A Vocal Data Management System*†

JEFFREY BARNETT

***Abstract*—This paper describes an implementation strategy for a vocal data management system (VDMS) being developed by the voice input/output project at the System Development Corporation. VDMS will accept connected speech of a language describable by 25–50 phrase equations and having a vocabulary of approximately 1000 words formed from about 100 data records.**

The strategy is based on the concept of predictive linguistic constraints (PLC). The present concepts of fixed directionality in parsing are replaced by a more generalized approach. To facilitate this flexibility, the system comprises a set of near-independent coroutines that are interconnected by a software busing structure. The VDMS acoustic processors verify the predictions. Very loose matching criteria are used for locating the predicted words. Special attention is given to word segments that are experimentally determined to be most invariant.

The Approach: Predictive Linguistic Constraints in Higher Level Processing

The ultimate goal of a speech-understanding system is the acceptance of and correct action upon a large subset of naturally spoken English, whether or not input utterances can be completely interpreted or translated to written text.

In view of the success in computer processing of natural and formal languages, [4], [5] one may wonder at the difficulties experienced in attempts

*Manuscript received April 30, 1972. This work was supported by the Advanced Research Projects Agency of the Department of Defense under contract DAHC15–67–C–0149. The author is with the System Development Corporation (SDC), Santa Monica, Calif. 90406.

†This article originally appeared in the IEEE Transactions on Audio and Electroacoustics AU–21 (3), 1973, 185–188.

to process speech utterances [1], [6]. A review of the literature indicates that: 1) the information provided by acoustic phonetics is neither complete enough nor invariant enough to ensure successful computer processing of speech; 2) almost all efforts at speech processing have been concentrated on low-level phenomena, such as the recognition of phones and isolated words (in this paper, low level and bottom end are equivalent and refer to direct processing of acoustic data); and 3) with two exceptions [2], [3], almost no consideration has been given to the higher level linguistic phenomena of connected speech.

The obvious approach to constructing systems that recognize continuous speech is to employ the linguistic constraints offered by syntactic and semantic rules describing the domain of discourse as aids in recognizing and understanding utterances. However, the method of applying these constraints is not clear. All current natural language processors employ fixed, directional strategies made up of some combination of left-to-right, right-to-left, top-to-bottom, or bottom-to-top techniques, and none handles ungrammatical inputs in any general way. (One system [5] does handle a subset of ungrammatical inputs.)

To use similar strategies for speech understanding seems unwise. Acoustic data would have to be used in near isolation to generate candidate units, which would then be passed upward for inspection. This would make it essential that the low-level acoustic processing be very accurate. However, achieving such accuracy would require significant breakthroughs, and no serious investigator is presently claiming any for the near future. We assume that progress must be sought through higher level nondirectional strategies.

A nondirectional scanning approach will allow parsing algorithms to process an input utterance as a whole. The benefits are the ability to initiate processing at any point in the utterance at which there is a high probability of correct match or, upon recognition of any linguistic unit,[1] the ability to use that information to limit the possibilities or fill in gaps at other points in either side of the utterance. A corollary to the use of nondirectional scanning is the need to review descriptive meta-languages as active hypothesis generators and predictors instead of passive constraint checkers and labelers. Since syntax matching must occur upon partial satisfaction of the criteria,

[1] "Linguistic unit" refers to words such as "automobile" or "General Motors" found in a lexicon and to meta-words describing phrases such as "relational expression" or "where clause"; it does not refer to phonetic labels.

the rule set must not only validate the relationships between units already found but must become active and predict, on the basis of information already obtained, the subsets of units that would satisfy the remainder of the constraints.

Predictive linguistic constraints (PLC) is our name for descriptive metalanguages and systems with the above features and capabilities, and for the concept we are applying to the construction of VDMS. The use of PLC techniques by the higher levels greatly relaxes the low-level criteria for matching or accepting the existence of a lexical item because the task is limited to verifying that a predicted candidate is in the input stream.

The following sections describe the architecture of a PLC-VDMS system that employs active rules, the PLC concept, and a hypothesis-verifying bottom end.

System Architecture

The system will consist of four types of entities: modules, buses, global stores, and an executive. The modules are coroutines with some special restrictions: they may not directly call one another; they have no arguments and return no values; and all references to external data are through functions provided by the executive. The busses are prioritized queues; only the highest priority item is seen on each bus. Global stores are common data regions of two types: static and semistatic. An example of a static store is a lexicon; the semistatic stores are communication regions used to give data long-term visibility not achievable by the bussing arrangement. The executive performs several tasks. All modules are executed, and the busses are then updated by discarding the highest priority items and bringing the next highest to the top of the queues. This procedure continues until either the input is "understood" or no module indicates a desire to continue the processing. The commonly available data are, in effect, hidden from the modules and may be accessed or modified only through the executive—an arrangement that will allow the synchronization problems of parallel computing coroutines to be finessed. (A Raytheon 704 mini-computer will handle the low-level acoustic recording and processing, while the top end of the system will operate under ADEPT on an IBM 370/145.)

The use of busses for communication should remove major dependency on order of execution among the modules. The only order-of-execution dependencies that may occur are those caused by order of updates to the global

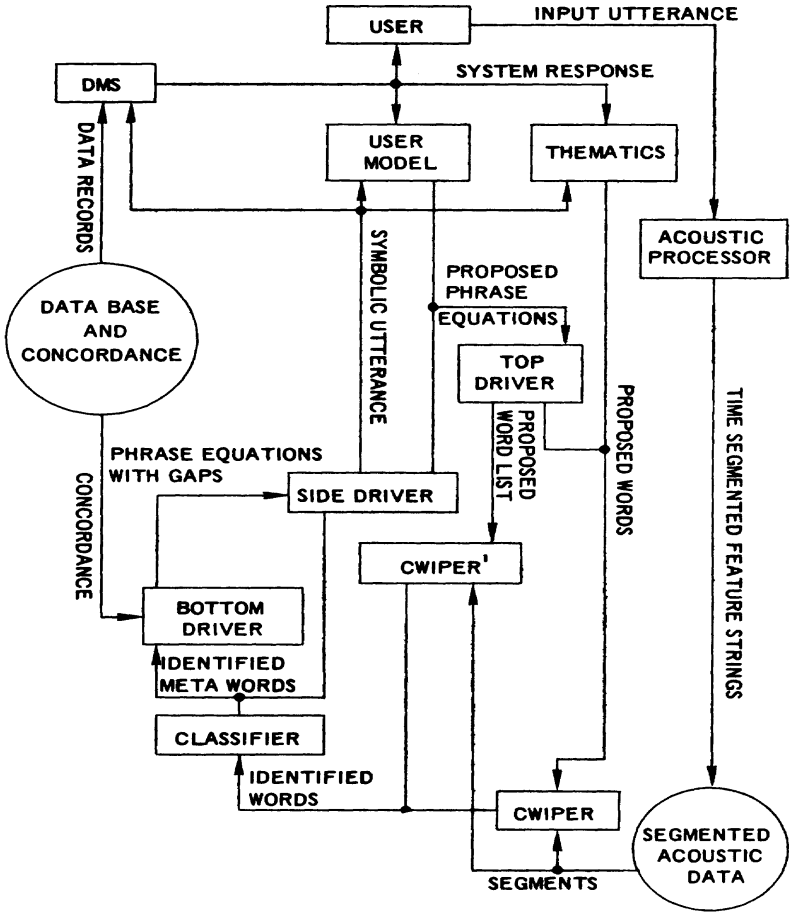

Fig. 1. PLC model for VDMS.

stores; if they hamper the long-term operation of the system, then the order of execution of the modules will have to be dealt with.

Fig. 1 is a gross representation of the system and some of the data flow paths among the modules (rectangles) and global stores (circles).

User-State Model

The user-state model establishes interutterance syntax limitations. A user may be thought of as being in a particular state or as trying to accomplish a particular task. For example, when he first approaches the computer (initial state), his only option is to log into the system. After logging in, he may query a data base or describe a report for off-line generation. The assumption

is that, after he enters either the interactive-query or report-generation state, he will remain in that state for several iterations. When he is finished with his activities, he signs off. A state of confusion may occur when he requests meta-information about the system. In each of these states, he uses a particular syntax; the system, by keeping track of what state he is in, can reduce to a minimum the number of syntactic possibilities for successive utterances. Thus, if the user is querying the data base, syntax for imperative report description would not be used except in a default or "nothing-else-works" situation.

Thematic Memory

The thematic memory is the content-word equivalent of the user-state syntax model. It is assumed that the user will exhibit goal-directed behavior toward finding specific information relating to a universe that is small compared to the whole data base. Content words (item names and values) contained in the last questions and answers are retained by the thematic memory and proposed as highly likely to occur in the next utterance. For example, consider the following query:

```
PRINT MANUFACTURER WHERE PRODUCT EQUALS AUTOMOBILE        (1)
```

and the answer is

```
GENERAL MOTORS, AMERICAN MOTORS, etc.                     (2)
```

The thematic memory would retain the words "manufacturer," "product," "automobile," "General Motors," "American Motors," etc. Each retained word is weighted by: 1) the number of times it has been used in the last several interactions; and 2) its use as (ranked from high to low): a) an item value in the answer; b) an unqualified item name in the question; c) a qualified item name in the question; or d) an item value in the question. Thus, the query

```
PRINT PRODUCT WHERE MANUFACTURER EQUALS AMERICAN MOTORS    (3)
```

is well predicted by the thematic memory assembled from (1) and (2).

Classifier

The classifier module is the output interface between the acoustic recognizers and the syntax modules. It simply identifies the recognized words as syntax terminals, item names, and/or item values.

Syntax Modules

Three modules handle intrautterance syntactic relationships: a bottom driver, a side driver, and a top driver.

In addition to syntactic relationships, the bottom driver also handles most of the intrautterance, nonsyntactic constraints. Recognized meta-words (phrases) are used to index the rule set for all phrase equations containing an item as a top-level occurrence. Previously recognized meta-words (phrases) are used to index the rule set for all phrase equations. The intrautterance rules that pertain to those items so far recognized are applied to determine whether the trial candidate is legitimate. The phrase, completed as best as possible, is bussed to the side driver.

The side driver examines the phrase to determine whether it is complete. If it is, the phrase label is used to bottom-drive the system up one level; if it is not, equations are formed that would allow successful filling of the holes, and the system is top-driven to search for such a match. If a completed phrase is a top-level equation, it is assumed to be the input utterance and is bussed to the various modules as shown in Fig. 1.

The syntax top driver strips key words (such as `PRINT` and `TALLY`) from the top-level equations to a shallow depth—say, two rules deep—to catch key words like `WHERE`. (The best depth will be determined experimentally.) The key words are proposed to the CWIPER modules for matching. It should be noted that content-word possibilities may be so restricted by the bottom and side drivers that they may be handled as alternative key words.

Some examples of restrictions arising from the bottom driver may help to clarify these procedures. For instance, if the word `MANUFACTURER` is found, a reduced phrase equation may be

$$\texttt{MANUFACTURER} \left\{ \begin{array}{c} \texttt{EQUALS} \\ \texttt{IS-NOT-EQUAL} \end{array} \right\} \text{manufacture name} \tag{4}$$

Where manufacturer name is `GENERAL MOTORS`, `CHRYSLER`, etc. The original

equation was

$$\text{item name} \quad \text{relation} \quad \text{item value.}$$

As another example, if the word CHRYSLER is found, then a reduced phrase equation is

$$\texttt{MANUFACTURER} \left\{ \begin{array}{c} \texttt{EQUALS} \\ \texttt{IS-NOT-EQUAL} \end{array} \right\} \texttt{CHRYSLER} \tag{5}$$

Of course, for each of these examples, other equations would also be proposed; but the reduction in the number of possibilities for a data base of the size described is a factor of between 500 and 50,000. The predictive powers of the system are enormous and, fortunately, easy to formulate. In the above, recognition of partial phrases quickly reduced the remaining items to syntactic terminals or a relatively restricted set of content words. The side driver would strip them and propose them to the top driver.

Low-Level Processing

Low-level processing in VDMS is accomplished by three modules: the acoustic processor, constrained word in phrase extraction routine (CWIPER), and CWIPER'. The acoustic processor, a combination of hardware and software that segments the speech signal in the time domain, is based on the Stanford Speech Recognition System [2] originally developed by Vicens and Reddy; the segmentation is tentative and is done to facilitate quick operation by CWIPER and CWIPER'.

CWIPER and CWIPER' perform hypothesis-verification tasks. CWIPER looks for a plausible lexical item in a time-constrained portion of the acoustical data. CWIPER' performs a similar task, except that a mutually exclusive list of plausible lexical items is presented along with the time constraints [an example would be locating either "EQUALS" or "IS-NOT-EQUAL"—but not both—for (4) and (5)]. The time constraint may be virtually nonexistent, as in the case of propositions from the thematic memory, which merely predicts the occurrence of a highly plausible word somewhere in the utterance, or it may be quite tight, as in the case of filling a gap between tentatively identified left and right constituents.

So that the CWIPER modules can take advantage of the looser matching criteria, a new technique for word recognition is being developed and tested.

Each lexical item (word) is recorded in several contexts of continuously spoken utterances. In each utterance, the item is isolated by the experimenter in an interactive mode by ear. Each copy of the item is then segmented by the acoustic processor and grossly labeled; the labeling resembles the recognition process described by Vicens. The least variable labeled segments are then determined and noted. This process is repeated for the entire vocabulary. From this data, a confusion matrix is built for the labeled segments showing little variation across contexts. It is assumed that such "invariants" will most often be stressed vowels or other interior segments high in energy and long in duration [7]. Finally, the item is stored in a lexicon along with its labeled patterns and data on the variance of individual segments.

Given such a lexicon, the plausibility-recognition procedure performed by CWIPER is fairly straightforward. The stored pattern for a proposed word is retrieved, and the least variable segment is used for a fast scan of the segmented acoustic input data. The confusion matrix is used to locate candidate segments. When a candidate is found, the rest of the word is matched in a middle-outward manner, and a goodness-of-match criterion is computed. The less variable the segment, the more emphasis it receives in the computation. Computations that produce a goodness criterion exceeding a threshold qualify the candidate as found. To facilitate this matching procedure, CWIPER may readjust the trial time-segmentation performed by the acoustic processor. This procedure grossly resembles Vicens's mapping.

Segmentation

Our acoustic-segmentation philosophy for the initial version of PLC-VDMS tends to place segment boundaries at points of transition. This is contrasted to the "transeme" approach [3], in which the boundaries are generally placed at steady-state points. (Although the transeme approach has produced some of the best results to date for continuous recognition, it is rejected for the present because of the unusual, nondirectional requirements of PLC and CWIPER and because of the complexity of an already formidable implementation task.) Because CWIPER is searching on least variable segments, it is necessary that steady-state phenomena be easily located.

In later versions of the system, both types of segmentation will be performed. Proposed words will be given not only with time constraints but with direct left and/or right neighbors when tentatively known. This will allow extended use of recognition and transition rules for adjacent items.

We realize that some such boundary-condition analyzer must eventually be included if the system is to obtain reasonable results. Hopefully, this approach will allow much of the present knowledge of acoustic processing to be included in a meaningful way to improve overall performance.

Conclusion

Speech-understanding systems modeled on directional natural-language processors do offer significant hopes of success. By replacing directional parsing strategies by nondirectional strategies, and by making descriptive metalanguages function as active predictors of hypotheses verified by low-level acoustic processors, we believe that a system that will understand continuous speech within a limited contextual domain is feasible in the next two to three years.

References

[1] A. Newell *et al.*, "Speech-understanding systems: Final report of a study group," Carnegie-Mellon Univ., Pittsburgh, Pa., Final Rep., May 1971.

[2] P. Vicens, "Aspects of speech recognition by computer," Stanford Univ., Stanford, Calif., AI Memo 85 (C5127), 1969.

[3] "The use of dynamic segments in the automatic recognition of continuous speech," IBM Corp., Rep. RADC-TR-70-22, May 1970.

[4] C. Kellogg *et al.*, "CONVERSE: Current status and plans," presented at the Ass. Comput. Mach. Symp. Inform. Retrieval, Univ. Maryland, College Park, Apr. 1971.

[5] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," Ph.D dissertation, Project MAC TR-84, Feb. 1971.

[6] S. P. Hyde, "Automatic Speech Recognition Literature Survey and Discussion," Post Office Res. Dep., Dallis Hill, London NW2, England, GOP Telecommunications Headquarters Res. Rep. 65, Sept. 1968.

[7] D. H. Klatt and K. N. Stevens, "Strategies for recognition of spoken sentences from visual examinations of spectrograms," Bolt Beranek and Newman, Inc., Cambridge, Mass., BBN Rep. 2154, June 1971.