output to the printer when no space is available in the machine copy data array. The size of this array must be changed to accommodate extra copies if this error is encountered.)

CKST and GWRD file data errors are the same as those described for GZEC.

13. ADDER

Title: ADDER.SV

Purpose:

ADDER lists the transition and loop letter set violations by vocabulary item for each utterance which has been processed by REVEX. This violation data is stored in two tables: one for real recognitions and one for artifacts.

Printout:

If desired, ADDER prints violation data for each utterance processed as well as the two violation tables.

User Dialog:

ADDER

(The program then responds:)

PROGRAM ADDER

DO YOU WANT TO PLACE OUTPUT ON A DISK FILE? (Y/N)

(If the answer is yes, the program responds:)

ENTER DESIRED FILENAME (16 CHAR MAX)

(If a bad filename is chosen, the program prompts:)

SOMETHING IS WRONG WITH YOUR CHOICE OF FILENAME.

CHOOSE ANOTHER.

Then,

ENTER RELEVANT COMMENT (40 CHAR MAX)

ENTER DISK CONTAINING CDAT, CIDX DATA FILES (3 CHAR)

(e.g., enter DP2)

ENTER SUBDIRECTORY LOCATION OF CDAT, CIDX DATA FILES (3 CHAR)

DO YOU WANT THE LONG FORM PRINTOUT? (Y/N)

(ADDER then proceeds to process utterances one by one, storing violation data in the two violation matrices.)

Input Files:

CDAT.RV counter data file from REVEX

CIDX.RV index file for the counter data file

Output Files:

LTVF.MM file containing violation table user by DEALER

Error Messages:

STOP PROBLEM WITH LTVF.MM

(There was a problem opening the file LTVF.MM)

KARMA -- FILE DOES NOT EXIST

(Either CDAT.RV or CIDX.RV does not exist)

KARMA -- UNKNOWN ERROR

(Problem with status of CDAT.RV or CIDX.RV files)

14. AVRAJ

Title: AVRAJ.SV

Purpose:

AVRAJ computes average word length for real recognitions.

Printout:

AVRAJ prints out the average word length for each machine.

User Dialog:

AVRAJ

(The program responds:)

PROGRAM AVRAJ

ENTER DISK CONTAINING CDAT, CIDX DATA FILES (3 CHAR)

ENTER SUBDIRECTORY LOCATION OF CDAT, CIDX DATA FILES (3 CHAR)

(The program then proceeds to run through the CDAT.RV file, calculating average word length for each vocabulary item over all real recognitions of that item. When this process is complete, the message below appears.)

AVERAGE WORD LENGTHS ALSO EXIST ON BINARY FILE: AVRWRD.ST

STOP AVRAJ IS FINISHED.

Input Files:

CDAT.RV Counter data file created by REVEX

CIDX.RV Index file to CDAT.RV

Output Files:

AVRWRD.ST

File of average word lengths

Error Messages:

KARMA -- FILE DOES NOT EXIST

(Either CDAT.RV or CIDX.RV file does not exist on the specified subdirectory.)

KARMA -- UNKNOWN ERROR

(Problem with status of CDAT.RV or CIDX.RV files.)

STOP PROBLEM OPENING AVRWRD.ST

(There was a problem opening AVRWRD.ST.)

15. CRAP

Title: CRAP.SV

Purpose:

CRAP determines the critical word association factors for each vocabulary item, going through each utterance processed by REVEX to check associations occurring in the utterance at nine levels of overlap.

Printout:

For each of the nine levels of overlap, CRAP prints out a matrix of overlap parameters. Also, the critical association parameters for each vocabulary item are printed.

User Dialog:

```
CRAP
```

(And the program responds:)

```
PROGRAM CRAP

DO YOU WANT TO PLACE OUTPUT ON A DISK FILE?  (Y/N)
```

(If the user answers affirmatively, the program requests a filename. The disk file, if chosen, receives all output that would otherwise go to the printers.)

(Then the program asks for the location of the CDAT.RV and CIDX.RV files:)

```
ENTER DISK CONTAINING CDAT, CIDX DATA FILES (3 CHAR):
```

(e.g., enter "DP2")

```
ENTER SUBDIRECTORY LOCATION OF CDAT, CIDX DATA FILES (3 CHAR)
```

(The program then asks for the machine types of machines 11 and 12:)

```
ENTER VOCABULARY TYPE FOR MACHINE 11:  (I.E. '2' or '4')

ENTER VOCABULARY TYPE FOR MACHINE 12:  (I.E. '2' or '4')
```

(An entry of - 1 should be made if the machine is not being used.)

(Then, at the end:)

```
CRITICAL ASSOCIATION PARAMETERS HAVE BEEN OUTPUT TO CAP.ST

STOP CRAP FINISHED
```

Input Files:

AVRWRD.ST - file of average word lengths
CDAT.RV - counter data file
CIDX.RV - index file for the counter data file

Output files:

CONGAP - file for contiguous real gap matrix
CAP.ST - file of critical association parameters

Error Messages:

```
STOP PROBLEM OPENING OUTPUT FILE - for disk file output
STOP PROBLEM OPENING CONGAP
STOP PROBLEM OPENING AVERAGE WORD FILE "AVRWRD.ST"

SOMETHING IS WRONG WITH YOUR CHOICE OF FILENAME.  CHOOSE ANOTHER
```

If a disk file output is chosen and the filename given is not acceptable, another name is asked for.

16. GAPSTER

Title: GAPSTER.SV

Purpose:

GAPSTER is responsible for determining the association, gap and delay values; GAPSTER creates the gap matrix GAP.DT and the QASM matrix needed in the MIND file.

Printout:

GAPSTER prints out time gap statistics for real and artifact recognitions as well as the gap matrix and the QASM matrix.

User Dialog:

GAPSTER

PROGRAM GAPSTER

DO YOU WANT TO PLACE OUTPUT ON A DISK FILE? (Y/N)

(If the user answers affirmatively, the program requests a filename. If the filename chosen is bad, the program asks for another name.)

(Then, just as in CRAP, the program asks for the location of the CDAT.RV and CIDX.RV data files.

If these data files are found, the machine types of machines 11 and 12 are requested.)

ENTER VOCABULARY ITEM FOR MACHINE 11

ENTER VOCABULARY ITEM FOR MACHINE 12

(If a machine is not being used, enter "-1"

The program then asks for the critical association factor:)

ENTER CRITICAL ASSOCIATION FACTOR

(The recommended response here is "1.0")

(At this point, the user is asked to enter the total number of machines to be used.)

ENTER TOTAL NUMBER OF MACHINES TO BE USED

(For example, if machines 0-10 were being used, the response would be "11")

(After running a few minutes, the program halts and requests:)

ENTER REAL STD.DEV.SPREAD FACTOR FOR GAP MATRIX

(The recommended reply here is "1.0")

(A little bit later the user is asked:)

DO YOU WANT THE QUARTILE AND MEDIAN CALUCLATIONS? (Y/N)

(These calculations are not required and may be omitted.)

(Then, at the end:)

STOP GAPSTER IS FINISHED

Input Files:

CONGAP - file of contiguous real gap matrix

Output Files:

GAPMAX - file holding maximum gap value

GAP.DT - file holding gap matrix

QASM.DT - file holding QASM matrix

Error Messages:

STOP PROBLEM OPENING OUTPUT FILE - for disk listing file

STOP PROBLEM WITH OPENING CONGAP

STOP PROBLEM OPENING GAPMAX

STOP PROBLEM OPENING CAP.ST

FILE DOES NOT EXIST - the comprised data file in question does not exist

STOP PROBLEM OPENING QASTMP - cannot open QASM.DT

17/18. SORTRA, SORTRB

Title: SORTRA.SV, SORTRB.SV

Purpose:

SORTRA/SORTRB sorts the file GAPMAX/CONGAP, putting the entries in ascending numerical order.

Printout:

SORTRA prints a plot of the ordered values of the file GAPMAX, and SORTRB does the same thing for the file CONGAP.

User Dialog:

SORTRA (SORTRB)

(SORTRA/SORTRB proceeds to sort the entries in the file GAPMAX/CONGAP and then plot the sorted values.

Input Files:

GAPMAX - for SORTRA

CONGAP - for SORTRB

Error Messages:

None

19. MUTE

Title: MUTE.SV

Purpose:

MUTE computes the L-counter parameters MDLA0, MDLA1, and MDLA2.

Printout:

MUTE prints out the MDLA0, MDLA1, and MDLA2 values for each machine, as well as the parameters P_o for reals and artifacts and for reals and artifacts.

User Dialog:

MUTE

ENTER 2-DIGIT STARTING MACHINE NUMBER

ENTER 2-DIGIT END MACHINE NUMBER

(MUTE proceeds to compute the MDLA** values for each machine beginning with the starting number machine.)

Input Files:

MUDT**.RR - file of values for reals of item **

MUDT**.AF - file of values for artifacts of item **

Output Files:

LOOPY - file of MDLA** values for all machines

QLSTATS - file of P_o and values for all machines

Error Messages:

TOO MANY MU VALUES - the number of values exceeds 600

FILE OPEN ERROR - one of the MUDT** files cannot be opened

20. GLOVE

Title: GLOVE.SV

Purpose:

GLOVE is a least squares routine designed to fit a curve through the observed points of the cumulative distribution of the Q_T quality function values for both real and artifact recognitions.

Printout:

GLOVE prints out five coefficients for each real vocabulary item and five coefficients for each artifact vocabulary item. These coefficients are the "adjustable parameters" determined so that, with these values used as coefficients in the general functional form, the fit to a particular sec of data points is best in the sense of least squares.

User Dialog:

```
GLOVE

ENTER STARTING MACHINE NUMBER (0-29)

ENTER END MACHINE NUMBER (0-29)
```

(GLOVE then proceeds to compute coefficients for each vocabulary item, real and artifact, doing reals first in ascending order then artifacts in ascending order.)

Input Files:

QDAT**.RR - file of real delta values for item **

QDAT**.AF - file of artifact delta values for item **

Output Files:

QDFT**.RR - file of coefficients, median delta and range for reals

QDFT**.AF - file of coefficients and median delta for artifacts

APR - file holding number of real deltas for each vocabulary item

APA - file holding number of artifacts deltas for each vocabulary item

Error Messages:

FILE STATUS ERROR FOR ITEM: _______

TOO MANY DELTA VALUES FOR THIS ITEM

(The number of deltas exceeds available array size.)

21. TAILOR

Title: TAILOR.SV

Purpose:

TAILOR calculates T-counter quality function values. TAILOR reads coefficients and a range of values to fit from files QDAT**.RR and QDAT**.AF (where ** is the machine type), then fits a quadratic to the ratio of the real and artifact data. The coefficients of the fitted curve are then transformed into a MEX usable form and written into the file WHAT.

Printout:

The real coefficients from QDAT**.RR.

The range at delta values to be used.

The coefficients for artifacts.

The range of delta values actually used in the fit.

The determinant of the matrix used in the least squares fit.

The coefficients of the fitted curve.

A plot of the fitted curve and data points.

The MDTA* values.

User Dialog:

TAILOR [beginning machine number/B] [ending machine number/E]

If /B option is omitted, beginning machine number is assumed to be 0.

If /E option is omitted, ending machine number is assumed to be 10.

TAILOR will type the matrix generated by the least squares fit for each machine type processed.

Input Files:

QDAT**.RR, coefficients for real recognition

QDAT**.AF, coefficients for artifacts, where ** goes from 00 to 10

Output Files:

WHAT T-counter quality function values

Error Messages:

```
     STOP - OPEN ERROR - QDAT**.-

     STOP - READ ERROR - QDAT**.-
```

22. BUILDER

Title: BUILDER.SV

Purpose:

BUIILDER builds the machine data file MDFL.MM from the input files LOOPY and WHAT created by MUTE and TAILOR respectively.

Printout:

None

User Dialog:

BUILDER

(BUILDER then proceeds to read the files LOOPY and WHAT and then merge them to create MDFL.MM. When this is complete, the message

MDFL.MM CREATED

appears at the CRT.)

Input Files:

LOOPY, file of MDLA* values created by MUTE

WHAT, file of MDTA* values created by TAILOR

Output Files:

MDFL.MM, machine copy data file needed by DEALER

Error messages:

PROBLEM OPENING LOOPY - cannot open LOOPY

PROBLEM OPENING WHAT - cannot open WHAT

LOOPY AND WHAT INCOMPATIBLE - the number of entries in LOOPY is not the same as the number of entries in WHAT. This terminates the program.

23. DEALER

Title: DEALER.SV

Purpose:

DEALER pulls together the various files created by CRAP, GAPSTER, and BUILDER to create the file MIND.VD.

Printout:

None

User Dialog:

```
DEALER

PROGRAM DEALER

ENTER REVISION NUMBER FOR THIS JOB:
```

(The current revision number is "*0")

(The program then asks for number of vocabulary items, and the disk and subdirectory containing the data.)

```
ENTER NUMBER OF VOCABULARY ITEMS (0-13)

ENTER "DISK: SUBDIR":  (8 CHAR. MAX)
```

(For example, the user might enter "DP2:ABC")

If there are universal machines (machines 11 and 12) the program requests

```
ENTER VOCABULARY ID AND END TIME FOR MACHINE _____ SEPARATED BY COMMA
```

(So, for example, the user might enter "2,25" for machine 11, indicating that machine 11 is universal machine for vocabulary item 2, and that the end time for this item is 25.)

Finally, if no errors occur, the user is asked for comment to add to the data file.

Then

```
LOOKS LIKE WE MADE IT, FOLKS
```

appears, and the MIND file is complete, except for the play factors.

Input Files:

LVTF.MM - transition/loop letter set violation table file

MC**.TL - transition letter set files

MC**.LP - loop letter set files

COV.ST - covariance statistics file

INCM** - inverted covariance matrix files

RVX.ST - statistics file created by REVEX

MDF.MM - machine data file created by BUILDER

GAP.DT - gap matrix file

CAP.ST - critical association parameter file

Output File:

MIND.VD - incomplete MIND file

Error Messages:

ERROR NO. _______ OCCURRED IN STAT CALL FOR FILE _______

(The status of the named file is bad. The RDOS error code is used.)

STOP -- TOO MANY STAGES

(The number of transition letter sets is too large.)

STOP PROBLEM WITH COV.ST

STOP PROBLEM WITH RVX.ST

STOP PROBLEM WITH A INCM** FILE

STOP PROBLEM WITH MDFIL - MDFL.MM file is bad

STOP PROBLEM WITH LTVF

STOP PROBLEM WITH QASM.DT

STOP PROBLEM WITH GAP.DT

(Usually an error of this kind simply means that the named file does not exist on the subdirectory in question.)

"MIND" IS WARPED - status of output file is bad, enter another

GIVE ME A NEW FILENAME

24. PHEW

Title: PHEW.SV

Purpose:

PHEW writes the three play factors to the end of the MIND file.

Printout:

PHEW prints out the a priori costs for each machine

User Dialog:

PHEW

ENTER NUMBER OF VOCABULARY ITEMS

(PHEW opens the MIND.VD file for appending, computes a priori costs for each machine and writes these costs to the end of the MIND file together with the three gap matrix play factors. When this is accomplished the message

MIND HAS BEEN CREATED

appears on the CRT.)

Input Files:

MIND.VD, the data file created by DEALER

APR, the file holding number of real deltas for each machine

APA, the file holding number of artifact deltas for each machine

Output Files:

MIND.VD, the complete MIND file

Error Messages:

PROBLEM OPENING MIND.VD - the file created by DEALER cannot be opened.

VOCABULARY ITEM NUMBER MISMATCH - the number of vocabulary items input does not match the number used in DEALER.

25. ESDIT (Auxiliary)

Title: ESDIT.SV

Purpose:

ESDIT Is the example space file editor. It provides the capability to change individual start/stop values in the example space using an ESG or GENRLIZ printout. For a GENRLIZ printout, the starting record number and the end record number of the compressed data entered by the user must be offset.

Printout:

ESDIT produces a printout of the edit showing the element in the example space which was changed, and the old and new values associated with it.

User Dialog:

ESDIT

ENTER NAME OF EXAMPLE SPACE FILE:

ARE YOU EDITING FROM AN ESG PRINTOUT? (Y OR N):

(If the user answers no, then)

ARE YOU EDITING FROM A GENRLIZ PRINTOUT? (Y OR N):

```
ENTER RECORD NUMBER:
FILE:    ,STARTING RECORD:          ,ENDING RECORD:
```

(This is a statement of the current information in the specified record number.)

ENTER NEW STARTING RECORD:
ENTER NEW ENDING RECORD:

ARE THERE OTHER CHANGES TO THIS EXAMPLE SPACE FILE? (Y OR N):

(If there are further changes, inputs of record number, new starting record, and new ending record are requested.)

DO YOU WANT TO PROCESS ANOTHER EXAMPLE SPACE? (Y OR N):

(If another example space is to be processed, the user dialog is repeated.)

STOP

Input Files:

ES-, the example space file.

Output Files:

On output, the example space file is updated.

Error Messages:

STOP FILE STATUS ERROR

(The example space file status must be perfect for the program to continue.)

26. ESGDIT (Auxiliary)

Title: ESGDIT.SV

Purpose:

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

Printout:

A hardcopy listing of utterances used and those omitted is produced.

User Dialog:

```
ESGDIT

ENTER NAME OF EXAMPLE SPACE:

ENTER VOCABULARY ITEM (0...P):

ENTER NEW EXAMPLE SPACE NAME:

OLD FILE DESCRIPTION:  file description
ENTER NEW FILE DESCRIPTION:

STOP ESGDIT IS FINISHED
```

Input Files:

ES -, the example space file

Output Files:

ES -, the new example space file.

Error Messages:

```
FILE ALREADY EXISTS, FILE:  file name
ENTER NEW EXAMPLE SPACE NAME.
```

27. GASP (Auxiliary)

Title: GASP.SV

Purpose:

GASP (the Great American Speech Printout routine) was created during the closing moments of the LCSR project phase 0 for the final report. For each specified machine number, GASP prints the transition letter sets or the merged transition and loop letter sets.

Printout:

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

User Dialog:

GASP

ENTER THE 3-LETTER SUBDIRECTORY NAME:

ARE THE TRANSITION AND LOOP LETTER SETS TO BE MERGED IN THE PRINTOUT (Y/N)?

(If the letter sets are not merged, only the transition letter sets are printed for the machine number.)

ENTER THE STARTING MACHINE NUMBER:

ENTER THE END MACHINE NUMBER:

(GASP prints the machines in ascending order, beginning with the starting machine number and finishing with the end machine number. The current limits on the machine numbers are 0-15.)

STOP-ALL DONE

Input Files:

MC**.TL, Transition letter set file for the machine number **

MC**.LP, Loop letter set file for the machine number **

Output Files:

N/A

Error Messages:

INVALID ENTRY

(Invalid starting or end machine numbers were entered. Another input is requested.)

27. GASP (Auxiliary)

Title: GASP.SV

Purpose:

GASP (the Great American Speech Printout routine) was created during the closing moments of the LCSR project phase 0 for the final report. For each specified machine number, GASP prints the transition letter sets or the merged transition and loop lettar sets.

Printout:

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

User Dialog:

GASP

ENTER THE 3-LETTER SUBDIRECTORY NAME:

ARE THE TRANSITION AND LOOP LETTER SETS TO BE MERGED IN THE PRINTOUT (Y/N)?

(If the letter sets are not merged, only the transition letter sets are printed for the machine number.)

ENTER THE STARTING MACHINE NUMBER:

ENTER THE END MACHINE NUMBER:

(GASP prints the machines in ascending order, beginning with the starting machine number and finishing with the end machine number. The current limits on the machine numbers are 0-15.)

STOP-ALL DONE

Input Files:

MC**.TL, Transition letter set file for the machine number **

MC**.LP, Loop letter set file for the machine number **

Output Files:

N/A

Error Messages:

INVALID ENTRY

(Invalid starting or end machine numbers were entered. Another input is requested.)

CKST--FILE DOES NOT EXIST:

(If any of the input files do not exist for machine number **, this message is output and no machine printout is made.)

CKST--UNKNOWN ERROR;   FILE:

28. GWIZ (Auxiliary)

Title: GWIZ.SV

Purpose:

GWIZ is an auxiliary investigative program which delineates the words within an utterance using the given WIZARD statistics. It prints the compressed data file blatantly noting the delineations. The compressed data files to be used are listed in SUB: GWIZ.CD.

Printout:

GWIZ prints the compressed data files noting the delineations.

User Dialog:

```
GWIZ

ENTER THE 3-LETTER SUBDIRECTORY NAME:

STOP
```

Input Files:

GWIZ.CD, file containing the compressed data file names.
WIZ.ST, file of length and stretch factors which resides on the main directory.
-.CD, the specified compressed data files.

Output Files:

None

Error Messages:

```
CKST--FILE DOES NOT EXIST

CKST -- UNKNOWN ERROR:                    FILE:
```

(If a file error is detected on file WIZ.ST, then GWIZ terminates with the message

```
STOP - FILE WIZ.ST DOES NOT EXIST
```

GWIZ also terminates on an error from file SUB: GWIZ.CD. GWIZ outputs the error message and continues processing on an error from a compressed data file.)

29. MEND (Auxiliary)

Title: MEND.SV

Purpose:

MEND is an auxiliary program which creates example space for all the vocabulary items from the handmarked input data obtained from the GWIZ printout. These are the example spaces to be input to GZEC.

Programs GWIZ and MEND bridge the need to execute WIZARD and ESG given a WIZARD statistics file.

Printout:

None

User Dialog:

```
MEND

ENTER THE 3-LETTER SUBDIRECTORY NAME:

WARNING--FILES ES$<SUB>$** WILL BE DELETED FOR ALL MACHINE
 NUMBERS** (00-11).

 DO YOU WISH TO CONTINUE (Y/N)?
```

(If specified, the program terminates with STOP PROCESSING. Otherwise, the specified files are deleted and the program continues).

```
STOP ALL DONE
```

Input Files:

MEND.WD, file of handmarked input data to create the example spaces.

The handmarked input data file is organized as follows: Two lines are associated with data coming from each -.CD file. The first line (beginning in column 1) contains the number of words in the utterance and the -.CD filename. (For example, the utterance "1234" might produce: '4, LHN:A1234.CD where LHN is the subdirectory which holds the -.CD files and the A1234.CD is the relevant condensed data file.) The second line entry has the format: machine number, beginning record, end record separated by commas for each word in the utterance.

Output Files:

ES$SUB$**, example spaces for machine**, ** = 0,11

Error Messages:

```
CKST - FILE DOES NOT EXIST:

CKST -- UNKNOWN ERROR:                    FILE:
```

(If there is an error from the input file MEND.WD, the program outputs the message and terminates with)

```
STOP ON ERROR
```

A.9 FILE OESCRIPTION OF VDGS USER-CREATED FILES

File description o" VDGS user-created files are presented on the following pages.

Name:	Magic number sets
File:	MNSET* where * is a 1-character number set identifier.

Description: The card images for the number sets include the number spoken and the file name of the compressed data file as shown below. The digits plus the word "point" comprise the base vocabulary.

Note that the list includes 2 to 4 word numbers and that each list has the following properties:

1. Every digit occurs 15 times; the word "point" occurs 14 times.
2. Every digit occurs first 4 times and last 4 times; so does the word "point".
3. Every transition between two digits (e.g. 67, 68, 99 etc.) and between a digit and the word "point," and between the word "point" and a digit, occurs exactly once in each set.

For data collection purposes, each set is augmented with the eleven base vocabulary words; hence each set consists of 55 numbers (including the single word "point").

Created By: N/A

Format: Randomly organized, 256 words/record

Columns	Contents
1 - 4	Number, right-justified
5 - 6	Blanks
7 - 12	File name, ending with ".", right-justified

Name: Card image file for the counter data file editor

File: RVCARDS

Description: The card images for the counter data file editor contain the counter data record numbers to be kept in the new counter data file to be created.

Created By: N/A

Format: Randomly organized, 256 words/record

Note: The counter data record numbers to be kept are written from right to left in ascending order.

Columns	Contents
1 - 2	The number of entries on this card, right-justifed (no more than 10 entries. Otherwise, it is blank and 10 entries are on the card.) The last card contains -1 and the remainder of the card is blank.
3 - 4	Blanks
5 - 8	Counter data record number to be kept, right-justified, in ascending order
9 - 10	Blanks
11 - 14	Blanks or counter data record numbers to be kept, right-justified for each entry, in ascending order
17 - 20	
23 - 26	
29 - 32	
35 - 38	
41 - 44	
47 - 50	
43 - 56	
59 - 62	

Name: Prompting file for ESG or GWIZ

File: PFILE for ESG, GWIZ.CD for GWIZ

Description: This file holds the card images of the names of all the compressed data file comprising the total set of training data.

Created By: User

Format: Randomly organized, 256 words/record

Columns	Contents
1 - 8	compressed data file name (including -.CD extension)

Name: File of hand-cutting results

File: MEND.WD

Description: This user-created file holds all data gleaned from the hand-cutting procedure including the number of words in each utterance and the start and end times for each word.

Created By: User

Format: Randomly organized, 256 words/record

	Columns	Contents
line 1 -	1 - 2	number of words in utterance, followed by a comma
	3 - 14	compressed data file name, including three letter subdirectory name, colon, data file name plus -.CD extension
line 2 -	1 - 22	machine number, beginning time, end time separated by commas for each word in the utterance

A typical entry for the compressed data file A1234.CD might then look like this:

```
4,A1234.CD
1,1,25,2,20,50,3,45,80,4,70,100
------ ------- ------- --------
 "1"     "2"     "3"     "4"
```

Name: RESCUE indexes identifying numbers of best transition letter sets

File: REDEEM

Description: For the CHAINMIND version of VDGS, REDEEM is a user created file holding the RESCUE indexes of each transition letter set to be chosen by RESCUE, one per line, in order, so that the first number corresponds to machine 0, the second to machine 1, etc.

Created By: User

Format: Randomly organized, 256 words/record

Columns	Contents
1 - 2	RESCUE index for this item.

Name: Data location file

File: WHERE

Description: For tha CHAINMIND version of VDGS, the file WHERE holds the location of the counter data files created by REVEX. This location is to be specified in the form disk unit:subdirectory name.

Created By: User

Format: Randomly organized, 256 words/record

Columns	Contents
1 - 7	disk unit:subdirectory name

For example, a typical entry in WHERE might be DP2:USG

A.10 DATA FILES AND COMMAND FILES FOR VDGS PROCESSING

The following pages contain tables of important data files used during VDGS processing, compile and load macros for all VDGS programs, and command files for the execution of CHAINMIND.

TABLE A4. SINE QUA NON FILES OF VDGS PROCESSING

Routines	Input files	Output Files
EXTRACT	MNSET*	-.CD, -.RD
ESG	PFILE -.CD WIZ.ST	ES$***$**
GZEC	ES$***$** -.CD	TRIX**.TM TRLS**.TM
RESCUE	TRIX**.TM TRLS**.TM REDEEM	MC**.TL
SIGH	MC**.TL	MC**.TL MC**.XY
LOOPER	ES$***$** -.CD MC**.TL	MC**.LP
REVEXA	MNSET* -.CD MC**.TL MC**.LP	CDAT.RV CIDX.RV
RVDIT	CDAT.RV CIDX.RV (RVCARDS)	CDAT**.RV
COVERT	CDAT**.RV	CM** COV.ST
INVERT	CM**	INCM**
CROAK	INCM** CDAT**.RV CIDX.RV COV.ST	QDAT**.RR QDAT**.AF MUDT**.RR MUDT**.AF RVX.ST
REVEX	INCM** COV.ST RVX.ST MNSET* MC**.TL MC**.LP	CDAT.RV CIDX.RV

TABLE A4. SINE QUA NON FILES OF VDGS PROCESSING (Cont)

Routines	Input files	Output Files
ADDER	CDAT.RV CIDX.RV	LTVF.MM
AVRAJ	CDAT.RV CIDX.RV	AVRWRD.ST
CRAP	AVRWRD.ST CDAT.RV CIDX.RV	CONGAP CAP.ST
GAPSTER	CONGAP CAP.ST	GAPMAX GAP.DT QASM.DT
SORTRA/ SORTRB	GAPMAX/ CONGAP	GAPMAX/ CONGAP
MUTE	MUDT**.RR MUDT**.AF	LOOPY QLSTATS
GLOVE	QDAT**.RR QDAT**.AF	QDFT**.RR QDFT**.AF APA APR
TAILOR	QDFT**.RR QDFT**.AF	WHAT
BUILDER	LOOPY WHAT	MDFL.MM
DEALER	CAP.ST WHERE LVTF.MM MC**.TL MC**.LP COV.ST INCM** RVX.ST MDFL.MM QASM.DT GAP.DT	MIND.VD
PHEW	MIND.VD APA APR	MIND.VD

TABLE A4. SINE QUA NON FILES OF VDGS PROCESSING (Cont)

Routines	Input files	Output Files
(Auxiliary Routines)		
GASP	MC**.TL MC**.LP	none
ESDIT	ES$***$--	ES$***$**
ESGDIT	ES$***$--	ES$***$**
GWIZ	GWIZ.CD -.CD	none
MEND	MEND.WD	ES$***$**

TABLE A5. COMPILE AND LOAD MACROS FOR VDGS ROUTINES

Routines	Compile macro	Load macro
EXTRACT	EXTCP.XM	EXTLD.XM
ESG	ESGCP.XM	ESGLD.XM
GZEC	GENCP.XM	GENLD.XM
RESCUE	RESCP.XM	RESLD.XM
SIGH	SIGHCP.XM	SIGHLD.XM
LOOPER	LPCP.XM	LPLD.XM
REVEXA	RVXACP.XM	RVALD.XM
RVDIT	RVDCP.XM	RVDLD.XM
COVERT	COVCP.XM	COVLD.XM
INVERT	INVCP.XM	INVLD.XM
CROAK	CROCP.XM	CROLD.XM
REVEX	RVXCP.XM	RVLD.XM
ADDER	ADDERCP.XM	ADDERLD.XM
AVRAJ	AVRAJCP.XM	AVRAJLD.XM
CRAP	CRAPCP.XM	CRAPLD.XM
GAPSTER	GAPSTERCP.XM	GAPSTERLD.XM
SORTRA	SORTRACP.XM	SORTRALD.XM
SORTRB	SORTRBCP.XM	SORTRBLD.XM
MUTE	MUTECP.XM	MUTELD.XM
GLOVE	GLOVECP.XM	GLOVELD.XM
TAILOR	TAILORCP.XM	TAILORLD.XM
BUILDER	BUILDERCP.XM	BUILDERLD.XM
DEALER	DEALERCP.XM	DEALERLD.XM
PHEW	PHEWCP.XM	PHEWLD.XM

TABLE A6. COMPILE AND LOAD MACROS FOR SPECIAL CHAINMIND ROUTINES

Routines	Compile macro	Load macro
ZESG	ZESGCP.XM	ZESGLD.XM
ZGZEC	ZGENCP.XM	ZGENLD.XM
ZRESCUE	ZRESCP.XM	ZRESLD.XM
ZSIGH	ZSIGHCP.XM	ZSIGHLD.XM
ZLOOPER	ZLPCP.XM	ZLPLD.XM
ZREVEXA	ZRVXACP.XM	ZRVALD.XM
ZRVDIT	ZRVDCP.XM	ZRVDLD.XM
ZCOVERT	ZCOVCP.XM	ZCOVLD.XM
ZINVERT	ZINVCP.XM	ZINVLD.XM
ZCROAK	ZCROCP.XM	ZCROLD.XM
ZREVEX	ZRVXCP.XM	ZRVLD.XM
ZADDER	ZADDERCP.XM	ZADDERLD.XM
ZAVRAJ	ZAVRAJCP.XM	ZAVRAJLD.XM
ZCRAP	ZCRAPCP.XM	ZCRAPLD.XM
ZGAPSTER	ZGAPSTERCP.XM	ZGAPSTERLD.XM
ZMUTE	ZMUTECP.XM	ZMUTELD.XM
ZGLOVE	ZGLOVECP.XM	ZGLOVELD.XM
ZTAILOR	ZTAILORCP.XM	ZTAILORLD.XM
ZDEALER	ZDEALERCP.XM	ZDEALERLD.XM
ZPHEW	ZPHEWCP.XM	ZPHEWLD.XM

TABLE A7. COMPILE AND LOAD MACROS FOR AUXILIARY ROUTINES

Routines	Compile macro	Load macro
ESDIT	ESDCP.XM	ESDLD.XM
ESGDIT	ESG1CP.XM	ESG1LD.XM
GASP	GSPCP.XM	GSPLD.XM
GWIZ	GWZCP.XM	GWZLD.XM
MEND	MENCP.XM	MENLD.XM

TABLE A8. DATA FILES DELIVERED WITH VDGS

MNSET* - magic number set files
WIZ.ST - file of length and stretch factors
PFILE - prompting file used by ZESG in CHAINMIND

TABLE A9. COMMAND FILES FOR CHAINMIND

```
/3-26-79
/GENTL - MACRO TO CREATE EXAMPLE SPACES AND TRANSITION LETTER SETS
/THIS IS THE FIRST MACRO OF CHAINMIND.
DELETE ES$-. -
DELETE TRIX-. TM TRLS-. TM
MESSAGE START PROGRAM ESG
ZESG
MESSAGE START PROGRAM GZEC
ZGZEC
```

```
/3-26-79
/MAKEMIND - MACRO TO BUILD MIND FILE GIVEN EXAMPLE SPACES AND TRANSITION LETTER SETS
/THIS IS THE SECOND MACRO IN CHAINMIND.
DELETE MC-.TL
MESSAGE START PROGRAM RESCUE
ZRESCUE
MESSAGE START PROGRAM SIGH
ZSIGH
DELETE MC-. XY MC-. LP
MESSAGE START PROGRAM LOOPER
ZLOOPER
DELETE CDAT-. - CIDX. RV
MESSAGE START PROGRAM REVEXA
ZREVEXA
MESSAGE START PROGRAM RVDIT
ZRVDIT
MESSAGE START PROGRAM COVERT
ZCOVERT
MESSAGE START PROGRAM INVERT
ZINVERT
MESSAGE START PROGRAM CROAK
ZCROAK
DELETE CDAT-. - CIDX. RV QDAT-. - MUDT-. - RVX.ST
MESSAGE START PROGRAM REVEX
ZREVEX
MESSAGE START PROGRM RVDIT
ZRVDIT
MESSAGE START PROGRAM CROAK
ZCROAK
MESSAGE START PROGRAM ADDER
ZADDER
MESSAGE START PROGRAM AVRAJ
ZAVRAJ
MESSAGE START PROGRAM CRAP
ZCRAP
MESSAGE START PROGRAM G[illegible]ER
ZGAPSTER
MESSAGE START PROGRAM SORTRA
SORTRA
MESSAGE START PROGRAM SORTRB
SORTRB
DELETE GAPMAX QASM.DT GAP.DT
MESSAGE START PROGRAM GAPSTER
ZGAPSTER
DELETE LOOPY
MESSAGE START PROGRAM MUTE
ZMUTE
DELETE APA APR
MESSAGE START PROGRAM GLOVE
ZGLOVE
```

```
DELETE WHAT
MESSAGE START PROGRAM TAILOR
TAILOR
MESSAGE START PROGRAM BUILDER
BUILDER
MESSAGE START PROGRAM DEALER
ZDEALER
MESSAGE START PROGRAM PHEW
ZPHEW
```

APPENDIX B

PERFORMANCE ANALYSIS SUBSYSTEM USERS MANUAL

B.1 PROGRAM DESCRIPTIONS

The following pages include descriptions of the four programs which comprise the Performance Analysis Subsystem (PASS) of VIAS. These programs are designed to exercise BIGMINT, the research version of MINT, and to analyze the data collected by BIGMINT.

BIGMINT

Title: BIGMINT. SV

Purpose:

The purpose of BIGMINT is to find the 10 best explanations of each utterance, as viewed by the Mint algorithm.

Printout:

For each utterance:

1) A table of properties and costs associated with each node in the utterance.

2) The IQGAP matrix, showing which gap costs were computed and the resulting costs.

3) A table of the ten best paths.

User Dialog:

```
BIGMINT <MEX DATA PACKETS>/I <OUTPUT FILE>/O <MIND FILE>/D
        <NUMBER OF UTTERANCES TO PROCESS>/N [LISTING FILE]/L

STOP ALL DONE
```

Global Switches:

/A Use STATSUM type MIND file
/P Print the MIND file

Note: If the /A option is not used, BIGMINT will create a STATSUM type MIND file named "STATSUM.VD".

Local Switches:

/D MIND file of either type

/I input file name of data packers created by MEX using the global /A option.

/O cutput file of the ten best paths for use in STATSUM. The output file must exist.

/N the maximum number of data packets to be processed.

Optional:

/L listing file name, default is the line printer.

Input Files:

MIND file	- VDGS generated voice data file.
data packer file	- output from MEX with global /A option.

Output Files:

recognition file - (-.RE) contains the 1) best paths found by BIGMINT for later use in STATSUM.

Error Messages:

STOP NO MIND FILE GIVEN

(This occurs when no MIND file is given in the command line.)

STOP RECOGNITION FILE OPEN ERROR

(This occurs when the recognition file specified in the command line does not exist.)

NON-MATCHING NUMBER OF MACHINES BETWEEN VOICE DATA FILE:<MIND FILE> AND MINT, NAMELY <MINTS #> AND <MIND FILES #>

(If the number of machine types MINT expects is not equal to the number of machine types the MIND file was created for, MINT won't run. To correct this, recompile MINT with new value for parameter "MACHN".)

NON-MATCHING REVISION KEYS BETWEEN VOICE DATA FILE: <MILD FILE> AND MINT, NAMELY <MIND FILES REVISION KEY> AND <MINTS REVISION KEY>

(If these keys are different, it means that the MIND file and MINT are expecting different formats of the voice data.)

STOP - END OF DATA PACKETS

(This happens when the number of packets to process specified in the command line with the local /N is greater than the number of utterances in the input file.)

STATSUM

Title: STATSUM.SV

Purpose:

The purpose of STATSUM is to break up the decision-making process used in the MINT algorithm into its component parts. This enables the user to examine the contribution of each of the MINT cost functions.

STATSUM has three main functions:

1) To determine the recognition error category and type for each utterance.

2) To gather data for later use in SSPLOT.

3) To gather data for use in LICVAT.

Printout:

For each utterance:

1) The total cost over each path of each cost component.

2) The category and type of the "toughest critical decision".

3) The difference in total costs between each incorrect path, and the best correct path.

User Dialog:

```
STATSUM <MEX DATA PACKETS>/I <OUTPUT FILE>/O <MIND FILE>/D
        [LISTING FILE]/L

STOP - STATSUM ALL DONE
```

Global Switches:

/A	use STATSUM type MIND file
/N	create the data for SSPLOT
/P	print the MIND file
/Q	generate the data for LICVAT

Note: If the /A option is not used, STATSUM will create a STATSUM type MIND file named "STATSUM.VD".

Local Switches:

/D MIND file of either type

/I input file name of data packets created by MEX using the global /A option.

/O output file of the ten best paths for use in STATSUM. The output file must exist.

Optional:

/L listing file name, default is the line printer.

Input Files:

MIND file — (—.VD) VDGS-generated voice data file.

data packet file — (—.PK) output from MEX with global /A option.

recognition file — (—.RE) contains the 10 best paths found by BIGMINT

STATSUM.NM — this file contains the base of the temporary files that STATSUM will append data to for later SSPLOT and LICVAT.

Note: STATSUM should contain "XXX." where XXX are any three valid characters for RDOS filenames.

Output Files:

XXX.00 temporary files to store data for
XXX.01 use in SSPLOT and LICVAT
.
.
.
XXX.12

SSCOUNTER — contains counts of gap occurrences intrinsic properties

Note: SSCOUNTER and the other temporary files are appended to and should be deleted and created before each block of data (test data, interim test data, etc.) that PASS is run over.

Error Messages:

STOP NO MIND FILE GIVEN

(This occurs when no MIND file is given in the command line.)

STOP RECOGNITION FILE OPEN ERROR

(This occurs when the recognition file specified in the command line does not exist.)

NON-MATCHING NUMBER OF MACHINES BETWEEN VOICE DATA FILE:<MIND FILE> AND STATSUM, NAMELY <STATSUMS #> AND <MIND FILES #>

(If the number of machine types STATSUM experts is not equal to the number of machine types the MIND file was created for, STATSUM won't run. To correct this, recompile STATSUM with new value for parameter "MACHN".)

NON-MATCHING REVISION KEYS BETWEEN VOICE DATA FILE: <MIND FILE> AND STATSUM, NAMELY <MIND FILES REVISION KEY> AND <STATSUM'S REVISION KEY>

(If these keys are different, it means that the MIND file and STATSUM are expecting different formats of the voice data.)

STOP – NON MATCHING PACKETS AND RECOGNITION – PATHREAD

(This occurs when the MEX data packets and BIGMINT recognition data were created from different data.)

SSPLOT

Title: SSPLOT.SV

Purpose:

The purpose of SSPLOT is to plot cumulative distributions of each of the individual cost components used in the MINT algorithm, to evaluate the usefulness of each cost function as an information source, and to give a list of each interesting group (category and type) of errors for several magic number sets.

Printout:

For each category and type and cost of interest:

1) An ordered list of the utterances and costs differences used in each plot

2) A cumulative plot of the costs differences (if possible).

3) The amount of information contained in this cost.

User Dialog:

```
SSPLOT

ENTER DESCRIPTION OF PLOTS

DO YOU WANT TO PLOT ALL COSTS (Y OR N)?
```

(A "Y" answer causes a separate plot to be made for each cost component, an "N" will get the following question:)

```
ENTER COST # (1 - 12) THAT YOU WANT PLOTTED
OR -1 TO END
```

(The user may enter any or all of the costs, one at a time. After each number entered the question will be repeated until a "-1" is entered.)

```
DO YOU WANT A PLOT OF ALL CATEGORIES (Y OR N)?
```

(If you answer "Y" all categories will be plotted and the next question will be skipped. If you answer "N" the following question will appear.)

```
ENTER THE CATEGORY TO BE PLOTTED OR -1 TO END
```

(Here you enter the categories you want, one at a time the program will repeat the question after each entry, until you enter "-1" to end.)

```
DO YOU WANT A PLOT OF ALL TYPES IN CATEGORY 1 (Y/N)?
```

("Y" gets a separate plot for each cost for insertions, deletions, and substitutions, an "N" gets the following question:)

```
ENTER TYPE: 0 = (0,1), 1 = (1,0), 2 = (1,1), -1 = GO ON
```

(You enter -1, 0, 1, or 2 depending on the type you want. This question will also repeat until you enter -1.)

```
DO YOU WANT A PLOT OF INCORRECT RECOGNITIONS ONLY ON THE
SAME AXIS?  (Y/N)
```

(Enter "Y" or "N")

```
ENTER THE SCALE OF THE PLOT
```

(Enter an integer (10 is a nice number) for the scale of the cost axis of the plots.)

```
STOP - SSPLOT ALL DONE
```

Global Switches:

/A This will cause every possible cost, category, type, plot to be done with incorrect only on same axis and a scale of 10, and will eliminate all of the above questions.

Input Files:

STATSUM.NM	- This file contains the root of the temporary files ("XXX.-") used by SSPLOT.
XXX.-	- STATSUM-generated data files.

Output Files:

None

Error Messages:

None

LICVAT

Title: LICVAT.SV

Purpose:

The purpose of LICVAT is to test certain assumptions about the distribution of some of the properties and costs used in LISTEN. Namely, the occurrence of each violation category, the L-counter costs, the inter-word gap lengths, and the frequency of association between each pair of machine types.

Printout:

1) A table of counts of violation categories by machine type for real recognitions and artifact nodes.

2) A table of association counts machine type by machine type for both reals and artifacts.

3) A cumulative plot of a function of L-counter values designed to produce a rectangular distribution for reals and artifacts.

4) A table of start gap data by machine type

5) A table of end gap data by machine type

6) A cumulative plot of a function of gap values, designed to give a rectangular distribution for reals and artifacts.

User Dialog:

```
LICVAT

STOP ALL DONE
```

Global Ssitches:

/C	print the counts of violation categories and associations
/L	print the cumulative plot for L-counter data
/Q	print the start-end gap count data and the plot of the adjusted gap function

Note: If no global switch is given, LICVAT will do nothing

Input Files:

SSCOUNTER – the accumulated counts for violation, association, and gap data; STATSUM produced.

STATSUM.NM – holds the root of the name for the temporary files (XXX.-)

QLSTATS – created by MUTE, this file contains data necessary to compute the L-counter plot

XXX.- – the temporary files created by STATSUM

Output Files:

None

Error Messages:

None