AUTONOTE: A PERSONAL INFORMATION STORAGE AND RETRIEVAL SYSTEM

Walter Reitman, R. Bruce Roberts, Richard W. Sauvain, Daniel D. Wheeler, and William Linn
The University of Michigan
Ann Arbor, Michigan

Summary

This paper describes AUTONOTE, a personal storage and retrieval system designed for use by individuals working with large bodies of information. The user may enter a variety of textual materials and assign descriptors and phrases by which these materials may be retrieved. He has available mechanisms for deleting, replacing, linking, and hierarchically organizing text items. The system is operating on-line in a time-sharing environment, and can be utilized from a variety of terminals. Both use and implementation are discussed in detail, with special attention to utilization of AUTONOTE through an alphanumeric CRT display. Also mentioned are potential artificial intelligence extensions and the use of the system in a study of scientific problem solving.

Introduction

Because of their own limited ability to process information, men often regard omniscience as a defining property of deity. Only God always knows where He is. Only God can access all the relevant facts. Only God always is in context.

As the amount of information we must cope with continues to increase, the problems of organizing it, staying in context, and insuring the availability of relevant materials become a major theoretical and practical concern. These problems are particularly acute in such areas as the social and behavioral sciences where, as the recent National Research Council report[1] on Communication Systems and Resources in the Behavioral Sciences points out, scientists must work in complex, diffusely structured informational contexts. The bodies of knowledge they have available

> "...change rather rapidly and lack general and widely accepted definitions and principles. A large part of knowledge in the social and behavioral sciences is an assortment of empirical generalizations, hypotheses, hunches, descriptive phrases, uninterpreted sentences, and the like....
>
> "Future behavioral science theories are unlikely to consist simply of a few touchstone principles. ...[They] will evolve out of conscious syntheses of thought and research, with repeated reorganization of past conceptualizations and findings...."

The AUTONOTE (Automatic Notebook) system described below is a computer based facility meant to serve both as a practical means of dealing with information organization problems and also as a research tool for investigating these problems in an interactive context. The description that follows outlines the main features of the present running system: basic design goals; the principle features of AUTONOTE operations; and the hardware and software used.

Long Range Goals

The basic goal is to develop in stages an increasingly intelligent on-line computer support system for individual users or for small groups working with textual information. This system should allow the user to store, index, organize, and retrieve information. We anticipate that subsequent versions of the system will use information about the user, the subject matter, and the current working context, and make plausible inferences about the user's wants and the best ways of filling them. In sum, the long-range goal is a trainable system, a system that can change its strategies, make appropriate inferences, and in general improve the speed, efficiency, and utility of communication as man and system increasingly "understand" one another better.

General Properties of the Data Base

In contrast to typical document retrieval systems, in which the user is retrieving material generated by others, the AUTONOTE data base is generated and organized by the user himself. The basic units of material stored in AUTONOTE are items; these normally, though not necessarily, are of roughly paragraph length.*

The user may store any kind of textual material, in whatever format he finds convenient. Typical materials users enter include notes, bibliographic references, drafts of papers, ideas for further research, hypotheses, illustrations and paradigms, and transcripts or synopses of conversations with other scientists. In short, users enter those things they could not remember in entirety, but which may be of use at some later date.

Practical Considerations

AUTONOTE is meant to be a practical tool even for the man who cannot remember what he was

*The size of an item may range from one to over 100,000 characters.

working on yesterday, where he put what he wrote the day before, and why he is working on the problem before him today. Since many such people have neither extensive prior experience with computers not substantial funds to pay for them, the system is designed to be easy to learn and cheap to run.

As explained below, the novice can begin with a very simple version of the system. Then, as he acquires experience, he can promote himself out of the novice class and gain access to a broad range of options which make it possible for him to specialize the system to suit his particular needs and working habits. The system minimizes users' housekeeping problems. It permits deletion of low value information at will. It accepts interleaved commands, thus enabling the user to halt work on one problem temporarily, begin work on some other problem, and later come back if he chooses.

Economy of operation, at least with respect to present implementation, is achieved by taking advantage of available file handling routines and by keeping the basic program small so as to take a minimum of space in core. Since it does not require much central processor capacity, it is well suited for use in an on-line time-sharing mode.

Finally, to facilitate employment by a variety of users, AUTONOTE is designed to operate with several different terminal devices, as described below in the section on hardware.

The Use of AUTONOTE

Having established these design requirements, let us look briefly at the essential operations of AUTONOTE, with the understanding that these comprise only a small portion of the ultimate services with which a user can be supplied. Once this ground work has been completed, we will look at the extended modes of operation available to a user of AUTONOTE.

The Basic System

Storage. While there are extensive facilities for entering quantities of textual materials off-line, it should be easy for a user to create new items and enter material as it comes to mind during the course of a work session, and to intersperse this storage activity with retrieval and the manipulation of linkages among items.

At any moment, a user is working in the context of an "active item." Anything typed which is not a command line becomes part of the text of this item. The command "*ENTER" first causes the current active item to be concluded and then opens a new item, assigning to it a numerical tag by which it may always be referenced. This item number, prefixed by a # (pound) sign, is displayed and the system returns to a monitoring state.

On the basis of the first character, AUTONOTE is able to recognize a variety of line types. It stores successive occurrences of each type in distinct regions of the text file. Beyond this, there are no internally imposed restrictions on the format of a text line. Each of these line types is associated with a mechanism by which that line is processed.

Descriptor entry. Descriptor terms are supplied by the user. Any arbitrary string of characters up to 16 characters long may be used as a descriptor.

To associate one or more descriptors with the text item he is entering, the user begins an input line with an @ (at) sign (thereby producing what hereafter will be referred to as an @-line). Each of the words occurring in an @-line is treated as a descriptor of that text item. Regular text lines and @-lines may be interspersed in any order during entry of a text item.

Any word used as a descriptor in one or more text items is associated with a line in a dictionary. Each such dictionary line contains pointers to all those text items in which that word occurs as a descriptor. Since the @-lines associated with a text item are stored with that item in the text file, the descriptors associated with an item also are available whenever the item is encountered.

There are no significant restrictions either on the number of distinct descriptors that may be defined, or on the number of descriptor terms that may be associated with an individual text item. We estimate, for example, that as the system now stands, the upper limit in both cases is on the order of 10,000.

Retrieval. An item may be displayed by issuing the "*PRINT" command and supplying an item number as argument. This rather direct method presupposes knowledge of the numerical tag of the item of interest. If this is not known, the user may enter a "*LIST" command, supplying as argument a word he has used to describe the contents of the desired items. AUTONOTE first responds with a count of the referenced items, i.e., a count of the number of times that word has been used as a descriptor. It then displays a list of item numbers referenced by the descriptor. These two operations are combined in the "*RETRIEVE" command, which again accepts a descriptor as argument and causes those items having this word as a descriptor to be displayed.

Manipulation of stored items. If a large body of information is to be maximally useful, there must be means for revising it, reorganizing it, and keeping it up to date. Several AUTONOTE commands are provided for this purpose.

The operations of deletion and replacement of text items are available through the "*DELETE" command. The form of this command is:

*

*DELETE #item-number [R=#item-number]

For example, following the command "*DELETE #34," the text item specified (item 34) is marked deleted. So are all references to the item in the dictionary.

After a "*DELETE" command, retrieval requests incorporating descriptors assigned to a deleted item no longer will return that item. A direct attempt to "*PRINT" a deleted item will produce a comment indicating the item's status and the date of deletion. If, in addition, a replacement item is specified (either at the time of the original deletion or later), any subsequent request to display the deleted item is referred to the replacement item. For example, if the command "*DELETE #34 R=#104," is followed at a later time by "*PRINT #34," item 104 will be displayed. It is of course possible that item 104 is itself a deleted item with a replacement; if this is true, the request to display the item is passed further on down the chain.

In addition to the implicit linkage between items possessing the same descriptor word or phrase, a user may link items together explicitly, with an "*APPEND" command. The form of this command is:

*APPEND #item-number [TO] #item-number [comment]

For example:

*APPEND #33 TO #12

*APPEND #121 #63

*APPEND #46 TO #27 PAPER BY SAME AUTHOR

In the last example, for instance, a new line is created in item 27 (an &-line, since it is prefixed with an ampersand). This line contains the number of the appended item (#46), followed by the comment shown. Any request to display an item possessing such a link, arising either from a direct "*PRINT" command or from a retrieval request, causes AUTONOTE to display the item followed by the accompanying list of appended items and the associated comments. This mechanism is useful, for example, when the user is browsing through a text item, is reminded of another relevant item, and wishes to jot down a reminder to "see also...."

Conclusion

It was intended that the preceding discussion familiarize the reader with the basic mechanisms of AUTONOTE. With knowledge of only these primitive capabilities, it is possible for a novice to begin using the system to store and subsequently display textual material, to retrieve individual items and groups of items on the basis of pre-assigned descriptors, to link items together, and to delete or replace irrelevant or obsolete units of text without disrupting previously established linkages.

Our rationale has been to present the novice user with mechanisms for organizing and describing his information which restrict him as little as possible. With practice, a user develops a stock of methods, conventions, and heuristics compatible with his work style, and not unnecessarily corrupted by externally imposed constraints.

*Throughout this paper, square brackets will be used in command descriptions to indicate optional argument forms.

Extended AUTONOTE

Time-Saving Aids

AUTONOTE incorporates a variety of elementary time-saving aids.

1. Only the first two characters (e.g., *E, *P, *R) are necessary to specify any command.

2. Commands accept multiple arguments of the appropriate type, acting on each in turn.

3. Recall that during entry of text items, AUTONOTE generates consecutive item number tags. The references in each dictionary entry also are temporally ordered. When retrieving items, the user may include with a descriptor an index (N) that refers to this temporal ordering. The series of retrieved items then begins with the Nth dictionary reference associated with this descriptor. For words that are used frequently to describe items, this is a distinct advantage.

Examples of indexed "*LIST" and "*RETRIEVE" commands, and their effects:

*L THESIS[LAST-4]
(displays numbers of the five most recent items entered containing "THESIS" as a descriptor)

*R PAPER[10]
(displays, from the 10th to the last, those items referenced by "PAPER")

4. Requests to "*PRINT" blocks of items, either backwards or forwards, are allowed.

*P #126, #76...79, #129...127
(displays, in order, items 126, 76, 77, 78, 79, 129, 128, 127)

5. Should an argument be omitted, the system will supply a default option in its place (possible default conditions are discussed below).

*P
(displays current item)

*A TO #56
(appends current item to item 56)

6. During an average work session in which a user is interspersing a variety of print, retrieve and enter operations, it is often necessary to remember important item numbers or descriptors for subsequent use. For this purpose, AUTONOTE provides a user with a mechanism for augmenting his own all-too-limited short term memory. A command "*HOARD," which accepts as argument any string of characters, places this string in a buffer, recoverable at any time with the command "*FETCH."

7. A user is able to interrupt execution of an AUTONOTE command at any time, causing immediate return to a monitoring state to await further commands or additions to text.

Advanced Capabilities

Though the aids just described make AUTONOTE easier to use, by themselves they add little to the basic capabilities outlined above. There are, however, several features of AUTONOTE which substantially increase an individual's ability to manipulate and retrieve stored information.

Composite descriptors. By employing "AND," "OR," and "MINUS" operators, users may construct combinations of descriptors for use in retrieving text items. These composite descriptors may be formed on a command by command basis, by specifying a composite descriptor as one of the arguments of a "*RETRIEVE" command; or if the composite descriptor is to be used frequently, the user may create a special temporary dictionary entry for it. The advantages of this second mechanism are twofold: subsequent retrieval attempts utilizing this composite descriptor will be processed much more rapidly; and this new dictionary entry may be combined with still other words to formulate even more specific retrieval requests.

The command "*KEYWORD," which sets up a dictionary entry for a composite descriptor, has the following form:

*KEYWORD $descriptor1=descriptor2.OP.descriptor3

OP may be "AND," or "OR," corresponding respectively to the intersection and union of the sets of text items referenced by descriptor2 and descriptor3; or OP may be "MINUS," corresponding to the intersection of descriptor2's references with the complement of those of descriptor3. The newly defined composite descriptor may subsequently be used anywhere an ordinary descriptor can appear. Any composite descriptor must begin with a $ (dollar) sign, to distinguish it from elementary descriptors.

It also is possible to specify a composite descriptor directly in a "*RETRIEVE" or "*LIST" command. If a name for this composite descriptor is given, AUTONOTE returns the requested items and forms the appropriate dictionary entry as mentioned above; otherwise requested items are returned but no change is made in the dictionary. Examples:

*L JONES.AND.1969
(displays numbers of items referenced by both "JONES" and "1969")

*K $POWER=MOTOR.OR.ENGINE
(creates a new dictionary entry [$POWER] containing the numbers of all items referenced by either "MOTOR" or "ENGINE")

*R ROCKET.MINUS.$POWER
(displays all items referenced by "ROCKET," but not by "MOTOR" or "ENGINE")

*R $CAR=HORSELESS.AND.CARRIAGE
(creates a new dictionary entry [$CAR] containing the numbers of all items referenced by"HORSELESS" and "CARRIAGE," and displays those items)

Descriptor modification. In an evolving information structure, some of the descriptors initially associated with an item subsequently may need to be changed. The user may come to regard an item in new ways, and thus wish to add new descriptors or delete old ones. Or the meanings of certain descriptors may be altered over time, either deliberately, or as a reflection of the user's changing conceptual framework.

To accommodate the possibility that the current assignment of particular descriptors to certain items no longer may be appropriate, AUTONOTE enables users to add and delete @-lines. The "*MODIFY" command provided for this purpose has two forms:

1) *M #item-number + descriptor[,descriptor,..]

2) *M #item-number - $\{^{N}_{ALL}\}^{*}$

The result of a "*MODIFY" command of the first type is to add a new line containing the specified descriptors to the @-lines already associated with the given item. In addition, a reference to the item is inserted in the dictionary entry for each descriptor specified.

In the second type of "*MODIFY" command, if (N) is used it specifies the ordinal position of the line to be removed in the block of @-lines associated with the item. Alternatively, "ALL" may be used to indicate that all @-lines are to be removed from an item. In either case, the designated @-line(s) is erased from the text and the reference to the item is deleted from the dictionary entries of each of the descriptors involved.

Grouping. AUTONOTE includes a grouping convention which permits the user to organize text items in several useful ways. Suppose, for example, that a user has entered items corresponding to points to be made in a lecture or in a chapter

of a book, and that he now wishes to treat the set of items as a unit. The grouping convention enables him to define a new higher order element which groups the individual elements in an ordered list. If the @-line portion of an item begins with the word "GROUP," followed by a list of item numbers, this item can be treated by the "*PRINT" command in a special way. An attempt to display such a grouping item results instead in the display of the component items, in the order specified in the "GROUP" @-line.

In effect, a grouping item is a node in an inverted tree structure, with downward branches to those items listed in its "GROUP" @-lines. Since any of the component items in a grouping item may themselves be grouping items, it is possible to generate a complex hierarchical organization in this way. Given such a hierarchy, a request to display an item containing a "GROUP" @-line initiates recursive processing of intermediate nodes, and produces the terminal items of the subtree headed by that item. Alternatively, the user may specify either that only the non-terminal items be displayed, or that the terminal and non-terminal items be displayed. Note that when limited to a single level, the grouping mechanism provides an easy way to specify an arbitrary order of many items for subsequent display.

The hierarchical structures set up by grouping are easily altered with the deletion, replacement, append, and descriptor modification operations previously described, thus enabling users to generate, update, or reorganize any portion of a total text. It also becomes possible to store any number of alternative organizations of materials at the same time.

In addition to moving down through structures set up by the use of grouping, it also is possible to find out which if any immediately higher order structures a given item is used in, by requesting retrieval on the conjunction of the item number and "GROUP." This is so since the word "GROUP" and the numbers of all grouped items are descriptors in the @-lines of the same grouping item. With successively higher order item numbers as inputs, repeated applications of this technique yield all the higher order structures of which the initial given item is a part.

Symbolic and default item number. One often wishes to refer repeatedly to a particular item, or to refer to the last item displayed (for example, when many modifications are to be made to an item, or when the user is stepping through series of contiguous items). The * (asterisk) is provided as a symbolic item number reference. It may be used in a command line in place of an item number, and will be interpreted in one of three ways, as determined by the user (see the section on command modifiers immediately below). The possible meanings are: 1) the current active item (that is, the item into which text is being inserted); 2) the item last displayed or listed; 3) a fixed item set by the user. A user also may specify items relative to that symbolized by the "*", by using arguments of the form "*±N" in a command line (where N is any integer).

The following sequence of commands demonstrates possible uses of the symbolic item number facility (assume the present interpretation of "*" to be the last item displayed).

```
*PRINT #304...306
     (displays items 304, 305, and 306)

*MODIFY * + SUMMARY, REVISION
     (adds descriptors "SUMMARY" and "REVISION"
      to item 306)

*PRINT *+1
     (displays item 307)

*APPEND #500 TO *
     (appends item 500 to item 307)
```

It is also possible for the user to specify any of the above three interpretations for the default item number - that number used by AUTONOTE when an item number is completely omitted from a command.

Consider as an example the following sequence of commands (assume the present interpretation of the default item to be the active item).

```
*P #217
     (displays item 217)

*E
     (creates new  item)

*APPEND TO #217
     (appends newly created item to item 217)
```

Command modifiers. A system of modifiers makes it possible to control the execution of many of the commands discussed. The following represent some of the options available to the user.

1. The format of a displayed item may be set to consist of any or all of the following: a header (item number and bookkeeping details); the main body of text (possibly truncated to N lines); the @-lines; or the &-lines.

2. The extensiveness of the error comments may be altered, depending on the proficiency of the particular user. Normally, any error evokes an explanatory comment. In a "terse" mode, an error produces only a "?". There upon the user may elicit the explanatory comment with a "*WHY" command, if desired.

3. Expansion of a grouping item, as mentioned earlier, may be controlled.

4. The user may "trace" a chain of replacements for a deleted item.

5. The listing of item numbers following a "*LIST" command may be omitted, producing only a count of the number of references.

6. As mentioned previously, the interpretation of the symbolic item reference "*" and the default item reference may be specified individually.

The user controls these options by changing the values of a set of modifiers. Most modifiers have only two possible settings: ON or OFF. A few may take on other kinds of values.

Each of the modifiers has a default value. The default values are chosen to simplify use by the novice, so that he initially need not know anything about the modifiers or what they do.

The values of modifiers may be changed from their default conditions with the "*SET" command. Giving a modifier name as argument to this command turns on the modifier; prefixing the name with "NO" turns it off. For those modifiers which can take on a broader range of values than just ON or OFF, one gives both the modifier name and the desired value. These changes accumulate over successive "*SET" commands; that is, at any time the state of a command is determined by the initial set of default values and all changes made to date. The original set of default values may be restored at any time by entering "*SET" with no arguments.

Local modification of the current set of values is allowed within command lines. The insertion of these modifiers, prefixed in this case with an & (ampersand) is effective for the duration of that command.

Examples of modifier usage and their effects:

```
*SET NO@
     (suppresses the display of @-lines)

*SET *=ACTIVE
     (subsequently, "*" will be interpreted as
      the item into which input lines are entered)

*SET NULL=#365
     (item 365 will be supplied for all missing
      item number arguments)

*SET
     (resets modifiers to system default values)

*RETRIEVE &NOMAIN &NO& ALGOL
     (displays @-lines of all items references by
      "ALGOL")

*PRINT &NO@ #302...305
     (displays items 302 through 305 without @-
      lines)

*PRINT &EXTENT=L  #23
     (displays first line of item 23)

*PRINT &RECURSIVE(ALL)  #12
     (displays all items, terminal and non-ter-
      minal, formed using the "GROUP" convention,
      beginning with item 12)
```

The "*COPY" command. At times a user may wish to employ a significant portion of some previously entered item (or items) as the basis of a new item. The "*COPY" command accepts the same type of arguments as the "*PRINT" command and transfers the specified items or portions of items to a work space with access to a text editing routine. The modified text then may be reentered to AUTONOTE or may serve as a source for finished copy.

Subsystems of AUTONOTE. In certain circumstances, a user may not require the full set of AUTONOTE routines. For example, a user with an existing collection of notes or other materials may wish to enter substantial amounts of text off-line. For these purposes, he requires only those AUTONOTE routines which implement the "*ENTER" command. Simple procedures exist which enable users to specify certain restricted sets of command requirements and processing capabilities, with resulting reductions in cost.

File Organization Strategy

In the introduction, reference was made to a broad range of possible AUTONOTE textual material: notes, bibliographic references, summaries, and so forth. Now that the operation of AUTONOTE has been described, some further possibilities may be worth noting.

1. Consider the utility of storing as items information about descriptors. We may, for example, store definitions, synonyms, abbreviations, relations to other descriptors, and membership in classes, in effect building up a thesaurus.

2. Recall in the discussion of the grouping convention that items could be considered nodes of a tree structure. At the same time that the @-lines of such an item specify the subordinate branching, the text of the grouping item also may contain a full description of the relation intended, or supplementary material relevant to the derived items as a whole.

3. The "*APPEND" construction can be used to specify long chains of associative item-item links (via &-lines).

4. Without disturbing the present system, it would be feasible to create additional line-types within an item, along with additional conventions for each. For example, it would be possible to equate a line-type with a particular item-item relation.

In sum, our experience suggests that AUTONOTE can be a powerful tool for facilitating personal information storage and retrieval activity

for several different kinds of users. Every effort has been made to insure that AUTONOTE is maximally adaptable to individual needs.

Hardware and Software Details

System Environment

AUTONOTE runs as a user program on the Michigan Terminal System (MTS), a time-sharing system implemented on The University of Michigan's IBM 360/67 computer. As such, it can be run from any of the various terminals to that system. The terminal user also has available all the other benefits of a general purpose time-sharing terminal (e.g., file creation and editing, use of many programming languages and library subprograms, remote entry of batch jobs, input from a variety of other system equipment, etc.). The system currently supports as terminals, via the dial-up telephone network, Model 33 and 35 teletypes, IBM 1050's and 2741's, TOUCHTONE telephones, and certain special facilities linked by medium and high-speed datasets. Typical system response times, under normal MTS loading, range from 1 to 3 seconds for most AUTONOTE commands (this is from the time the command is transmitted to the time the retrieved information begins to come back).

Display Terminal

The output rate of the available typewriter-class terminals was considered too slow for many applications, in particular those which require frequent retrieval of significant quantities of text. We therefore employ a faster (and quieter) CRT display as the main retrieval station. This station is a Computer Communications, Inc. Model 30 keyboard-display unit. It is connected via a Bell System Model 202C data set and voice-grade phone line to a small terminal handling computer (an augmented PDP-8), located at the Computing Center, which provides the interface to the 360. The special programming support needed for controlling the remote display terminal is located in the PDP-8.

This configuration provides a data transmission rate of 120 characters per second. The effective character rate is somewhat less due to line turn around overhead. The roughly paragraph-size (800 character) display screen can be filled in 8 to 12 seconds, depending on record length.

The display (which is a slightly modified television set) is arranged in 20 lines of 40 characters each, with a displayable character set of 68 upper case alphabetic, numeric, and special symbols. It contains a 1024 byte core memory, from which the display is locally refreshed at 60 cps. As the user types at the keyboard, the information goes directly into this local core buffer, and is displayed as he types. He is free to edit the material with a moveable cursor before it is transmitted to the computer.

Various device commands are available from the terminal. The user may clear the display screen when needed, or may have it done automatically at the end of a "page" (i.e., when the screen is full). There is also a mode in which the display stops at the end of each page and waits for a key to be pressed before continuing with the next page of information. (See a description of the Baronet system[2] for a more specifically page-oriented system using similar keyboard display facilities.)

Protocol record. To compensate for the lack of a hard copy facility at the display terminal, AUTONOTE includes a means for storing a complete record of an AUTONOTE session (all lines typed by the user, and all lines displayed by the system) in an MTS file, for later display perusal or off-line printing. This protocol recording facility may be activated at the beginning of a session, or may be left inactivated if such a record is not needed. See also the section on the study of scientific thinking, below, for a discussion of other uses of these protocol records.

Programming Details

AUTONOTE is written in 360 assembly language. It uses the extensive disk storage facilities of MTS for both the text of the data base, and for the dictionary containing the descriptors and lists of referenced text items.

MTS disk storage is organized into line-files, the unit of information being a line of from 1 to 255 characters. Each line has a unique serial number, in the range -99999.999 to +99999.999. Each file has a line directory, ordered by line number, which allows rapid retrieval by line number.

As AUTONOTE text items are entered, they are assigned sequential item numbers, and each line is entered sequentially into a block of line numbers unique to that item (e.g., item 6 goes into line 6.001, 6.002,...); @-lines also are kept in the text file, but in a distinct line number range. Thus, to retrieve both the text and descriptor material of an item, the program need know only its serial number.

As mentioned earlier, descriptors, in addition to their storage in @-lines with the text, are entered into a dictionary along with the associated item number. The dictionary file line number in which the references are entered is determined by hash-coding the descriptor. That is, this line number is an arithmetic function of the character representation of the discriptor.* To find all those text items which have been described with a particular word, the system

*The line number used is the remainder from the division of the character representation by the product of two large primes. The usual mechanisms are used to resolve multiple hits, but they are rare because of the very large range of this function - zero to about 90 million.

simply hash codes the word to a line number and reads that line from the dictionary file. The dictionary line contains the text file line numbers of all those items referenced. The significance of this arrangement is that retrieval time is substantially independent of the size of the data base. This is achieved without using large amounts of core storage for the associative mechanism, which is of vital importance in keeping the cost of using the system down.

Practical Limits of the Current Running System

There are two minor limitations on the content of stored items. Lines of text may not begin with either @ or *, and certain special characters (.,;:?!'¢"[]) may not be used in descriptors. They may occur in @-lines, to improve readability, but are considered break characters in the process of segmenting the @-line into descriptors.

More important are those limitations that might affect the practical utility of the system. The possible limitations here have to do with cost of operation, speed of retrieval, and amount of material to be stored.

Cost of using the system varies with the amount of connect time, the amount of interaction taking place during a session, the amount of virtual storage occupied, and the amount of disk storage occupied by the text and dictionary files. We have been able to keep virtual storage requirements down to two to five pages (a page is 4096 eight-bit bytes), the exact amount depending on the particular set of AUTONOTE features in use. This results in costs on the order of $7 to $12 per terminal connect hour under the current MTS billing algorithm. This is very little more than the minimum attainable connect time cost for MTS.

At MTS's current rates per page hour of file storage, the cost of storing the equivalent of 100 8-1/2 x 11 pages of double-spaced typed text is about $2.50 per week (plus about half that amount for a typical dictionary file). It should be emphasized that these file storage charges are dependent on the set of system facilities available. For example, Michigan is about to implement a data cell form of mass secondary storage. It is anticipated that files stored in this device will cost approximately 1/5th the present file storage cost.

Retrieval speed has already been mentioned. Simple retrieval requests typically are answered within 3 seconds, more complex ones take 5 or 10 seconds. Response times of this magnitude, in conjunction with the 100 character per second display output rate, produce a fairly high degree of user satisfaction.

The last limitation we need to consider is on data base size. One MTS file will hold approximately 10,000 lines of average length (about 72 characters per line). However this limit is not in any sense the upper limit for AUTONOTE. That limit is determined by the number of distinct items that the dictionary can recognize and reference. At present, the system can recognize about 90,000 distinct items. If we assume that each item contains the equivalent of 10 lines of typed text per item (the upper limit is 500 lines per item), this gives the system an upper limit of 900,000 lines of material, or approximately 30,000 8-1/2 x 11 double spaced pages of typed text.

To sum up, the present operating system in its intended use for personal informational storage and retrieval seems quite practical with respect to the range of materials it can store, its operating costs, its speed of operation, and the volume of material it can handle.

AUTONOTE and the Study of Thinking

The foregoing description has focused upon AUTONOTE as a working system, an adjunct to the information organizing and handling activities of individual users. In point of fact, however, AUTONOTE originated as part of a study of scientific thinking. Only recently have we come to think of it as a general means of facilitating personal information processing, somewhat along the lines of systems such as those discussed by Englebart and English,[3] and Nelson.[4]

In wanting to know more of how working scientists use information in thinking, our concerns were very close to those expressed in the NRC report[2] cited previously:

> "The scientist makes progress... by adding new information to existing knowledge in ways that suggest more powerful explanatory principles and point to fruitful hypotheses for additional research. Very little is known, however, about the cognitive process by which this is accomplished. Since the facilitation of inquiry is the ultimate purpose of a scientific information system, it would be useful to learn: (a) how scientists...organize existing knowledge and use it as the basis for seeking information; (b) how information...can best be 'packaged' to meet scientists' needs. Studies directed to these questions might suggest ways of indexing and classifying information to complement association patterns in the minds of scientists, and to develop more efficient information 'packages.'"

AUTONOTE is a tool for pursuing these ends. The many options available in the full AUTONOTE system enable individual scientists to shape the system to their own needs and their own ways of organizing and using their materials. In so

doing, they provide us with a more accurate picture of how they work. Thus AUTONOTE helps make explicit a substantial amount of the scientist's information handling activities.

The AUTONOTE protocol facility described above, obtains this information for us. When switched on, the protocol recording routines, which are transparent to the user, store for subsequent analysis a complete record of all user commands and all materials the user produced and retrieved. Such records serve two purposes: they provide detailed and accurate data for the psychological study of thinking; and they indicate usage patterns, thus providing information that can be used to make the system of greater practical value. By making available to us behavior records that reflect the complex information handling strategies of intelligent users, the protocol facility helps us to induce those strategies and to try to implement them directly, thus augmenting the power and intelligence of the system.

The theoretical framework for these investigations[5,6] emphasizes relationships between memory and communication. It is concerned with the use of what we know to disambiguate, make plausible inferences, and fill in implied meanings.

Ordinary language rarely communicates complete information explicitly. It depends upon the listener's ability to make inferences from prior information, from context, and from a knowledge of the speaker and the world. Communicating in this way, we risk occasional misunderstanding as the price for avoiding verbose, redundant messages largely consisting of material the listener already knows. Efficient communication is achieved by referring to already stored schemas and indicating properties or modifications peculiar to the present instance or situation.

Present research with AUTONOTE is investigating ways of incorporating such techniques to permit the user to call larger strategy units with fewer commands and, consequently, to reduce the number of explicit information interchanges needed to achieve the desired result. We are basing this software development on constructs inferred from observations of use of the current system. By asking in effect how an artificial intelligence system might take over some of these functions, we hope to achieve both a more powerful aid for human problem solvers and a better understanding of the structure of human intellectual activity.

Acknowledgements

We wish to thank Mr. Peter Bono for his help in writing portions of the AUTONOTE system, and Dr. Richard Bailey and Dr. John Reidy for helping to evaluate and improve its effectiveness. We also wish to acknowledge our appreciation of support for the work received under U.S.P.H.S. Grant MH12160 to The University of Michigan.

References

[1] National Research Council, Division of Behavioral Sciences, Communication Systems and Resources in the Behavioral Sciences. Washington, D.C.: Nat. Acad. of Sciences, Publication 1575, 1967.

[2] J. W. Young, "BARONET: A simplified CRT-based information storage and retrieval system," Proc. ACM 23rd Nat. Conf., pp. 53-59, 1968.

[3] D. C. Engelbart and W. K. English, "A research center for augmenting human intellect," Proc. FJCC, pp. 395-410, 1968.

[4] T. H. Nelson, "A file structure for the complex, the changing, and the indeterminate," Proc. ACM 20th Nat. Conf., pp. 84-100, 1965.

[5] W. R. Reitman, Cognition and Thought. New York: Wiley, 1965.

[6] W. R. Reitman, "The uses of experience: open statements, ill-defined strategies, and intelligent information processing," Cognitive Studies, vol. 1, J. Hellmuth, Ed. Seattle: Special Child Pub., 1969.

[7] R. W. Bailey and J. L. Robinson, "Computers and dictionaries," Proc. Conf. on Computers and Old English Concordances," Toronto, March 1969.

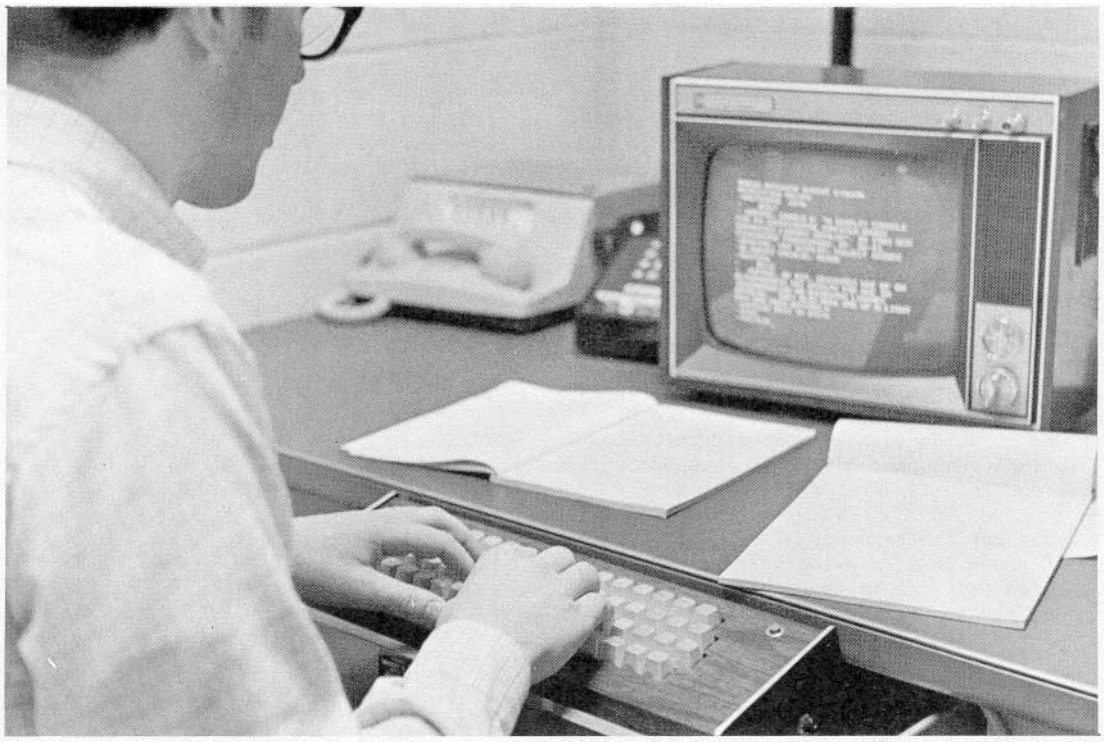

Figure 1. Working environment of the display environment.

```
-*RETRIEVE  CRT↵
-     #357
- HARING, DONALD R. "A DISPLAY CONSOLE
FOR AN EXPERIMENTAL COMPUTER-BASED
AUGMENTED LIBRARY CATALOG". IN 1968 ACM
CONFERENCE PROCEEDINGS, PP. 35-43.
-@HARING, CRT DISPLAY, PROJECT INTREX
-@LIBRARY CATALOG, ACM68
*ENTER↵
-     #512
-   HARING, AT MIT, DESCRIBES USE OF AN
ALPHANUMERIC CRT DISPLAY IN INTERROGATIN
G AN AUGMENTED LIBRARY CATALOG.
-*APPEND #512 TO #367↵
-*DONE
-*LIST   INTREX.AND.THESAURUS↵
-KEYWORD RELATION: INTREX.AND.THESAURUS
-     3 REFERENCES
-       #65      #217      #218
-*P #65↵_
```

Figure 2. Sample of display output.