STANFORD RESEARCH INSTITUTE
Menlo Park, California 94025 · U.S.A.

December 1976

LIFER: A NATURAL LANGUAGE INTERFACE FACILITY

by

Gary G. Hendrix

Artificial Intelligence Center
Stanford Research Institute

Technical Note 135

SRI Project 7400D32 CTC

The work reported herein was conducted under
SRI's Internal Research and Development program.

LIFER: A NATURAL LANGUAGE INTERFACE FACILITY

Gary G. Hendrix
Artificial Intelligence Center
Stanford Research Institute
Menlo Park, California 94025

This note describes LIFER, a practical facility for creating natural language interfaces to other computer software. Emphasizing human engineering, LIFER has bundled natural language specification and parsing technology into one convenient package. This package includes an automatic facility for handling elliptical (i.e., incomplete) inputs, a spelling corrector, a grammar editor, and a mechanism that allows even novices to extend the language recognized by the system through the use of paraphrase. Offering a range of capabilities that supports both simple and complex interfaces, LIFER allows casual users to rapidly create workable systems while giving ambitious users the tools needed to produce powerful and more efficient language definitions. Experience with the system has shown that for some applications, very habitable interfaces may be created in a few days. The resulting systems are directly usable by business executives, office workers, and military officials whose areas of expertise lie outside the field of computer science.

LIFER is composed of two basic parts: a set of interactive language specification functions and a parser. In standard practice, a system builder uses the language specification

functions to define an application language. This application language is a subset of a natural language (e.g., English) that is appropriate for interacting with an existing software product. Using this language specification, the LIFER parser will then interpret natural language inputs, translating them into appropriate interactions with the application software.

Example interactions with a LIFER application language for a data base access system are presented in Figure 1. (This particular language definition, called INLAND, was developed by Earl Sacerdoti and others as a part of SRI's LADDER project for ARPA.) The system user types in a query or command in ordinary English, followed by a carriage return. The LIFER parser then begins processing the input. When analysis is complete, LIFER types "PARSED!" and invokes application software to respond.

An important feature of the LIFER parser is an ability to process elliptical (incomplete) inputs. Thus, if the system is asked

WHAT IS THE SPEED OF THE KITTY HAWK

then the input

OF THE ETHAN ALLEN

will be interpreted as WHAT IS THE SPEED OF THE ETHAN ALLEN. Analysis of incomplete inputs is performed automatically by LIFER, making it unnecessary for the system builder to explicitly define elliptical constructions in the application language.

FIGURE 1 EXAMPLE INTERACTIONS WITH LIFER

```
-What is the speed of the Kitty Hawk
PARSED!
((SPEED 35 KNOTS))

-Of the Ethan Allen
TRYING ELLIPSIS:  WHAT IS THE SPEED OF THE ETHAN ALLEN
((SPEED 30 KNOTS))

-Displacement
TRYING ELLIPSIS:  WHAT IS THE DISPLACEMENT OF THE ETHAN ALLEN
((STANDARD-DISPLACEMENT 6900 HUNDRED-TONS))

-length of the fastest Soviet sub
TRYING ELLIPSIS:  WHAT IS THE LENGTH OF THE FASTEST SOVIET SUB
((LENGTH 285 FEET / SPEED 30 KNOTS))

-Who onws the KIEV
     OWNS <==(assumed spelling error)
PARSED!
((COUNTRY USSR))

-who owns the JFK
TRYING ELLIPSIS:  ELLIPSIS HAS FAILED
THE PARSER DOES NOT EXPECT THE WORD "JFK" TO FOLLOW "WHO OWNS THE"
OPTIONS FOR NEXT WORD OR META-SYMBOL ARE:
<SHIP-NAME>

-Define JFK to be like Kennedy
PARSED!
 . {JFK is now a synonym for KENNEDY, which is a ship name}
 .
-REDO -2          {that is, parse WHO OWNS THE JFK}
PARSED!
((COUNTRY USA))

-? BUILT LAFAYETTE
TRYING ELLIPSIS:  ELLIPSIS HAS FAILED
 . {error message omitted}
 .
-Let "? built Lafayette" be a paraphrase of "who built the Lafayette"
PARSED!
 .
 .
-? built Lafayette
PARSED!
((BUILDER GENERAL.DYNAMICS))

-owns longest nuclear submarine
TRYING ELLIPSIS:  ? OWNS LONGEST NUCLEAR SUBMARINE
((COUNTRY USSR / LENGTH 426 FEET))
```

If a user misspells a word, LIFER attempts to correct the error, using the INTERLISP spelling corrector. If the parser cannot account for an input in terms of the application language definition, user-oriented error messages are printed that indicate what LIFER was able to understand and that suggest means of correcting the error.

The definer of the language interface can intermix calls to LIFER that extend or modify the language definition with calls to the parser that utilize the developing language system. This aids system builders in the task of defining the application language, allowing them to operate in a rapid extend and test mode. Perhaps more importantly, it provides the basis for a mechanism through which naive users may extend their language by employing easy-to-understand notions such as synonyms and paraphrases. Provisions may be included in the application language for interfacing with LIFER's own language specification functions, making it possible for users to give natural language commands for extending the language itself. This is illustrated by the paraphrase example of Figure 1.

The LIFER parser uses an augmented, finite state transition network (Woods, 1970). The LIFER language specification functions construct these underlying transition networks automatically from language production rules of the type commonly used by both natural linguists and compiler builders. The production rules may be easily modified and tested

interactively, allowing sophisticated language definitions to be produced within a short period of time.

In using LIFER, interface builders typically embed considerable semantic information in the syntax of the application language. For example, words like JOHN and AGE would not be grouped together into a single <NOUN> category. Rather, JOHN would be treated as a <PERSON>, and AGE as an <ATTRIBUTE>. Similarly, very specific sentence patterns such as

WHAT IS THE <ATTRIBUTE> OF <PERSON>

are typically used in LIFER instead of more general patterns such as

<NOUN-PHRASE> <VERB-PHRASE>.

For each syntactic pattern, the interface builder supplies an expression for computing the interpretation of instances of the pattern. Expressions for sentence-level patterns usually invoke application software to answer questions or carry out commands.

Example interactions defining a LIFER application language are shown in Figure 2. First, application information concerning biographic data is stored on property lists for later querying. Then function MAKE.SET is called to define some word/phrase categories. The category <ATTRIBUTE>, for example, is defined to include such words as AGE and OCCUPATION. Next, function PATTERN.DEFINE is used to add the productions

<ATTR-SET> => <ATTRIBUTE>

and <ATTR-SET> => <ATTRIBUTE> AND <ATTR-SET>

FIGURE 2 DEFINING AN APPLICATION LANGUAGE

```
   {set up data to be queried}
-SETPROPLIST(JEWELL.FLEMING (AGE 35 OCCUPATION TEACHER HEIGHT 5.5
                             WEIGHT 105))
-SETPROPLIST(IVAN.FRYMIRE   (AGE 40 OCCUPATION FARMER HEIGHT 6.2
                             WEIGHT 225))

   {MAKE.SET and PATTERN.DEFINE extend the language definition}
-MAKE.SET(<PERSON> (JEWELL.FLEMING IVAN.FRYMIRE ...))
-MAKE.SET(<ATTRIBUTE> (AGE OCCUPATION HEIGHT WEIGHT))
-MAKE.SET(<IS/ARE> (IS ARE))
-PATTERN.DEFINE(<ATTR-SET> (<ATTRIBUTE>)
                (LIST <ATTRIBUTE>))
-PATTERN.DEFINE(<ATTR-SET> (<ATTRIBUTE> AND <ATTR-SET>)
                (CONS <ATTRIBUTE> <ATTR-SET>))
-PATTERN.DEFINE((WHAT <IS/ARE> THE <ATTR-SET> OF <PERSON>)
                (MAPCONC <ATTR-SET> (FUNCTION (LAMBDA (A)
                    (LIST A (GETPROP <PERSON> A))))))

   {a call to LIFER.INPUT sends subsequent inputs to the parser}
-(LIFER.INPUT)

   {start NL interactions using grammar defined above}
-what is the occupation of jewell.fleming
PARSED!
(OCCUPATION TEACHER)
-age and weight
TRYING ELLIPSIS:  WHAT IS THE AGE AND WEIGHT OF JEWELL.FLEMING
(AGE 35 WEIGHT 105)

   {MAKE.SET is called to add variety to persons' names}
   {leading ! sends line to LISP'S EVAL, instead of to parser}
-!MAKE.SET(<PERSON> ((JEWELL . JEWELL.FLEMING)
                     (IVAN . IVAN.FRYMIRE)
                     ((JEWELL FLEMING) . JEWELL.FLEMING)
                     ((IVAN FRYMIRE) . IVAN.FRYMIRE))

   {now more English input}
-what is the height of ivan frymier
 (assumed spelling error)==>FRYMIRE
PARSED!
(HEIGHT 6.2)
-of jewell
TRYING ELLIPSIS: WHAT IS THE HEIGHT OF JEWELL
(HEIGHT 5.5)
   {define a paraphrase in English}
-define "give the height of ivan" like "what is the height of ivan"
PARSED!
LIFER.TOP.GRAM => GIVE THE <ATTR-SET> OF <PERSON>
   {output above shows LIFER's generalization of the paraphrase}
   {now try an input based on the paraphrase above}
-give the age and occupation of jewell fleming
PARSED!
(AGE 35 OCCUPATION TEACHER)
```

to the language definition, establishing an <ATTR-SET> as a list of one or more attributes separated bv ANDs. The third call to PATTERN.DEFINE sets up a top-level sentence pattern of the form

WHAT <IS/ARE> THE <ATTR-SET> OF <PERSON>

which can match such queries as

WHAT IS THE AGE AND OCCUPATION OF JEWELL.FLEMING

The expression for computing the value of this query maps down the list of sought-after attributes and extracts their values from the property list of the <PERSON>.

After the function LIFER.INPUT is called, all lines of input are sent to the LIFER parser for processing. The first query of the example is a complete sentence, but the second is elliptical. Note that no special patterns are needed to deal with this elliptic query. A more complex use of MAKE.SET and examples of the spelling corrector are shown in later examples. Many other features are available, including a grammar editor, aids for processing anaphora, and a mechanism for using LISP predicates to define syntactic categories.

LIFER is implemented in PDP-10 INTERLISP, with the basic system requiring an additional 14K words above the 150K used by INTERLISP. An extensive language definition for communicating with a data base of information about ships requires (including some data base access routines) an additional 30K. Such sentences as

WHAT IS THE LENGTH OF THE FASTEST SOVIET SUB

parse in less than 2 seconds of CPU time (faster than the sentences are usually spoken or typed).

REFERENCES

Woods, W. A. (1970), "Transition Network Grammars for Natural Language Analysis," CACM 13 (10), 591-606.