MAN-MACHINE INTEGRATION IN A LEXICAL PROCESSING SYSTEM

by

Richard L. Venezky
Nathan Relles
Lynne Price
Jack Bennett

Computer Sciences Technical Report #280

September, 1976

MAN-MACHINE INTEGRATION IN A LEXICAL PROCESSING SYSTEM

by

Richard L. Venezky
Nathan Relles
Lynne Price
Jack Bennett

Computer Sciences Department
University of Wisconsin
Madison, Wisconsin

September 28, 1976

Abstract

LEXICO is a computer system that may be used for dictionary construction and other areas of lexical analysis. Textual material may be stored, edited, concorded, and lemmatized to assist the researcher. The system is designed to facilitate use by three types of users: (1) the novice, who has had no previous experience with the system, (2) the sustained user, who has learned how to accomplish desired tasks, and (3) the occasional user, who has used the system, but sometimes forgets a specific command or procedure. Despite the many and varied tasks which can be performed, the system appears to the user as a single, unified entity with a consistent means of communication. Many of the automated capabilities, such as homograph separation, reduce the amount of work that would be required by manual procedures. The system has many features that facilitate its maintenance; in addition, monitoring of user and system behavior allows continual refinement of system capabilities.

1. Introduction

LEXICO is a computer system which assists lexicographers in storing, editing, concording, lemmatizing, and formatting information for dictionaries and for other products of lexical analysis.[1] The design of this system began almost five years ago as part of an experimental study of data processing techniques for natural language texts. Through successive improvements, the initial design has evolved into a production system which is now being used for a variety of text processing tasks, including initial processing for the Dictionary of Old English[2]. Like many newer language processing systems LEXICO provides a command language, on-line interaction with the user, and extensive file handling; however, it differs from these systems in two important characteristics.

First, LEXICO was designed from the beginning to facilitate use by non-professional programmers; decisions on such matters as system capabilities, interaction modes, error recovery, and documentation were strongly influenced by projected user needs. Second, since LEXICO began as a research project, techniques and components were tested and revised systematically over a five-year period and

software components for monitoring both user and system behavior were incorporated in the initial design. The resulting system, although neither error free nor ideal in any sense, has nevertheless been extensively tested and refined and continues to provide monitoring data for planning further improvements. The present chapter will present the design and use of the LEXICO system with emphasis on the two characteristics just mentioned.

2. Criteria for Software Design

Given a well defined set of functions, the software designed to realize these functions is usually shaped by two factors: (1) the hardware-software environment in which the new system will operate and (2) the programming capabilities and style of the software designer. Advances over the past five years in both of these areas have strongly influenced software design. For large machines, mass storage costs have declined dramatically, on-line access techniques have become routine, and operating systems have begun to offer support services which previously had to be provided by processing software (e.g., file management). On the other hand, the price of powerful small machines (viz., minicomputers and micro processors) has declined to the point where many tasks which were considered large machine jobs five years ago are now done more efficiently on a small machine.

For program implementation, similar advancements have occurred. New languages like SIMULA have been developed especially for simplifying the writing of large software systems; approaches to program organization, such as structured programming, have been widely accepted, and implementation procedures such as the chief programmer team approach (Baker and Mills, 1973) have been devised.

All of these advances have lightened the programmer's burden , allowed a more efficient use of hardware resources, and have resulted in more elegant and sophisticated software systems. Yet in a world in which human costs continue to rise, the needs of the users of computing systems are becoming at least as important as those of the system designers.

To a limited degree the computer industry has professed an awareness of this need. Most software systems are now labeled 'user-oriented' and a variety of so called user-oriented features are beginning to appear in operating-systems, compilers and other programs which are designed for diverse audiences. These include such mechanisms as front-end processors which translate simplified user commands into the (unnatural or unwieldy) commands which a particular system was designed to receive (Hagerty 1972), HELP commands that provide brief explanations of particular error messages, and user-defined synonyms for keywords in commands.

Although such features represent a positive shift in the design of user-oriented systems, they are more like a new coat of paint than they are a structural change in software design. Software systems are still designed on the basis of hardware and software desiderata, with the user receiving only post hoc consideration.

In contrast to most other systems, LEXICO was designed with three types of users in mind: (1) the novice who knows nothing about the system, (2) the sustained user who has learned how to accomplish what he needs to do, and (3) the occasional user who has used the system, but occasionally forgets a specific command or procedure. These user classes were selected over the more standard professional - non-professional dichotomy because of the tasks which LEXICO incorporates and the nature of the anticipated user community. Except for file handling, LEXICO functions are conceptually quite simple: enter a text into a collection, correct a misspelled word, concord, and so on. In specifying such tasks, the professional programmer should have little advantage over the non-professional.

3. Tasks in Lexical Processing

The initial goal for the LEXICO project was to explore how computers could be employed in the construction of dictionaries. At the time this project began, some experience in lexical processing had already been gained by the St. Thomas Aquinas Project (Busa, 1964), the Italian Dictionary Project (Zampolli, 1968), the Hebrew Dictionary Project (Ben-Hayyim, 1966), and the Dictionary of American Regional English (Venezky, 1968). In addition, software for specific lexical processing

tasks such as concording and text editing had been developed and tested on a number of different computers.[3] From these experiences, initial consideration was given to the following tasks:

A. Corpus management

Dictionaries, and especially academic dictionaries,[4] generally are based upon quotations extracted from published materials. Until recently, quotations were extracted by human readers (either professional or volunteer). The Oxford English Dictionary, for example, managed to recruit almost 800 volunteer readers, who read their way through approximately 4,500 published works, extracting 656,900 quotations in the process.[5]

However, optical scanners and direct text entry techniques allow a multi-million word corpus to be translated into machine readable form at a cost which many projects have already found to be tolerable. Therefore, the first task for a lexicographic system is to facilitate the input and correcting of texts.

B. Selection of quotations

In the hand procedure just described, quotations are selected and recorded by readers. In a computer-based system, on the other hand, quotations can be extracted automatically. However, to avoid a deluge of quotations for the most common forms (such as the, of,

and in for English), and to ensure that sufficient syntactic and semantic context is included to allow proper sense selection, the dictionary editor must have some control over the words for which quotations are or are not selected and over the boundaries of the quotations themselves.

C. Lemmatization

In a non-mechanical system, quotations are recorded on slips, which are then usually sorted alphabetically on the words for which each quotation was selected. This sorting, however, is generally done on lemmata or headwords and not text forms. For a computer to do such a task requires that (a) alternate spellings be recognized and normalized, and (b) correspondences between text forms and lemmata be recognized. Partial schemes for such processes have been suggested for specific langages (e.g., Price, 1969) but no approach which could be applied to all languages has yet been devised.

D. Sense categorization

Once a lemmatized quotation (slip) file is formed, various editorial processes begin which lead eventually to the set of entries which compose the dictionary. The first of these is sorting quotations into sense classes, a task for which no computable algorithms exist for any language. However, much of the bookkeeping in this task might be computer-assisted.

E. Entry writing

Once sense categories for a word are derived and quotations sorted into them, the initial drafting of an entry begins. Before this process is completed, however, a variety of supplementary materials may be consulting, including word studies, other dictionaries and word lists, and the opinions of consultants. Whatever may be marshalled for assisting the editor, the end product is an entry which contains a description of the word's different senses, with (usually) illustrative quotations for each, in addition to supplementary items which vary with dictionary, period covered, and intended audience: pronunciation, variant orthographic and morphological forms, grammatical classes, etymology, synonyms and antonyms, etc.

Cogitation and inscription are clearly reserved for humans in this process, but formatting, insertion of quotations, and various types of error checking are candidates for computer assistance.

F. Publication

Initial drafts of entries are usually reviewed, revised, and then rechecked for accuracy and consistency. Finally, the dictionary is printed. Both editing by on-line terminal and publication by computer-driven photo composition are applicable to this stage of dictionary making.

4. Design Goals for LEXICO

From an analysis of the tasks just described and from interviews with dictionary editors, a set of functions were selected for computer implementation and testing. These included:

1. Formation and maintenance of a text collection
2. Text editing
3. Concording
4. Lemmatization
5. Slip generation

In addition, various utility functions were identified, including copying texts, deleting texts, and correcting concordance output. Of these the first two were selected for implementation but the third rejected because of the potentially high cost of such an operation. Sense categorization was considered for inclusion, but no efficient technique was found to justify its computer implementation. Assistance for entry writing and publication require special equipment (e.g., a photocomposition device) which was not available to the project.[6]

The users of LEXICO were seen as non-professional programmers: editors, linguists, and the like, who were comfortable with non-mechanical techniques and would not convert to a computer system just for novelty. In addition, the intended users would need to remain in constant communication with their data as it flowed through the various LEXICO processes. Previous experience with concording systems had shown that simple errors, if not corrected soon enough, could lead to disastrous expenses when large texts were involved. Therefore, constant monitoring was desired.

Finally, as discussed above, at least three different types of users were identified: the novice, the sustained user, and the occasional user.

Because of the large number of parameters which must be specified in doing certain tasks and the relatively unfamiliar vocabulary of lexicography, the novice, sustained user, and occasional user have widely differing needs. The novice needs to know what functions the system can perform and the statements required for initiating these functions, in addition to learning how and when to create backup files, how to recover from syntax errors, and which servicing priority to select for different cost and time demands.

On the other hand, the sustained user, once he acquires this knowledge, wants to make requests as briefly as possible. He needs little extra documentation; errors are quickly recognized and therefore do not require extensive explication; and servicing priorities are predetermined. Between the novice and the sustained user, however, is another class of users, the occasional users. These are people who mastered the system once, but because of infrequent practice, often forget an essential item. They need some assistance, but their memories can often be jarred with a minimum of verbiage. Thus, while the novice may require a full page explanation of a certain command, the occasional user may need only a sentence or two, and the sustained user desires at most an abbreviation.

From the intended users of LEXICO and from our previous experience with other text processing systems, we developed the following general design criteria:

A. The user should be totally isolated from the operating system. Certainly the user should not be required to understand file formats, program characteristics, and the like. But just as important is the need to isolate the user from the ciphers and incantations required to create files, use programs, and enter data. The user should be aware only of the data and tasks that are at his level of discourse. (A corollary of this requirement is that all communication between a user and the system should be in the user's vocabulary.)

B. The system should appear to the user as a single, unified entity, with consistent means of communication. The manner in which a user specifies tasks to be performed on texts should be similar for all tasks. The manner in which the system responds to user requests should likewise be consistent throughout the system.

C. The means of specifying tasks to be performed should be flexible and easy to use, and should minimize the chance of user errors. Previous experience demonstrated the inadequacy of an ever-expanding number of fixed-field control cards for specifying task parameters. For this reason, a command language is required for task specification and for ensuring a consistent means of communicating with all components of the system.

D. The user should be required to specify as little as possible to have a task performed. To achieve this, the system should have several levels of default values. These values are assumed for any parameters which are not overtly specified by the user.

E. The system should be easy to maintain. In particular, it should be possible for programmers maintaining the system to implement new capabilities, modify the command language to accomodate new or revised capabilities, and trace system errors with minimal effort. Facilities should be provided for monitoring both user and system behavior.

F. The system should not be too expensive to use. While this might seem an ovious goal, it needs to be emphasized that LEXICO was designed as a practical system. In addition to demonstrating the feasibility of automating certain tasks, our goal was to provide a useful and practical alternative to other methods. For this reason, all design decisions had to include a careful consideration of user costs.

These design goals were not always compatible with each other. Much of the system's development consisted of ensuring that the primary goals, efficiency and ease of use, would be satisfied. In some cases, entire components of the system were replaced or eliminated to achieve these goals.

5. The User's View

The user of LEXICO works with one or more collections of texts. A collection is composed of texts which the user wants to process in similar ways. In addition to the texts themselves, a collection contains a directory identifying and describing each text and may contain any of the following:

(1) conventions to be used when entering or concording texts;

(2) spelling conversion rules to be used for headword classification;

(3) a text form--base form conversion list to be used for headword classification;

(4) concordances of texts;

(5) for each concorded text, a list of words occurring in the text.

Features which may be common to all texts in a collection are called collection defaults. Once specified for a collection, these features need not be respecified for each text.

Figure 1 illustrates the components of a collection which are accessible by the user.

Insert Figure 1 about here

The typical processes which can occur for a collection of texts include:

(1) creation of a collection and, optionally, definition of collection default values;

(2) entry of one or more texts into the collection, optionally producing a listing of each text;

(3) correction of errors in texts (editing);

(4) concording texts;

(5) formation of a headword classification (lemmatization) process for the collection by definition of spelling conversions and base form--text form associations;

(6) headword classification of the words that occur in texts, producing, for each text, a listing of the headword (base form) associated with each text form;

(7) resorting of a concordance by base form

(8) generation of slips [7]

(9) deletion of texts from the collection; and

(10) deletion of the collection from the system (after all processing is complete).

In addition, users may at any time examine a summary (the collection directory) of the processes performed on each text. Users may also obtain listings of any text, of the spelling conversion rules and base form--text form associations, or of the base forms associated with each word in a text. The latter listing may be generated showing the contexts in which forms of each base occur. Also, collection default values may be inspected or updated; the headword classification process may be revised; and the text form--base form associations of any text may be corrected.

System Commands and Aids

The user enters commands to LEXICO from a remote terminal. Many tasks are performed on-line with the results displayed immediately at the same terminal. However, some tasks require so much time, generate so much output, or cost so much, that they are performed off-line. In this case, the user describes a task by commands entered at the terminal, but the tasks are actually performed later and results printed at the computing center.

Tasks are specified to the system by entering statements in the LEXICO command language. Most tasks are specified in the system as functional blocks. A block begins with a block header which is a command that identifies the task (e.g., CREATE, ADD, EDIT). Subsequent commands specify how the designated task is to be performed (e.g., delimiter specifications, printing options, collating sequence). Following these commands and declarations the END command is used to signify that the task has been completely described. (Many tasks are specified in one command and therefore do not require an END command.)

After some commands are entered, LEXICO asks a question of the user; for example, the command

```
EDIT ;
```

is followed by the prompt, WHICH TEXT?.

During interaction, the user may enter special commands called user aids to LEXICO for assistance in understanding error messages, questions, or requests for information. These capabilities are especially suited to the novice user,

but, of course, may be used by anyone.

These user aids include

EXPLAIN ERROR (*err). If an error is made, the system displays a brief description of that error. In response to *err, a more detailed explanation is displayed. *err may be entered several times for progressively more detailed explanations.

EXPLAIN QUESTION (*equ). Occasionally the system will solicit information in the form of a brief question. By successively entering *equ, progressively more detailed explanations of the solicited information are displayed.

EXAMPLE (*exa). Whenever the system solicits information or when an error has been made, examples of correct input may be obtained with this command.

MENU (*mnu). This command causes the system to display all commands allowed in the block in which the user is working.

Figure 2 diagrams communication between LEXICO and a user.

- - - - - - - - - - - -

Insert figure 2 about here

- - - - - - - - - - - -

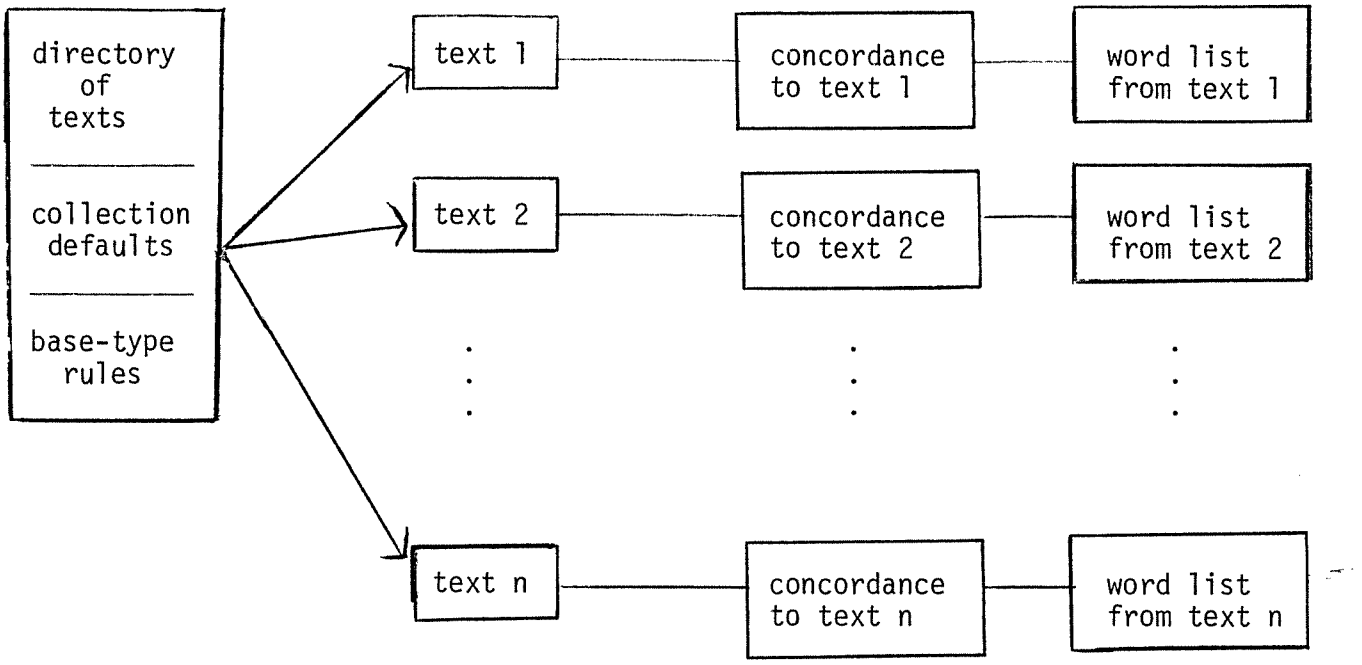


Figure 1

Components of a Collection which are accessible to the User

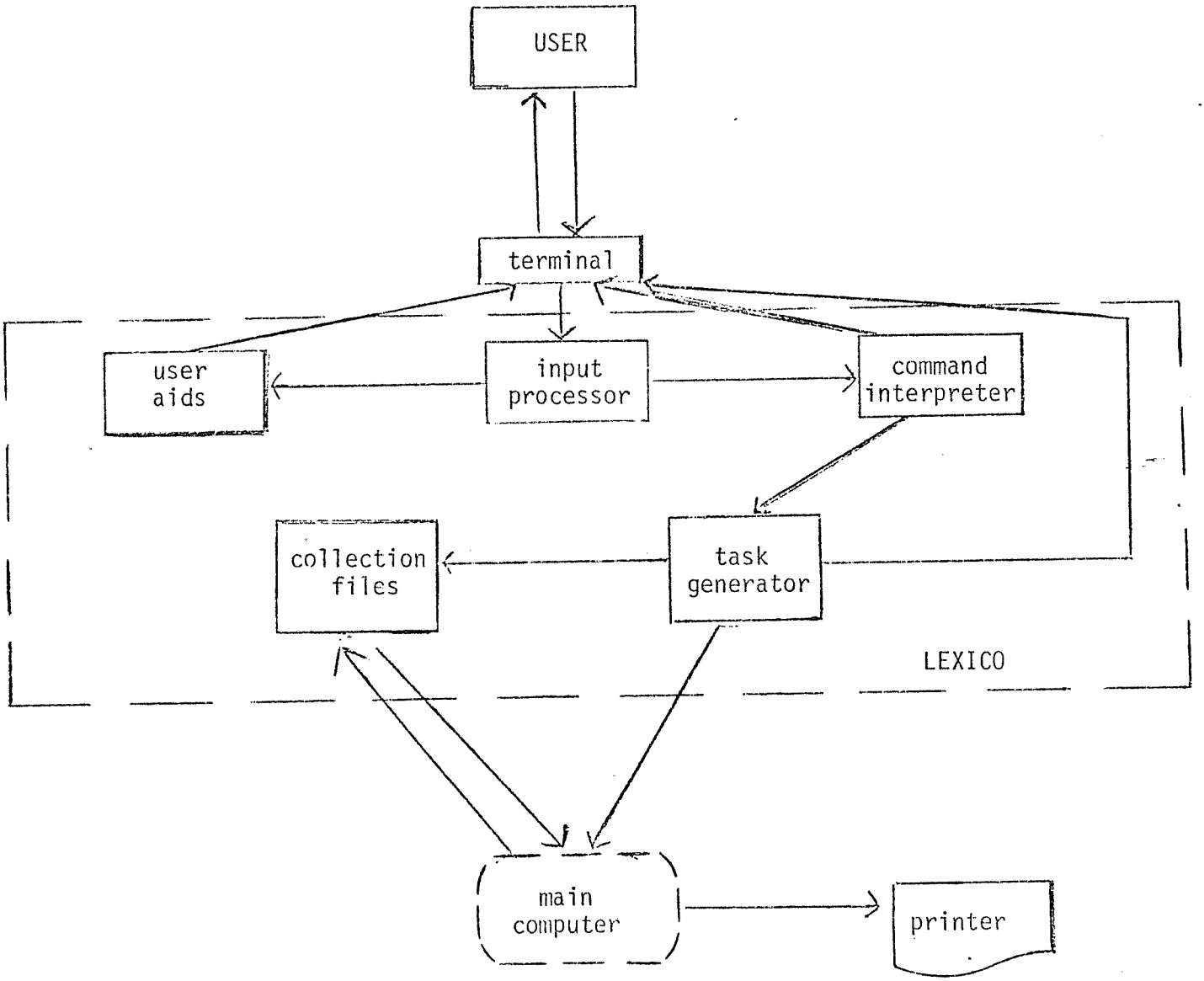


Figure 2

User-system Relationships

TEXT NAME: JOHN

1: 10 JOHN E FRANCESCO SEDEVANNO SUL BANCO [JOHN AND FRANCESCO WERE SITTING ON A BENCH].

2: 20 FRANCESCO LAVORAVA NEGLI UFFICJ DEL BANCO [FRANCESCO WORKED IN THE OFFICES OF THE BANK]; JOHN ERA UN UOMO SENZA PRINCIPJ [JOHN WAS A MAN WITHOUT PRINCIPLES].

3: 30 C'EFA UN BUCO NELLA SCARPA DI JOHN [THERE WAS A HOLE IN JOHN'S SHOE], E COSI' VOLEVA UN NUOVO PAJO DI SCARPE [SOHE WANTED A NEW PAIR OF SHOES]; MA NON AVEVA DENARO [BUT HE DIDN'T HAVE MONEY].

4: 40 JOHN VOLEVA CHE FRANCESCO LO AIUTASSE DI RUBARE IL BANCO [JOHN WNATED FRANCESCO TO HELP HIM ROB THE BANK].

5: 50 FRANCESCO LO HA CONSEGNATO AGLI UFFICII DEL BANCO [FRANCESCO TURNED HIM IN AT THE OFFICES OF THE BANK].

6: 60 MORALE: DECISIONI FATTE SU UN BANCO NON TI FA PADRONE DEL BANCO [MORAL: DECISIONS MADE ON A BENCH DON'T MAKE YOU BOSS OF THE BANK (IT LOSES SOMETHING IN TRANSLATION].

Figure 3

Output from an ADD command

```
1  AGLI     [F:     1]

2  AIUTASSE      [F:     1]

       1   (  1      4-40)     JOHN VOLEVA CHE FRANCESCO LO AIUTASSE DI RUBARE IL BANCO [JOHN WANTED FRANCESCO TO HELP HIM ROB
           THE BANK].

3  AVEVA    [F:     1]

4  BANCO    [F:     6]

       1   (  1      1-10)     JOHN E FRANCESCO SEDEVANNO SUL BANCO [JOHN AND FRANCESCO WERE SITTING ON A BENCH].

       2   (  1      2-20)     FRANCESCO LAVORAVA NEGLI UFFICJ DEL BANCO [FRANCESCO WORKED IN THE OFFICES OF THE BANK]; JOHN E
           RA UN UOMO SENZA PRINCIPJ [JOHN WAS A MAN WITHOUT PRINCIPLES].

       3   (  1      4-40)     JOHN VOLEVA CHE FRANCESCO LO AIUTASSE DI RUBARE IL BANCO [JOHN WANTED FRANCESCO TO HELP HIM ROB
           THE BANK].

       4   (  1      5-50)     FRANCESCO LO HA CONSEGNATO AGLI UFFICII DEL BANCO [FRANCESCO TURNED HIM IN AT THE OFFICES OF TH
           E BANK].

       5   (  1      6-60)     MORALE: DECISIONI FATTE SU UN BANCO NON TI FA PADRONE DEL BANCO [MORAL: DECISIONS MADE ON A BEN
           CH DON'T MAKE YOU BOSS OF THE BANK (IT LOSES SOMETHING IN TRANSLATION)].

       6   (  1      6-60)     MORALE: DECISIONI FATTE SU UN BANCO NON TI FA PADRONE DEL BANCO [MORAL: DECISIONS MADE ON A BEN
           CH DON'T MAKE YOU BOSS OF THE BANK (IT LOSES SOMETHING IN TRANSLATION)].

5  BUCO     [F:     1]

       1   (  1      3-30)     C'ERA UN BUCO NELLA SCARPA DI JOHN [THERE WAS A HOLE IN JOHN'S SHOE], E COSI/ VOLEVA UN NUOVO P
           AJO DI SCARPE [SO HE WANTED A NEW PAIR OF SHOES]; MA NON AVEVA DENARO [BUT HE DIDN'T HAVE MONEY].

6  CHE      [F:     1]

7  CONSEGNATO     [F:     1]

       1   (  1      5-50)     FRANCESCO LO HA CONSEGNATO AGLI UFFICII DEL BANCO [FRANCESCO TURNED HIM IN AT THE OFFICES OF TH
           E BANK].

8  COSI/    [F:     1]

9  C'ERA    [F:     1]

10 DECISIONI      [F:     1]
```

Figure 4

Concordance Output

```
LOOKUP  JOHN (   1)   IN EXAMPLE                    DATE 080876   PAGE 1

   ORIGINAL                RESPELLED          BASE

 1 AGLI                    AGLI

 2 AIUTASSE                AIUTASSE           AIUTARE

 3 AVEVA                   AVEVA

 4 BANCO                   BANCO              *HOMOGRAPH*
BASE 1 : BANCO(=BANK)
BASE 2 : BANCO(=BENCH)
     6 CITATIONS.  NO BASE ASSOCIATIONS EXIST.

 5 BUCO                    BUCO

                           :

16 FA                      FA                 FARE

17 FATTE                   FATTE              FARE

                           :

40 TI                      TI

41 UFFICII                 UFFICII            UFFICIO

42 UFFICJ                  UFFICII            UFFICIO

                           :

45 VOLEVA                  VOLEVA

   40 FORMS REMAIN UNMATCHED
```

no basetype rule exists for "ɤgli".

there is a rule giving the base for "aiutasse".

LEXICO has not yet been told which occurrences of "banco" belong with each base.

words 41 and 42 are both matched by the rule ufficio:ufficii.

The number of words which have no associated bases includes "banco" until each occurrence of "banco" is matched with an individual base.

Figure 5

Output from LOOKUP

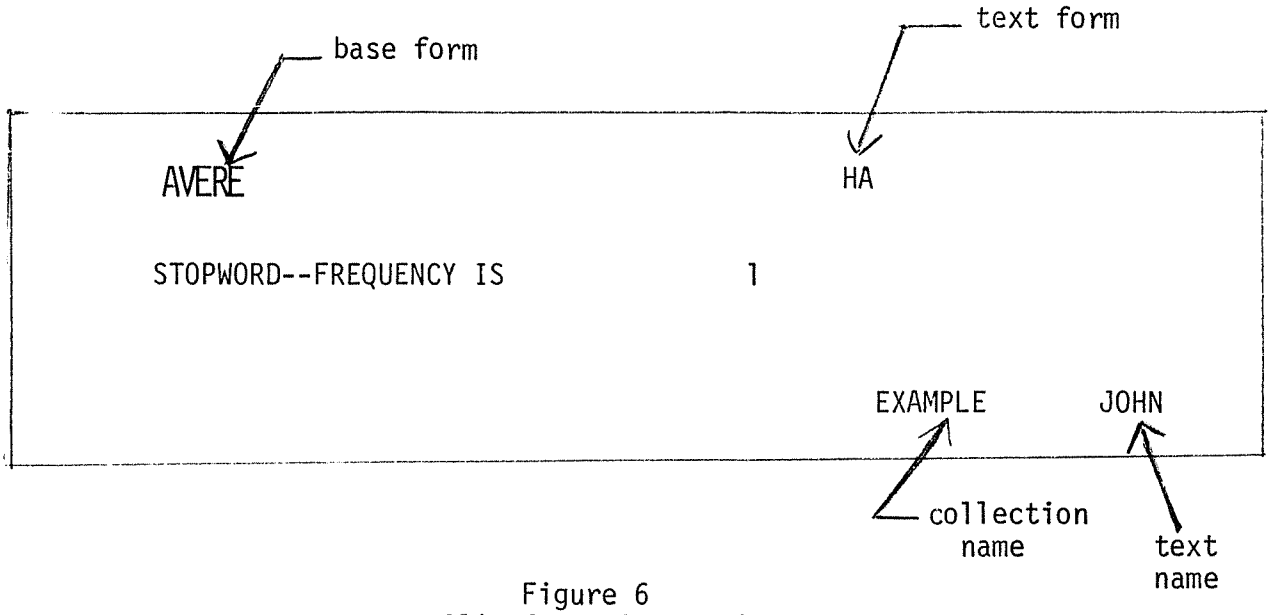


Figure 6
Slip for a Stopword

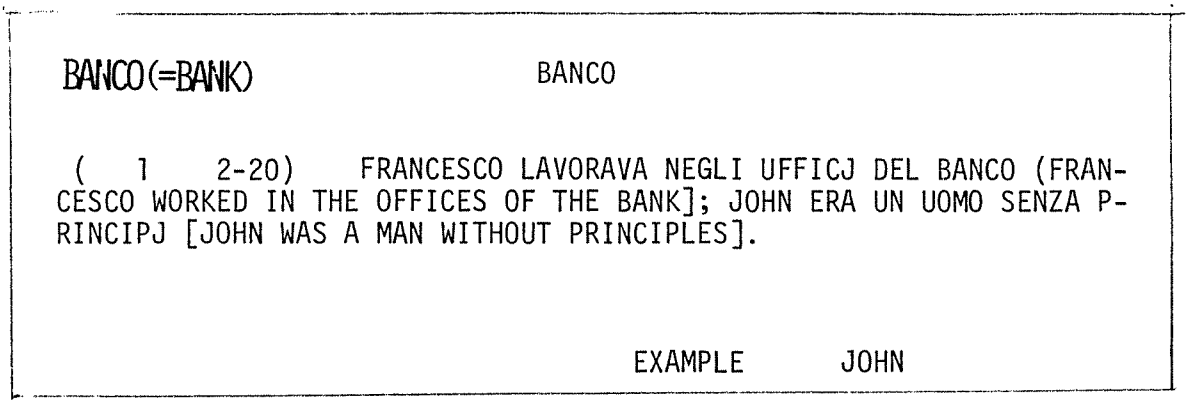


Figure 7

First Slip for a Keyword

BANCO(=BANK) BANCO

(1 4-40) JOHN VOLEVA CHE FRANCESCO LO AIUTASSE DI RUBARE IL BANCO [JOHN WANTED FRANCESCO TO HELP HIM ROB THE BANK].

EXAMPLE JOHN

Figure 8

Second Slip for the Same Keyword

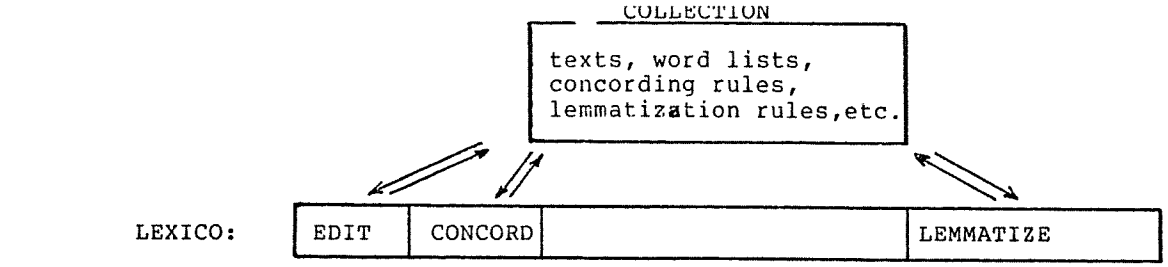
* (1 4-40) encodes text code (1), citation sequence number (4), and citation identifier (40).

```
BASE:      ESSERE    [F:      1] *
RESPELLED: ERA
ORIGINAL:  ERA

BASE:      FARE      [F:      1]
RESPELLED: FA
ORIGINAL:  FA

   1   (  1      6-60)    MORALE: DECISIONI FATTE SU UN BANCO NON TI FA PADRONE
       DEL BANCO [MORAL: DECISIONS MADE ON A BENCH DON'T MAKE YOU BOSS OF THE BANK
       (IT LOSES SOMETHING IN TRANSLATION)].

BASE:      FARE      [F:      1]
RESPELLED: FATTE
ORIGINAL:  FATTE

   1   (  1      6-60     MORALE: DECISIONI FATTE SU UN BANCO NON TI FA PADRONE
       DEL BANCO [MORAL: DECISIONS MADE ON A BENCH DON'T MAKE YOU BOSS OF THE BANK
       (IT LOSES SOMETHING IN TRANSLATION)].
```

Figure 9

Sample from a Base Concordance

* Frequency of occurrence for stopword (no citations given) or keyword (citations given).



(User view)

(Actual implementation;
programmer's view)

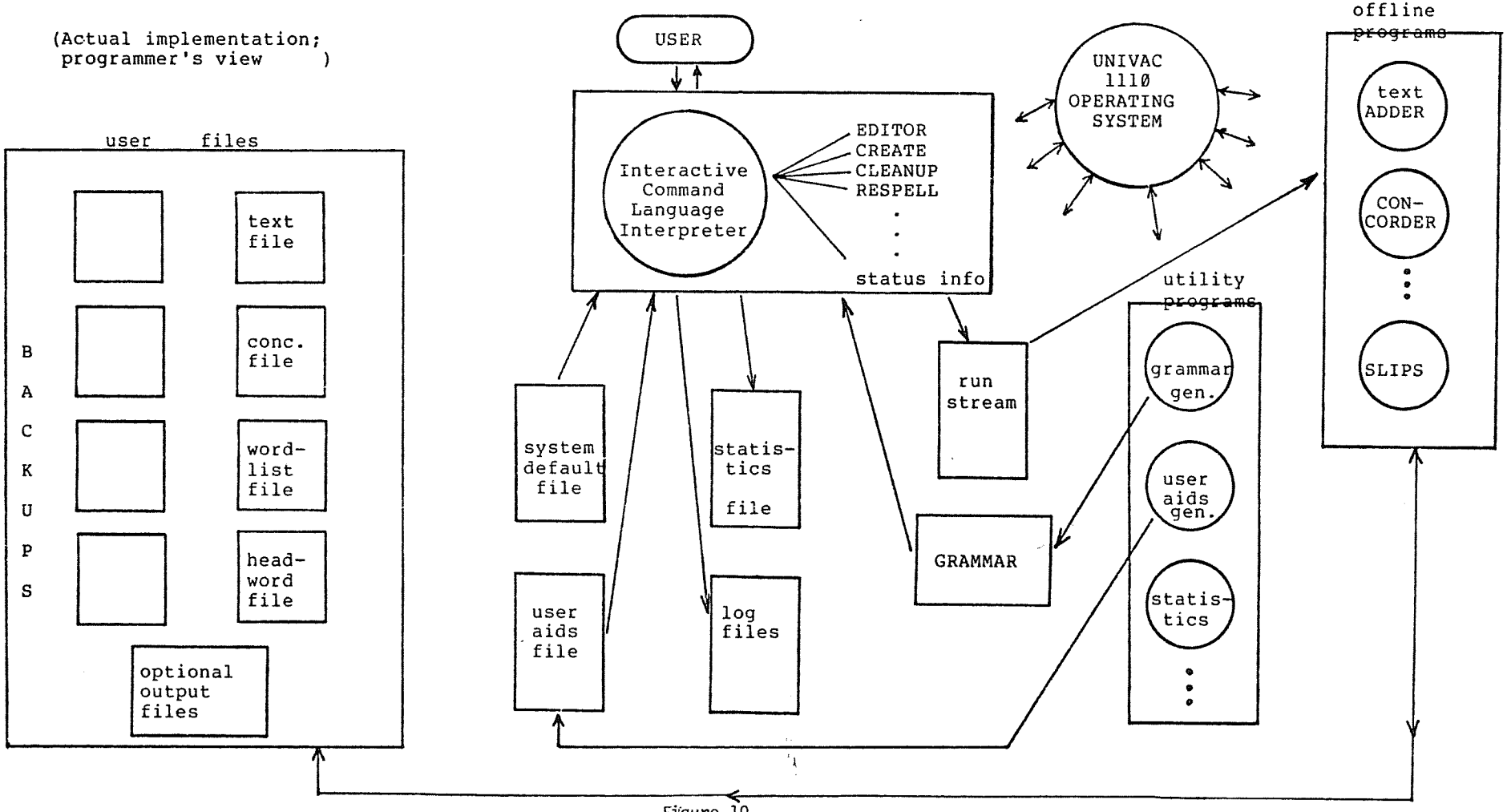


Figure 10
User and Programmer Views of LEXICO

Text preparation

The rest of this section demonstrates the flow of a text through the LEXICO system. The samples illustrate many of the system's capabilities, but do not reflect normal interactive sessions. Typically, a user would remain in a single task and would request fewer user aids. The text used for these examples was written specifically to illustrate some of the capabilities of the system. The first step is to prepare the text for input into a collection. Shown below is a sample text as it exists, prior to processing by LEXICO, on a card file called INPUT TEXT.

```
'JOHN'
(10)  JOHN E FRANCESCO SEDEVANNO SUL BANCO
[JOHN AND FRANCESCO WERE SITTING ON A BENCH].
(20)  FRANCESCO LAVORAVA NEGLI UFFICJ DEL BANCO
[FRANCESCO WORKED IN THE OFFICES OF THE BANK]:
JOHN ERA UN UOMO SENZA PRINCIPJ
[JOHN WAS A MAN WITHOUT PRINCIPLES].
(30  C'EFA UN BUCO NELLA SCARPA DI JOHN
[THERE WAS A HOLE IN JOHN'S SHOE].
E COSI/ VOLEVA UN NUOVO PAJO DI SCARPE
[SO HE WANTED A NEW PAIR OF SHOES];
MA NON AVEVA DENARO [BUT HE DIDN'T HAVE MONEY].
(40)  JOHN VOLEVA CHE FRANCESCO LO AIUTASSE DI RUBARE IL BANCO
[JOHN WNATED FRANCESCO TO HELP HIM ROB THE BANK].
(50)  FRANCESCO LO HA CONSEGNATO AGLI UFFICII DEL BANCO
[FRANCESCO TURNED HIM IN AT THE OFFICES OF THE BANK].
(60)  MORALE:  DECISIONI FATTE SU UN BANCO NON TI FA
PADRONE DEL BANCO [MORAL: DECISIONS MADE ON A BENCH
DON'T MAKE YOU BOSS OF THE BANK (IT LOSES SOMETHING
IN TRANSLATION)].
^ ^
```

The text begins with its title. The body of the text has been divided into citations--in this case, sentences. Each citation has been given an identifier. Here the identifiers are multiples of ten, but the system allows alphabetic or numeric identifiers of up to three levels. Typical identifiers may be page and line numbers or psalm and verse numbers. Notice that the identifiers are enclosed in parentheses. The English translation of the text has been included but is enclosed in square brackets to indicate that the English words are notes to the text which should not be concorded. The text ends with '^^'. (Note that in citation 30 the symbol "/" marks the accent in 'cosi'.)

Several of these symbols, called delimiters, have special meaning to LEXICO. A blank, comma, or semicolon marks the end of a word, a period indicates the end of a sentence, and parentheses, square brackets and '^^' have the uses mentioned above. All of these symbols are standard delimiters, called system defaults. However, a user may select other symbols to be used for these purposes.

Entry of a text into a collection

Once the text has been prepared in this manner, it is ready to be entered into a collection. The user creates a collection called "example" and, with some foresight, specifies stopwords. These are words which will be listed in a concordance only with the number of times they appeared in the concorded text. Words not designated as stopwords are keywords, which are listed in a concordance with each citation in which they occurred. The commands for creating the collection are shown below. (In all examples of dialogues between a user and LEXICO, user input is shown in lower case and is preceded by the symbol '>'. Answers to system questions are indented.)

```
>@le*ico.                                    the user initiates interaction with LEXICO.

 LEXICO VERSION 2.0    08/07/76    12:17:55  LEXICO responds
 COLLECTION NAME?                             and asks for a collection name .

     ->example                                the user enters the collection name .

 NEW COLLECTION MAY BE CREATED.               LEXICO recognizes that the user wants to
                                                work in a new collection .

 TASK COMMAND:                                LEXICO asks for a task to perform

>create;                                      the user wants to create a collection .

 COLLECTION 'EXAMPLE     'TO BE CREATED:      LEXICO acknowledges .
 YOU MAY ENTER COLLECTION DEFAULTS.

>add stopwords del di e era negli nella sul un  the user enters the stopwords; c'era is
> 'c''era' cosi/ ma non aveva che lo il ha      entered as 'c"era' beaause it contains
> agli su ti;                                   an apostrophe.


>end;                                         the user ends the CREATE block .

COLLECTION CREATED.                          LEXICO creates the collection .
```

Having created a collection, the user continues in the same interactive session, causing the text to be added to the collection as shown below.

```
 TASK COMMAND:                               LEXICO asks what to do next .

>add john;                                   the user wants to add a text called "John".

 TEXT CODE ASSIGNED TO 'JOHN       ' IS  1.  from now on the text may be referred to as
                                               "John" or as "1".

>input on card file input.text;              the user tells LEXICO where to find the text.

>end;                                        the user has no further specifications .

 CREATE BACKUP IMMEDIATELY BEFORE THIS       LEXICO asks if a copy of the collection
   PROCESS? (Y OR N)                           should be saved in case the computer goes
                                               down while the text is being added.

       ->n                                   the user does not want a backup .

 WHEN? (I, T, O, W)                          LEXICO asks for a run priority .

       ->*equ                                the user requests an explanation of the
                                               question .
```

```
THE CHARACTER YOU ENTER WILL DETERMINE WHEN
THE RUN WILL BE INITIATED:
I=>IMMEDIATELY    T=>TODAY
O=>OVERNIGHT      W=>WEEKEND
ANYTHING ELSE =>CANCEL RUN
(FOR COST IMPLICATIONS, PRESS EXPLAIN-QUESTION AGAIN)

WHEN? (I, T, O, W)
      ->*equ
NOTE:  I=VERY EXPENSIVE   T=LESS EXPENSIVE
       O=INEXPENSIVE      W=LEAST EXPENSIVE
                            (BUT NOT CHEAP)

       ->o

RUN IDENTIFICATION: XDO848 (SAVERNO807*6122421)

TASK COMMAND:
```

LEXICO explains the options.

the user wants more detail.

the user selects overnight priority.

LEXICO tells the user how to identify the output at the computing center.

LEXICO asks for the next task.

The output which results from these commands is shown in figure 3.

Insert figure 3 about here

Text editing

Now that the text has been added, the user notices it contains some errors. These are corrected as follows.

```
>@le*ico
 LEXICO VERSION 2.0   08/07/76   12:52:26
 COLLECTION NAME?
        ->example

 COLLECTION ACCESSIBLE; PROCEED
```

LEXICO recognizes the collection created above.

```
 TASK COMMAND:
>edit john;
 TEXT 'JOHN        ' (TEXT CODE:     1) AVAILABLE
 CURRENT CITATION IS SEQUENCE NUMBER 1; PROCEED
   TO EDIT.
```

the user wants to edit the text.

```
>*mnu
```

the user requests a display of the editing commands.

```
 THE FOLLOWING COMMANDS OR DECLARATIONS MAY BE
   MADE WITHIN THIS BLOCK:

   END;      (TO END THE BLOCK)
   COST;
   MEMO 'NOTE ABOUT THE TEXT' ;
COMMANDS TO MOVE THROUGH THE TEXT,
COMMANDS TO ADD AND DELETE CITATIONS,
COMMANDS TO MODIFY TEXT,
COMMANDS PERTAINING TO DISPLAY OF TEXT,
COMMANDS ABOUT RELEVANT DELIMITERS.
(MATERIAL IN SQUARE BRACKETS IS OPTIONAL)

COMMANDS TO MOVE THROUGH THE TEXT ARE:

/TOP\ ;      <BOTTOM> ;      <GO TO N> ;
\ T /        \  B  /         \ G N /
                              \  N  /

 NEXT  [N] ;  UP  [N] ;     OFFSET [-] N ;
  N           U

 FOR OTHER ALLOWABLE COMMANDS, PRESS
   EXPLAIN-QUESTION (OR ENTER *EQU )
```

gN, go to N and N, where N is a sequence number of a citation, allow changes to be made to that citation.

LEXICO shows the types of commands and gives some of them, showing alternate forms.

```
>3;
```

the user selects the third citation.

```
 ID:              30
```

LEXICO displays the citation and its identifier.

```
  C'EFA UN BUCO NELLA SCARPA DI JOHN [THERE WAS A HOLE IN JOHN'S SH
 OE], E COSI/ VOLEVA UN NUOVO PAJO DI SCARPE [SO HE WANTED A NEW PA
 IR OF SHOES]; MA NON AVENA DENARO [BUT HE DIDN'T HAVE MONEY].

>change f ro r;
```

the user attempts to correct the spelling of "c'era", but enters "ro" instead of "to".

```
 UNRECOGNIZED KEYWORD OR PUNCTUATION: F
>*err

 I WAS EXPECTING A KEYWORD OR PUNCTUATION IN PLACE
   OF THE ABOVE STRING
 YOUR ERROR IS DUE TO:
    1.  A MISSPELLED KEYWORD.
    2.  A COMMAND NOT ALLOWED IN THIS BLOCK,
    3.  A WORD THAT ISN'T A KEYWORD, OR
    4.  INCORRECT OR OMITTED PUNCTUATION.

 PRESS EXPLAIN-ERROR FOR CORRECTIVE MEASURES,
   EXAMPLE-KEY FOR EXAMPLES.
>f to r;

  C'ERA UN BUCO NELLA SCARPA DI JOHN [THERE WAS
 A HOLE IN JOHN'S SHOE], E COSI/ VOLEVA UN NUOVO
 PAJO DI SCARPE [SO HE WANTED A NEW PIIR OF SHOES];
 MA NON AVEVA DENARO [BUT HE DIDN'T HAVE MONEY].

>Next;

 ID;                  40
  JOHN VOLEVA CHE FRANCESCO LO  AIUTASSE DI RUBARE
 IL BANCO [JOHN WNATED FRANCESCO TO HELP HIM ROB
 THE BANK].

>c na to an;


  JOHN VOLEVA CHE FRANCESCO LO AIUTASSE DI RUBARE
 IL BANCO [JOHN WANTED FRANCESCO TO HELP HIM ROB
 THE BANK].
>end;
```

LEXICO does not understand the previous input.

The user wants the error message explained.

the keyword "to" was misspelled.

beginning with the symbol where the error occurred, the command is reentered.

the corrected citation is displayed.

the user continues to the next citation.

the citation is displayed.

the spelling of "wanted" is corrected, using "c" as an abbreviation for "change'

all corrections are made so the block is ended.

Concording

The corrected text is ready for concording. In the following exchange, the user schedules the concordance.

```
>concord 1;                                          The user refers to the text by text code
                                                       instead of text name.
 TEXT'JOHN          '     (TEXT CODE:     1)AVAILABLE:
 YOU MAY ENTER CONCORDANCE SPECIFICATIONS.
>end;
 CREATE BACKUP IMMEDIATELY BEFORE THIS PROCESS? ( Y OR N)

     -> n                                            the user does not want a backup.
 WHEN? (I, T, O, W)

     -> o                                            he selects the overnight priority.
 RUN IDENTIFICATION:  XDO851     (SAVERNO807*6125642)
```

For this text, it is unnecessary to enter other commands within the CONCORD block. However, at this point the user may specify, for example, that a set of delimiters is to be used for this text which differs from those used for other texts in the collection. Figure 4 shows part of the concordance generated as a result of this block. It should be mentioned that for texts that do not require editing, there is an ADD-CONCORD block which allows a text to be added and concorded in one run. It is also possible to add, or to add and concord, several texts at one time.

Insert figure 4 about here

Spelling Conversion

One result of concording a text is to store in the collection an alphabetized list of all the keywords and stopwords which appeared in the text. Headword classification is the process of associating a base form or headword with each entry in this list. When this has been completed, slips may be generated using this list and the concordance. An optional step in the headword classification process is the standardization of spelling. When applying spelling rules of the form new:old, LEXICO replaces every occurrence of the string "old" with "new" in the word list of a text. No change is made in the body of the text. Blanks are used in spelling rules to indicate that the rules apply only at the beginning or end of words. Spelling rules which convert word-final "j" to "ii" and every other occurrence of "j" to "i", and the application of these rules to the word list of the sample text, are shown below.

```
 TASK COMMAND:
>update;
```

the spelling rules are entered in an UPDATE block.

```
 COLLECTION 'EXAMPLE     ' TO BE UPDATED;
 YOU MAY ENTER COLLECTION DEFAULTS.
>add spelling rules ii:'j ', i:j;
>end;

 TASK COMMAND:
>respell 1;
```

the user requests the rules to be applied to the word list.

```
 HERE AND NOW? (Y OR N)
```

LEXICO asks if this should be done on-line or off-line.

```
      -> *equ
```

the user requests an explanation.

```
 YOU CAN CAUSE A WORD LIST TO BE RESPELLED
 IMMEDIATELY (HERE AND NOW) OR AT A LATER TIME,
 AT THE COMPUTING CENTER, AT A REDUCED COST.  IF
 YOU WANT RESPELLING TO TAKE PLACE HERE AND NOW,
 ENTER Y (IN COLUMN 1).  OTHERWISE, ENTER N.
 HERE AND NOW?  (Y OR N)

      -> y
 WANT A LISTING? (Y OR N)
```

the user wants the RESPELL done on-line.

```
      -> y
```

the user wants the original and respelled forms to be displayed.

```
 CURRENT SPELLING RULES ARE:
     'I'      :  'J'
     'II'     :  'J '

 (WORDS THAT ARE CHANGED ARE MARKED WITH ***)

     1 AGLI                          AGLI

     2 AIUTASSE                      AIUTASSE

                       :
    20 IL                            IL

    21 JOHN                          IOHN
   ***

    22 LAVORAVA                      LAVORAVA
                       :
    32 PAJO                          PAIO
   ***
                       :
    41 UFFICII                       UFFICII
```

the system first displays the spelling rules and then shows the word list.

```
    42 UFFICJ                       UFFICII
   ***
                    :
    45 VOLEVA                       VOLEVA
 RESPELLING COMPLETED.
 TASK COMMAND:
>cleanup 1;                                    the CLEANUP block may be used to correct an
                                                 exception to the spelling rules.
 TEXT 'JOHN    '      (TEXT CODE:    1) AVAILABLE:
 PROCEED WITH CLEANUP.
>type 21;                                      the user selects the word to correct.
    21   JOHN                       IOHN       LEXICO displays the entry in the word list.
>new respelled john;                           the user enters the correction.
    21   JOHN                       JOHN       the change is displayed.
>end;                                          the user ends the block.
```

Since headwords are assigned to respelled forms rather than text forms, respelling can greatly reduce the number of bases which must be specified for texts which have many variant spellings.

Lemmatization

A base form may be associated with each entry in the word list of a text in either of two ways. Previously defined basetype rules may be applied to the word list in an offline process called LOOKUP, or the bases may be entered explicitly in a CLEANUP block. Basetype rules may be entered as follows:

```
>add basetype rules                            in basetype rules, the base precedes
                                                 the colon.
   ufficio : ufficii,
   fare : fa fatte,                            basetype pairs are separated by
   aiutare : aiutasse,                           a comma; a semicolon ends
   'nel gli ' : negli;                           the entire list.
>add homograph rule banco  :banco(=bank)
  banco(=bench);                               homograph rules permit more than one
                                                 base to be associated with a word:
                                                   in these the bases follow the
                                                     colon.
```

Figure 5 shows the output of a LOOKUP performed on the sample text using the basetype rules entered above.

Insert figure 5 about here

The CLEANUP block may be used to enter bases in the word list and to associate citations with each occurrence of the homograph. In the following sequence, the SHOW UNMATCHED command is used to have LEXICO display each entry that has no base.

```
>show unmatched;
 ENTER BASE FOR TYPE       1 AGLI
 RESPELLED FORM IS AGLI

      ->*equ
 ENTER BASE (WITHOUT ';') OR '?' OR '='
   OR 'H' or 'R' or ';'.

      ->*equ


 ? LEAVES TYPE UNMATCHED
 = ENTERS A BASE WHICH IS THE SAME AS THE
 RESPELLED TYPE, IF ANY EXISTS, AND OTHERWISE,
 IS THE SAME AS THE ORIGINAL TYPE
 H ALLOWS YOU TO ENTER HOMOGRAPHS
 R REJECTS THE TYPE
 ; TERMINATES THE`SHOW UNMATCHED; PROCESS.
 (FURTHER EXPLANATION AVAILABLE).

      ->A GLI


BASE IS A GLI

'A GLI' : 'AGLI'
 ADD BASETYPE  RULE? (Y OR N)
      ->y


 ENTER BASE FOR TYPE       3  AVEVA

 RESPELLED FORM IS AVEVA
      ->avere
 BASE IS AVERE
 'AVERE' : 'AVEVA'
 ADD BASETYPE RULE? (Y OR N)
      ->y

     4 BANCO        BANCO         *HOMOGRAPH*


 BASE 1 : BANCO(=BANK)
 BASE 2 : BANCO(=BENCH)
      6 CITATIONS.  NO BASE ASSOCIATIONS EXIST.
 ENTER BASE 3

      ->;

 ENTER BASE NUMBER OR ;

      ->1

ENTER SEQUENCE NUMBERS FOR BASE 1 BANCO(=BANK)
  (TERMINATE WITH ;)
       ->2-4 6;
```

the show unmatched process begins.

the user asks for an explanation
the system displays the code which may be entered.

the user asks what the codes mean.

the system explains.

the user enters the base for the first word.

LEXICO formulates the corresponding base-type rule and asks if it should be saved in the collection for use in future LOOKUPs (if other texts are processed).

the second word, "aiutasse" was assigned a base during LOOKUP.

the fourth word is a homograph, LEXICO displays the the information it has, and asks for additional bases.

the user does not want to add other bases.

he wants to associate citations with the first base.

the second, third, fourth and sixth occurrences of "banco" in the text belong with the first base.

```
ENTER BASE NUMBER OR ;
     -> 2
ENTER SEQUENCE NUMBERS FOR BASE 2 BANCO(=BENCH)

(TERMINATE WITH ;)

     -> rest;

      4 BANCO          BANCO    *HOMOGRAPH*


BASE 1 : BANCO(=BANK)
  CITATIONS     2     3    4    6
 BASE 2 : BANCO(=BENCH)
  CITATIONS:    1     5
 ENTER BASE FOR TYPE    5   BUCO
 RESPELLED FORM IS BUCO

     -> =


 BASE IS BUCO
 'BUCO' : 'BUCO'
 ADD BASETYPE RULE? (Y OR N)
     -> y
 ENTER BASE FOR TYPE      6   CHE
                  :
                  .
```

all other occurrences of the word in the text are matched with the second base.

when all citations are matched, LEXICO displays the results and continues to the next word.

= is the code for entering a base identical to the respelled form.

As the headword classification process is completed, LEXICO will output information in any of several forms. Figures 6, 7, and 8 show slip output. Each slip is printed on a separate page and all the slips are alphabetized by the spelling of the base form. Figure 9 show part of a base concordance.

— — — — —

Insert figures 6-9 here

— — — — —

Summary

The user sees a single system centered around a text collection. Communication is established through an on-line terminal, requests are made, and results are observed either immediately on the terminal or at a later time as output from the computer printer. Requests for all tasks are made in the same form, viz., a combination of command statements and replies to system prompts.

When uncertain, puzzled, or completely unsure, the user can request help from LEXICO, either in the form of a list of legal statements for the current block (MENU) or as an explanation of a specific action (EXPLAIN ERROR, EXPLAIN QUESTION, EXAMPLE).

6. The Programmer's View

Although the user's view of LEXICO consists of a single system that operates on a collection of texts, LEXICO is in fact a complex of many programs and files that interact with one another. These components fall into at least five categories:

1) files representing the user's data,
2) system files used to support LEXICO functions,
3) an interactive program that interprets user commands and responds accordingly,
4) several off-line programs that perform user-requested tasks, and
5) programs used to generate or interpret system files.

The ability of LEXICO to isolate a user from its own underlying complexity is illustrated by the two views of the system shown in Figure 10.

Insert Figure 10 about here

LEXICO was designed both for a specific local environment and a more diffuse expanded environment. The local environment was a UNIVAC 1108 which was updated two years ago to a dual processor 1110 with .5 million characters of core memory and almost 1.5 billion characters of random access, secondary storage. At any time, up to 72 remote sites (keyboard terminals and Remote Job Entry stations) may have simultaneous access to the 1110. At the time this project began the system software for this machine included a variety of high level and assembly languages, but the only high level language practical for system development at that time was FORTRAN.

Operating system capabilities, although considerable expanded now, were quite limited when the 1110 was first installed. For example, random access to disk files was possible only through non-supported I/O routines and user-oriented aids such as message files and HELP capabilities were not available.

The expanded environment included users from outside the University of Wisconsin campus who would use LEXICO either by remote access, or by sending data initially processed at a local site, or by processing LEXICO output on their own systems. Therefore, LEXICO needed to facilitate the entry of data files produced on machines other than the UNIVAC 1110 and had to produce output which would be readily compatible with other major systems, including a microfiche processor.

For reasons primarily of convenience, the system was developed in FORTRAN on the UNIVAC 1110 and was not written to be especially transportable.

This latter decision involved a switch from initial intentions. However, we felt that an experimental system that would change frequently should not be distibuted to other users, hence the added cost of generality could not be justified.

A further decision, which we have now begun to question, at least in part, was to isolate the user from the various processors on the UNIVAC 1110. We might have developed LEXICO in about half the time it has required so far by utilizing several standard UNIVAC systems. But this would have required the user to learn not only the procedures unique to LEXICO, but also how to use the UNIVAC sort package, the text editor, and a variety of other systems which were written by different programmers and which utilize widely differing control languages, error handling, and documentation. LEXICO was designed to give the user a single, integrated system with consistent control commands, input and output formats, and error handling. This goal we feel is still desirable. The degree to which it can be realized with standard software components is something we are now re-evaluating (see Section 7).

The current system consists of 18 separately executable subsystems (programs). These are composed of 275 distinct functional modules (subroutines). The entire system contains almost 27,000 lines of code, 95% of which are FORTRAN, the remainder assembly language; this is equivalent to almost 100,000 machine instructions.

Components of the LEXICO System

All of the user's work centers on a collection, which LEXICO maintains internally as four separate files: a text file, a concordance file, a word list file, and a headword file. The text file contains 1) collection default values, 2) a directory of all texts entered in the collection, 3) the body of each text, 4) a table of identifiers associated with each text, and 5) text default values. The concordance file contains a directory and concordances (corresponding to texts in the text file). The word list file contains (for each concorded text) each keyword and stopword, its respelled form (if any), and its base. The headword file contains headword classification (basetype) rules.

The LEXICO command language is represented by a grammar, which specifies allowable LEXICO commands and (indirectly) their resulting interpretation. This grammar is coded in a concise format as shown below.

```
 STATEMENT  :   %SHOW    %DIR
       78
%SHOW   :    SHOW
%SHOW   :    S

%DIR  : DIRECTORY
%DIR  : DIR
```

This example indicates that any of the commands

```
S    DIR  ;
SHOW DIR  ;
S    DIRECTORY ;
SHOW DIRECTORY ;
```

are to cause action number 78 to be performed. A LEXICO support program produces from this grammar a condensed form, which is used at run time to interpret user commands.

During interaction with a user, all system routines must provide user aids to explain prompts, error messages, and user options. All such messages must be specified on a system file and are therefore treated consistently by the interactive system. Messages are specified at increasing levels of detail, each level being displayed at the user's request. An example of such a multi-level message (as specified by the programmer) is:

```
ESEQ13      ENTER CITATION:
ESEQ14      ENTER A NEW CITATION. USE AS
            MANY LINES AS NECESSARY; TRAILING
            BLANKS ARE IGNORED. TERMINATE
            WITH A CITATION DELIMITER.
ESEQ15      SEE USER GUIDE 5, SECTION 1.3;
```

The Command Language Interpreter operates in conjunction with the grammar and the User Aids file to communicate with the user. All collection-related data is obtained from (and updated on) appropriate files of the collection. Tasks that are scheduled for later processing take the form of runstreams stored on system-generated files.

The operations performed by LEXICO to concord a text illustrate the interpretation of user requests and resulting file management. When a user enters a CONCORD statement, LEXICO retrieves from the text file all pertinent defaults and searches the directory to ensure that the requested text exists. If this is the first concordance to be generated in the collection, the concordance file and word list file are created and initialized. When all declarations have been entered, followed by an END statement, LEXICO creates a system file to contain a runstream. Appropriate commands and data are entered in the file and the user is prompted for a scheduling priority, which is translated into a request to the operating system to schedule the task.

The text directory (and possibly text defaults) are updated to reflect scheduling of the concordance. Later, when the operating system begins the actual task, an offline component of LEXICO ensures that the collection (and text) still exist, and then begins to read citations from the text file. In addition to several intermediate system files (e.g., for sorting), the system generates a concordance in the concordance file and a new list of words in the word list file (updating directories in each file). The collection directory is updated and the concordance is output according to the user's request (i.e., on paper, magnetic tape, or mass storage). Finally, information about the task is logged on the system's statistics file for later analysis.

Programmer Aids

Several features of LEXICO were developed to simplify debugging and modification and to encourage more consistent, reliable programs. These were:

1) the ability to use LEXICO commands to perform system maintenance functions (e.g., updating system defaults, condensing the grammar);
2) diagnostic features selectable by entering commands during interaction;
3) the requirement that all communication with the user be established through user aids;
4) a set of utility routines for common collection-related operations.

7. Evaluation

The primary goal of harmonious and effective man-machine integration was attempted in LEXICO through a variety of software mechanisms: a single integrated system which shields the user from disparate computer components; a terminal-independent communication language; operational aids for correcting errors and reducing costs; learning aids; and special routines for assisting the programmer in diagnosing software faults and in planning system improvements. The contribution of each of these mechanisms towards the primary design goal can not yet be evaluated quantitatively; nevertheless, after approximately 600 man-hours of system usage, subjective assessment can be offered. In the sections which follow, the value of each mechanism mentioned above is discussed.

A Single, Integrated System

Even though LEXICO was developed over a five-year period by a team of programmers, the outward appearance of a single, integrated system has been achieved. A single communication structure is used throughout; aids, such as MENU, are available regardless of task; and error handling is the same throughout the system. Users are not required to learn protocols for independent systems. But the price of achieving this goal in cost and time has been exceedingly high. For example, a sorting routine was programmed into LEXICO even though a general sort system was available on the Univac 1110. Similarly, general 1110 systems for file handling and text editing were ignored in favor of our own versions of these.

Where Univac 1110 systems had to be utilized, such as in initiating running programs, elaborate procedures were implemented to isolate the user from the 1110 protocols. A smaller system utilizing existing 1110 routines might have been developed in one-half the time which LEXICO required, yet the user would have paid for this in increased complexity and increased learning time. If the project were initiated tomorrow, some, but not very many 1110 systems would probably be utilized.

Terminal-independent Communication Language

It was originally planned to allow a user to interact with LEXICO in a command mode, a forms mode, and a prompt mode. The forms mode was to have consisted of a series of displays, each showing some parameters of the system. The user would simply type over the values he wanted to change. This mode was discarded because the implementation would have been so dependent on a specific type of terminal that the class of potential users of the system would have been severely reduced. In the prompt mode, the user would select a task and LEXICO would ask him all relevant questions. Instead of permitting the user to alternate between command language and prompt mode, LEXICO itself decides when each is to be used. For example, the SHOW UNMATCHED process described in section 5 involves a temporary transition to prompt mode. Thus, instead of offering three communication options, with high terminal dependence, two modes are employed, but at LEXICO's option, with little terminal dependence.

The possibility of users selecting prompt or command mode was considered, and components of each actually designed. However,

a full prompt mode was eventually seen as a crutch for the novice which might actually inhibit the learning of the more efficient command mode. In addition, documentation for two independent modes was found to be clumsy. Hence, we decided to have the system determine when commands are required and when prompts are given. This solution appears to be satisfactory, especially with the command abbreviations which users tend to use as they become more familiar with the system. Users have, in fact, requested more extensive use of abbreviations.

Operational Aids

The user aids described in section 5 (explain error, explain question, etc.) tend to be used by both experienced and inexperienced users. However, writing explicit but complete messages proved to be exceedingly challenging. With almost 1300 lines of messages in LEXICO, obtaining sufficient user reactions to each message so that acceptable wordings can be empirically derived will probably require a millenium of continual user experience to achieve. Common sense and a proper appreciation for the simple and direct are welcomed allies in this task.

Assisting the user in minimizing costs has been moderately successful, yet has been limited by changes in the billing algorithms adopted by the University computing center. The original design of LEXICO was predicated on the availability of inexpensive disk packs and on cheap central memory usage compared to I/Ø usage. However, user-owned disk packs have yet to be implemented and central memory usage is now billed at a considerably higher rate relative to I/Ø usage.

Costs

The cost of performing different tasks is dependent on the priority at which off-line jobs are run and the time of day at which on-line interaction takes place. Since the system is still being modified, cost estimates can not be extremely accurate. For example, a recent change reduced the charge for initiating interaction with LEXICO from about $1.00 to $.43 during the day, or from $.35 to $.17 at night. Table 1 gives the cost of some tasks performed by the system in the last few months. All processing was billed at the least expensive rate. Table 2 gives some typical file charges.[8]

Insert Tables 1 and 2 about here

Learning Aids

The primary learning aids for LEXICO users are a series of seven user guides: an overview, and a guide for each of the major processes. In addition, a special guide describes features for system maintenance. These guides replaced a single, thick reference manual whose bulk was a psychological barrier to several prospective users.

The multiple guide approach has been well accepted and will be continued. Two problems remain to be resolved, however. The first, which will soon resolve itself, is the difficulty of maintaining up-to-date documentation on a changing system. Until recently, a guide more than a week old was an outdated guide. The second problem is one inherent in all software documentation, and that is the conflict created by attempting to make a single manual serve as both a reference manual, which is intended for experienced users, and a tutorial manual which is written for beginners. Expensive systems can afford both; for LEXICO, a compromise was adopted whereby explanations and examples were directed towards beginners, but format and summaries were directed towards non-beginners.

Programmer Aids

In manufacturing it is inefficient to develop a process just for producing a product. To be efficient, the process must also provide information for its own improvement. From this standpoint, LEXICO has been highly efficient. The programmer aids for diagnosing software faults and for gathering statistics on user and system performance have been continually utilized. Their implementation absorbed a major portion of the total development cost of LEXICO, but their utility has more than justified their expense.

8. Conclusions

There are several levels of success which software systems can attain. The lowest level, but one achieved by remarkably few projects, is to have a working system reasonably close to the projected completion date and costing not too much more than what was originally allocated.

The second level of success is achieved when documentation is complete, accurate, and readable; the system can be used without assistance from the programmers; and the system is being properly maintained. Finally, the third and highest level is attained when the system is found to be easy to use and cost-efficient.

With some stretching of the semantic range of 'reasonable', LEXICO can be classed as a success at the first level. On the second level, LEXICO is not too far from the stated criteria. Documentation is now complete, seemingly accurate, and readable, at least to those users who have commented on it. The University of Wisconsin Academic Computing Center is now offering LEXICO as a Center system and has taken over responsibility for system maintenance. So long as the system earns revenue for the Center, it will have proper maintenance. Finally, no major errors are known to exist in the system, even though improvements in communications, particularly in error messages, and in efficiency of file handling are still being done.

The degree of success at the third level, as discussed in the preceding section, is somewhat more difficult to gauge, especially without a larger user population. In time, success will be determined by the popularity of the system. Perhaps a more stingent test will be whether succeeding user-oriented systems adopt any of the man-machine techniques employed by LEXICO.

Footnotes

1. The initial work on LEXICO was supported by Grant GJ-32764 from the National Science Foundation. Subsequent work has been supported by the Graduate Research Committee at the University of Wisconsin, by the Foundation for Education and Social Development, and by the Canada Council through a grant to the Medieval Centre at the University of Toronto.

2. The Dictionary of Old English is described in A. Cameron and R. Frank (eds.), A Plan for the Dictionary of Old English. Toronto: University of Toronto Press, 1973.

3. See, for example, the papers on computer processing published in A. Cameron, R. Frank, and J. Leyerle (eds.), Concordances and Old English Texts. Toronto: University of Toronto Press, 1970.

4. Academic dictionaries is intended here to designate historical dictionaries and other works of primarily scholarly interest from dictionaries produced for a mass audience. Admittedly, such a distinction cannot be applied rigorously, in that some commercial dictionaries are as scholarly as any academically produced work.

5. The figures cited here are drawn from the introduction to Volume I of The Oxford English Dictionary, pp. XIVff.

6. A general approach to these functions is now being implemented at the University of Montreal (Bratley and Lusignan, 1975).

7. A slip is a listing of a base form with its associated text form and a single context in which the text form occurs. Generally, slips are printed in a manner that allows them to be filed in an index card box for hand processing.

8. Since space limitations do not allow a full description of the University of Wisconsin Computing Center charges and of the degree to which computing is subsidized by the University, these figures should be interpreted with caution.

References

Baker, F. T., and H. D. Mills, "Chief programmer teams," Datamation, 1973, 12, 58-61.

Ben-Hayyim, Z., "A Hebrew dictionary on historical principles," Ariel, 1966, 13, 14-20.

Bratley, P., and S. Lusignan, "Information processing in dictionary making: some technical guidelines, "Publication No. 196, Department d'Informatique, Université de Montréal [n.d.].

Busa, R., "An inventory of fifteen million words," in IBM Literary Data Processing Conference Proceedings September 9-11, 1964, NY:MLA, 1964, pp. 64-78.

Hagerty, P. E., "The University of Maryland DUM (Demand User's Monitor) System," Proceedings of the USE Conference, Fall, 1972, pp. 29-80.

Mills, R. G., "Man-machine communication and problem solving," in Annual Review of Information Science and Technology vol. 2, C. A. Cuadra (ed.), 1967.

Price, J. D., "An algorithm for analyzing Hebrew words," Computer Studies in the Humanities and Verbal Behavior, 1969, 2, 137-65.

Venezky, R. L., "Storage, retrieval, and editing of information for a dictionary," American Documentation, 1968, 19, 71-79.

Zampolli, A., "Intervento sul tema "Il dizionario italiano di macchina," Calcolo, 1968, 5 (suppl. n. 2), 109-126.

Process*	Total Cost	Cost of Printed Output
add and concord a text of 3500 words	$8.04	$4.65
generate a base concordance for 1000 words comprising a text of 1100 words	$1.64	$1.20
RESPELL a word list with 1,083 entries (without listing the word list)		
on-line	$ .31	--
off-line	$ .12	--
LOOKUP a word list with 1,083 entries	$1.00	$ .68
1 1/2 hours of on-line interaction	$7.88	--

*planned system modification should significantly reduce some of these charges

Table 1

Sample Costs for Using LEXICO

Data	Daily File Charge
A collection with 6 concorded texts totalling over 14,000 words. No basetype rules; concordances stored on tape.	$1.21
The concordances for the above collection (if stored in the collection rather than on tape).	$3.44
A file of about 8600 basetype rules.	$ .79

Table 2

Typical Daily File Charges