

## "PSEUDO-NATURAL" HUMAN-COMPUTER DIALOG, USING A SPEECH TERMINAL

J.P. TUBACH, P. CESARINI.

### ABSTRACT

This paper discusses a dialog system, using a speech terminal connected to a computer. The dialog application provides the user with a "pseudo-natural" query language, to retrieve informations about historical facts, via connected speech requests. The vocabulary and syntax have been designed so as to be both efficient for good speech recognition, and natural enough for the speaker.

From a technical standpoint, an interface is provided to PASCAL application programs, that allows them to receive sequences of recognized words, and send answers to the synthesizer. This interface is used in a PROLOG interpreter, which supports knowledge about historical facts, and enables the computer to answers requests in a rather "intelligent" way.

### INTRODUCTION

Very often, speech recognizers are used as standalone boxes, just connected to a display, for demonstration and "evaluation" purposes. In some more sophisticated applications, they are actually used as computer peripherals, but just for data entry, and with very limited dialog protocols. We have been interested in the implementation of a more elaborate human-computer interaction, using speech recognition and synthesis.

After giving informations about the devices we have used, this paper will mainly discuss two points:

- the software interface we have developped in the computer, allowing an application programmer to use vocal I/O without specific knowledge of the speech terminal's operations and protocols
- an application where connected word recognition is used for "pseudo natural" queries to an "artificial intelligence" program managing an historical knowledge base.

### SPEECH HARDWARE USED

The general purpose computer we have used is a VAX 750, from D.E.C., running the VMS operating system.

The speech recognizer and synthesizer is a commercial product of the VECSYS Company, Bievres, France; it is known as "Speech Terminal" (S.T.) and is based on research conducted at LIMSI, Orsay, France. It incorporates a connected word recognizer (RME 186, also known as MOZART) and a speech synthesizer from text (ICO 85, also known as ICOLOG).

ENST, SYC Dept (CNRS, Assoociated Unit 820), PARIS, FRANCE.

MOZART is a single speaker, connected words recognizer, based on the classical "global" recognition approach, comparing an unknown utterance with stored references of the words in the limited vocabulary; the dynamic time warping algorithm is used for word to references distance computations. Pauses between words are not needed. A "syntax" can be defined, using a context-free grammar, to reduce the number of legal words at each step of sentence analysis (the satisfactory performance in our experiments is heavily related to this point).

When using this device standalone, the speaker has to utter the words of the vocabulary at least once (training phase) before going to the recognition phase (we will see later that this constraint is somewhat changed when connected to a computer)

The recognizer board has a NEC 7720 signal processor for spectral analysis (implementing a digital filter bank), and an INTEL 80186 microprocessor, running the dynamic time warping distance computations and decision process. The maximum number of acoustical references (maximum size of the vocabulary) in the 64K data memory is about 200, much more than was needed here, as we will see.

Those two boards are located in the same chassis, and one single V24 link to the computer is needed for communication. The general layout is shown on figure 1 (VAX, Speech Terminal and conventional display/keyboard terminal).

#### SOFTWARE INTERFACE

The software interface we have developed to support the S.T. is available to UCSD PASCAL application programs, under VMS

First level: communication procedures.

The most basic level of communication is achieved through two procedures: LECTURE(CODE) and ECRITURE(CODE), which take into account the protocol used between the VAX (Master partner) and the S.T. (Slave partner). LECTURE sends to the S.T. the CODE character string, with the required format:

STX /command/ [/data/] ETX

LECTURE reads the S.T.'s answer, which is /return\_code/ CR. With those procedures, all functions of the S.T. can be started from the computer, and the corresponding return codes tested. This includes terminal initialisation, references dictionary management (saving and restoring), choice of recognition mode, parameters modification, syntax management... LECTURE and ECRITURE are not needed by the application programmer, but are used by all higher level procedures.

Second level: application procedures.

Those are used by the application programmer. INIT MOZ is the first instruction to be used. It asks for S.T. initialization; conversational screens on the display allow the user to specify a vocabulary file, a syntax file, and an acoustical references file (if no references file was previously saved for that speaker, an online training is performed, with the specified vocabulary; if one is available, it is loaded into the S.T., along with the syntax file).

Results from connected word recognition are given to the application program by the READ MOZ(SENTENCE,NBWORDS) procedure: SENTENCE is an array of character strings and NBWORDS the (integer) number of words yielded by the S.T. (conversion of the S.T.'s internal word numbers to character strings is done using the informations in the vocabulary file loaded at INIT MOZ time).

Sentences to be "said" by the synthesizer are output by SYNTHESIS(MESSAGE), where MESSAGE is a character string (using the S.T.'s conventions for special characters used in French).

READ MOZ and SYNTHESIS are used for vocal I/O instead of PASCAL's standard READLN and WRITELN procedures. All mentioned procedures are contained, as object modules, in a library, used at link-edit time.

#### DIALOG APPLICATION

This application is aimed at giving the user the impression he asks questions in a "normal" spoken French, to a "smart" system. He gets "normal" spoken answers. The vocabulary has 23 words: 14 grammatical words (5 interrogative expressions, 6 verbs, 3 relationships) and 9 "knowledge" words about historical persons, places... Only this last subset has to be extended to incorporate more facts into the application. Figure 2 shows the language definition chart (vocabulary and syntax). The sentences are uttered naturally, in connected speech.

The "syntactic" words in the vocabulary have been carefully chosen: they take into account speech recognition constraints and are very natural in spoken French: for instance the short interrogative words "ou" or "qui" are replaced by "en quel lieu" or "qui est-ce qui", which are longer, and more colloquial.

The software architecture of this application is based on PROLOG (PROgrammation LOGique, logical programming); a simple interpreter for a subset of PROLOG has been implemented (using UCSD PASCAL) on the VAX, and can use the vocal interface described in the previous section.

The "program" contains assertions, set definitions and rules. An assertion is a known fact, assertions encode the "knowledge" this application has. For instance:

FACT4(EST NE, HENRI 4, 1553, A PARIS)  
means that "Henri the 4th was born in Paris in year 1553".  
Predicate FACT4 has four arguments, FACT3 and FACT5 are also available (e.g.: FACT3 for: "Napoleon's wife was Josephine" and FACT5 for: "Ravillac murdered Henri the 4th in Paris in year 1610").

Set definitions associate vocabulary words with the same syntactical and semantical function in the query language: for instance SET1 is TRUE for verbs EST NE, EST MORT, A ETE COURONNE (was born, died, was crowned), SET2 is TRUE for interrogative formulas EN QUELLE ANNEE, EN QUEL LIEU, EST-CE QUE and SET3 is true for persons: HENRI 4, FRANCOIS 1, NAPOLEON,...

Rules are conditional facts: they search the facts for the answer. The head of a rule is the QUESTION predicate, using formal variables (identifiers starting with an asterisk); the body of a rule checks the truth of set memberships and facts, and generates the answer if those

conditions are met (ANSWER generates a natural language answer via word concatenation, and SEND stores it in a buffer. For instance:

```
QUESTION(*INTERROG,*VERB,*PERS)
      < SET1(*VERB) & SET2(*INTERROG) & SET3(*PERS)
        & FACT3(*VERB,*PERS,*DATE,*PLACE)
        & SEND(ANSWER(*PERS,*VERB,*PLACE,*DATE))
```

handles questions such as "en quelle annee est mort Henri 4?" (in which year did henry the 4th die?), or "est-ce que Napoleon est mort a Paris?" (did Napoleon die in Paris?), yielding answers such as "Henri 4 est mort a Paris en 1610" (Henri the 4th died in Paris in year 1610). There are 20 such rules, 23 facts and 5 set definitions.

The interface to the S.T. has been used very easily: connected words from the recognizer are obtained via the RAD MOZ procedure, (see previous section): the NBWORDS words in the SENTENCE array are transferred into the QUESTION(SENTENCE[1],SENTENCE[2], ... SENTENCE[NBWORDS]) predicate. The answer is extracted from the buffer where it has been stored by the SYNTHESIS procedure, and sent to the synthesizer.

### CONCLUSION

This system has been demonstrated to the general public, very successfully, in a national computer exhibition. The syntactical constraints are strong enough to allow some multi-speaker operation (same sex, same "kind of" voice). This prototype demonstrates the feasibility of "rich" human-computer interaction including voice.

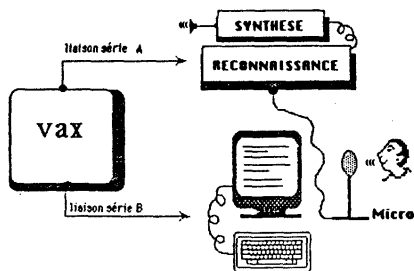


Figure 1

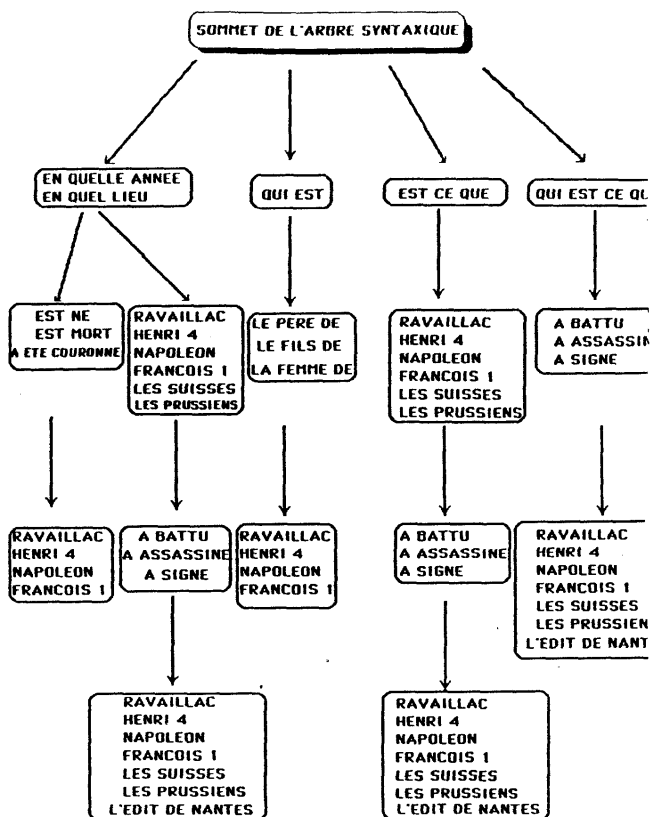


Figure 2 →