The Memory Extender Personal Filing System

William P. Jones

Microelectronics and Computer Technology Corporation
9430 Research Boulevard, Echelon #1
Austin, Texas 78759
512-834-3326

Abstract

The benefits of electronic information storage are enormous and largely unrealized. As its cost continues to decline, the number of files in the average user's personal database may increase substantially. How is a user to keep track of several thousand, perhaps several hundred thousand, files? The Memory Extender (ME) system improves the user interface to a personal database by actively modeling the user's own memory for files and for the context in which these files are used. Files are multiply indexed through a network of variably weighted term links. Context is similarly represented and is used to minimize the user input necessary to disambiguate a file. Files are retrieved from the context through a spreading-activation-like process. The system aims towards an ideal in which the computer provides a natural extension to the user's own memory.

1. Introduction

In its capacity for the storage of vast quantities of information in a readily accessible form, human memory is magnificent. However, human memory as a medium of information storage is not without its limitations. For example, much initial time and effort must be invested to integrate a piece of information into internal memory - both to insure its immediate comprehensibility and to insure its accessibility at a later point in time. Moreover, this information, once acquired, is often forgotten over time.

Human memory is extended, and its limitations partially compensated for, through the use of mediums of external information storage (e.g., paper, microfilm, magnetic disks and tapes, etc.). In this regard, the potential of electronic storage is enormous and largely unrealized. As its costs continue to decline, we may anticipate a time when the average user may wish to store several thousand - perhaps several hundred thousand - information objects (i.e. files) in an on-line personal directory. How is a user to keep track of all these files? Unfortunately, in the use of electronic storage (or any other external storage) limitations of human memory are exchanged for another set of limitations that are imposed by the filing systems through which information is stored and retrieved.

Consider the predominantly name-oriented approach of most current filing systems. The user of such systems experiences problems of both recall ("what name did I call that file by anyway?") and recognition ("what's in that file anyway? - I can't tell from its name"). The single-name representations of a computer-based filing system stand in stark contrast to the richly indexed representations of human memory. This leads to an impoverishment in the language of the human/computer dialogue. In one direction, few of the many attributes a user may be able to recall regarding a desired file are likely to be understood by the computer in the sense that these are used to delimit the computer's search for that file. In the other direction, when the computer's representation of a file (e.g., the file's name) is lacking in descriptive content, it may not be understood by the user in the sense that it activates an appropriate representation in the user's internal memory.

The Memory Extender (ME) system is motivated by an ideal in which a computer-based personal filing system realizes the benefits of electronic storage without sacrificing the benefits of human memory. Towards this end the ME system is equipped with mechanisms to actively model the user's representation of files and the context within which these files are used. Files are multiply indexed through variably weighted, bi-directional term links to insure both their recallability and recognizability. In this respect, the ME system functions much like a keyword retrieval system. However, the ME system also acts, through its representation of context and its adaptivity, to minimize the user input needed to distinguish a file from other files in the database (either for purposes of storage or retrieval).

The ME system is currently implemented in Zetalisp and can run on any 3600 series Symbolics Lisp Machine. It is integrated into the ZMACS(T) text editor and essentially "piggy backs" the variety of filing systems (e.g. Tops 20(T), VMS(T), Unix(T)) that currently service a Symbolics Lisp Machine. Consequently, the user is always free to use the standard, typically name-oriented, functionality of these filing systems.[1]

2. The design and usage of ME

The description of the ME system begins with a general overview of its representational scheme. This is followed by a look at some of the ways in which the ME system can be used. Finally, three important components of the ME system - its decay mechanisms, its means of adaptation, and its spreading activation matching algorithm - are each discussed in turn.

2.1. The representational scheme.

2.1.1. The representation of files

A file is represented, i.e. indexed, in the ME system by a set of variably weighted, bi-directional term links (see Figure 2-1). From the user's perspective, bi-directionality carries both a recall and a recognition function. In one direction, the user can directly access (i.e. recall) a file through a specification of some of its associated terms. In the other direction, the computer can represent the file to the user thus enabling its recognition (as distinct from other files) through a presentation of its associated terms.

The variable weighting of terms provides a basis by which to order files according to the likelihood of their reference (see Furnas, Landauer, Gomez, & Dumais, 1983) and so aids in file recall. Such a weighting scheme can also be used improve a file's recognizability. For example, the computer can order the terms in a file's representation on the basis of their weights before presenting these terms to the user.

2.1.2. The representation of context

For the purposes of this article, context is defined to be a set of representational givens that hold across a series of filing transactions. To the extent that the user and the computer share such a common context, exchanges can be simultaneously brief and precise. Consider a very simplified situation involving a filing system of letters in which all combinations of letter type ("business", "birthday", "Christmas", ...) and letter recipient

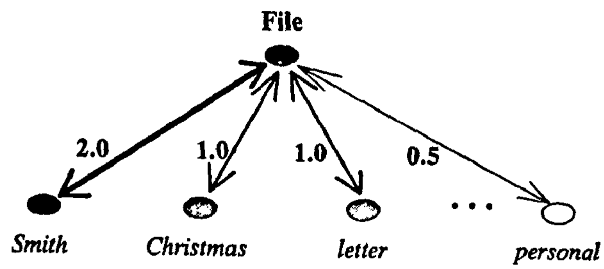

Figure 2-1: A sample representation of a file in the ME system

[1]The files of a conventional filing system can be given an initializing representation in the ME system through a procedure that segments file pathnames into indexing terms (e.g. "/usr/hank/letters/Christmas/smith" -> "usr", "hank", "letters", "Christmas", "smith").

("Jones", "Smith", "Brown", ...) are realized. A user's attempt to retrieve a letter through specification of the term "Smith" is then ambiguous unless it is understood that the user is working with a series of "Christmas" letters. A shared context thus obviates the repeated specification of the term "Christmas" with each new retrieval attempt.

A shared context can also be used to reduce the amount of information a user must specify regarding the representation of a newly created file for purposes of storage. In a context of Christmas letters, for example, a newly created letter might reasonably "inherit" the term "Christmas."

In the ME system, context, like a file, is represented by a set of variably weighted term links. In contrast to a file's representation, however, these links are uni-directional: A link can be traversed from the context object to its associated terms, but traversal in the reverse direction is not possible (i.e. it is not possible to "retrieve" the context through a specification of its terms). Among its many uses, the context object is a query during retrieval attempts and it forms the basis of a newly created file's representation.

The ME system's representation of context can be contrasted with the hierarchical approach of a system such as UNIX. In a file system hierarchy, the user establishes a context, of sorts, through the specification of a "path" leading to a desired subdirectory. This path is then appended to user-generated file names, both during storage and retrieval attempts.

Several problems are generally associated with the use of a file hierarchy:

- In its ability to represent the context of a user/computer interaction, a single hierarchy is extremely limited since files in the hierarchy are essentially partitioned in only one way.
- The hierarchy is static - it cannot be reconfigured except through explicit, often laborious, input from the user.
- The user must know a great deal in order to specify a file that is not in the current working directory. The user must generate a path to the file and the components of this path must be specified in their proper order.
- Within a working directory, file retrieval is essentially name-oriented.

2.2. Use of the ME system

Figure 2-2 presents one example of ME system usage. The user has initiated the formation of a list of partial matches to the current context. In this list, matching files and the current context are each represented by their own mouse-sensitive line of terms (stronger terms are leftmost). Clicking the rightmost mouse button on such a line temporarily selects the line's object (file or context) bringing, up a submenu of relevant mouse-sensitive operations.

If the selected object is a file, some of the relevant operations are:

create new version
: - A new file is created with contents and representation initially identical to those of the selected file. This file is placed inside its own buffer in the ZMACS editor.

specify terms
: - A link to a specified term is added to the file's representation with a default weight of one. If a link to this term already exists, its weight is increased by one.

specify terms and weights
: - A link to a specified term with the specified weight is added to the file's representation. If a link to this term already exists, its existing weight is replaced by the newly specified weight.

retrieve file
: - The file is copied into its own buffer in the ZMACS editor.

Additionally, the user can save the file's representation and its contents, erase the file's term links, or delete the file and its representation.

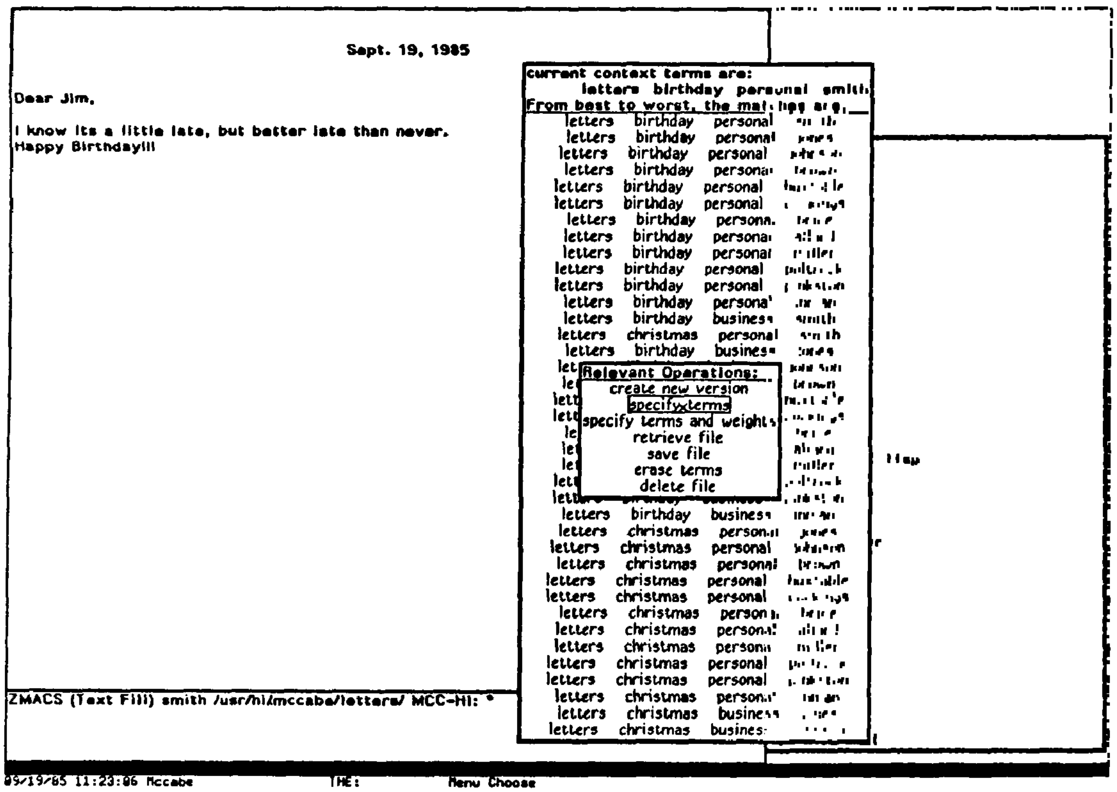

Figure 2-2: An example of ME system usage (the context and files are each represented in an ordered line of terms).

Analogous operations apply to a context object:

- Context terms (and their weights) can be specified.
- A new file with representation identical to that of the context object can be created (i.e. the file inherits context object attributes).
- Term links of the context object can be erased (the user may sometimes wish to "start all over").

The current context can also be saved and subsequently reinstated. The *save* operation adds a copy of the current context to a list of context objects. This list can be used to form a menu; selection of an object from this menu then causes it to become the current context. This facility is particularly important in interrupt-driven tasks in which the user may wish to switch rapidly from one context to another.

With reference to Figure 2-2, specification of terms, whether file or context, causes a dynamic readjustment in the partial match listing. Files may move up or down in this listing and the position of terms within a line representing a file or the current context may change depending upon the change in their relative weights. The user is thus given direct feedback on the effect of his/her input to the system. The prescribed strategy for file retrieval is to modify the current context through the specification of terms that are associated to the desired file. This can be an iterative procedure which continues until the file's representation is recognized somewhere on the list.

The user can also initiate the formation of a list of partial matches to a file that is being manipulated. The format of such a presentation is identical to that of Figure 2-2 except that the wording "current context" is replaced by "current file." This listing permits the user to retrieve or otherwise work with files that are related to the current file. The listing also helps the user to determine whether the representation of the current file is sufficiently distinct from that of other files in the database.

2.3. The Decay Mechanisms

Although the user is permitted to specify term weights directly, experience with the ME system indicates that this is seldom necessary. The simple specification of terms (with their default weight of one) generally suffices to accomplish the desired alteration in the representation of a file or the context. This would not be possible, however, were it not for the operation of two important mechanisms of decay.

Each time the current context object is modified, a *context decay mechanism* operates to keep the context object's *strength* - defined to be the sum of its term weights - at or below a constant value. Similarly, a *global decay mechanism* periodically operates so that file strengths average to a constant value. In both cases a *decay factor* is formed as follows:

$$d = P / A,$$

where P and A are the prescribed and actual strengths (or average strengths), respectively. The weights of all term links involved are then simply multiplied by d (providing $d < 1$). A term link is deleted if its new weight falls below a certain threshold (currently equal to .1). The value of P for purposes of both context and global decay is somewhat arbitrarily fixed in the current ME system at the constant value of 5.0. (Parameter choices of the ME system will be discussed later in this article.)

The operations of decay prevent a sort of "representational inertia" from making the system progressively more resistant to change. Decay also "washes away" links whose initial formation was accidental or whose current existence is no longer useful.

Although the average strength of a file remains constant in the ME system, there will generally be considerable variation in the strengths of individual files. In the context of the matching algorithm, to be discussed shortly, this has the generally beneficial effect of giving preferential treatment to files whose use is frequent.

On the other hand, if the use of a file is very infrequent, its strength will eventually approach zero, effectively rendering the file unretrievable. The ME system calls the user's attention to files whose strengths have fallen below a certain threshold under the assumption that these files may no longer be needed. The user then has the option either to delete these files or to strengthen their representations. The ME system thus provides a principled means by which to cleanse the database of files whose usefulness has passed. To the extent that the global decay mechanism mirrors a forgetting process in the user's own memory, this facility also serves to remind the user of the existence of "forgotten" files. (But, see Wickelgren, 1976, for a discussion of human forgetting).

2.4. Adaptation in the ME System

To the extent that the ME system can make appropriate modifications in its representations, the need for the direct specification of terms by the user is lessened. Towards this end the ME system is equipped with a mechanism of *representational information exchange*. This mechanism attempts to follow an important interplay that takes place between the files of a database and the context within which these files are used. In one direction, representational similarities among the files a user is currently working with can be used to define, at least in part, the current context. In the other direction, a given file's representation can be defined, in part, by the various contexts in which it has been used.

In the ME system, representations of the context and the file can viewed as vectors **C** and **F**, respectively, of term weights. After it has been determined that a newly retrieved file is in fact desired[2] , the context is modified in the following manner:

$$C_n = C_{n-1} + rF.$$

[2] The user can explicitly state that a file is desired or the system can infer this from user-initiated operations such as file storage/modification.

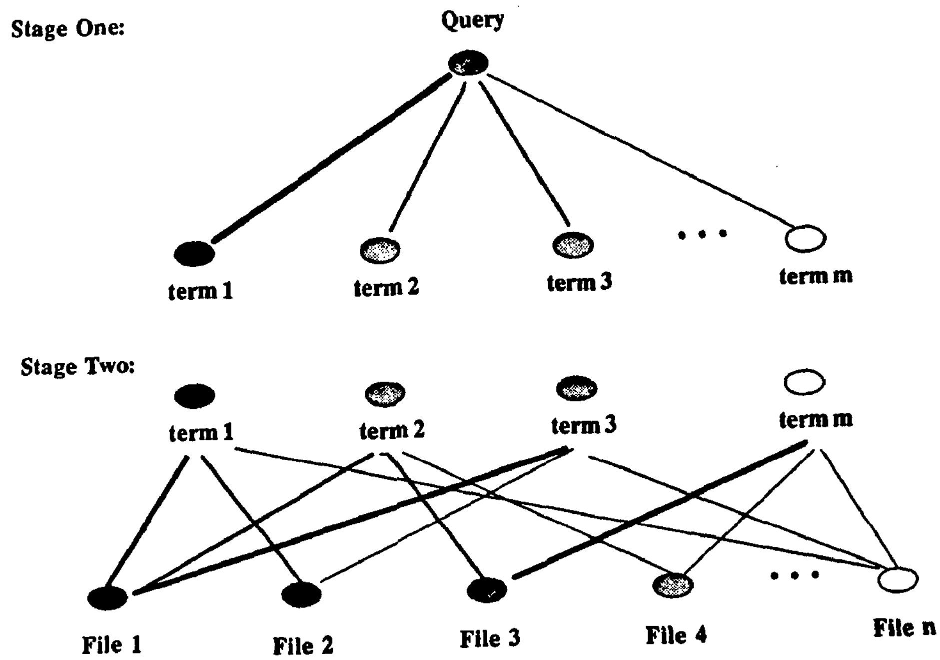

Figure 2-3: The ME system's spreading activation matching algorithm

Similarly, the file is modified in the following manner:

$$F_n = F_{n-1} + sC.$$

(Currently both r and s have the somewhat arbitrary value of .2). These operations are similar to the document retrieval operations of *relevance feedback* and *dynamic document vector alteration* (Ide & Salton, 1971; Salton & McGill, 1983).

A sort of one-sided exchange of information also takes place when a new file is created from the current context. In this case, the file's initial representation is identical to that of the current context, but the representation of the current context remains unchanged (since the new file has no information to give in return). In this manner the new file inherits representational givens of the current situation.

2.5. The Matching Process

The matching process of the ME system is used to order the files of a database according to their representational similarity to a given query (e.g. the context or an active file). In the first stage (see Figure 2-3), a single unit of activation A is divided among the terms of the query (recall that this can be either a context or a file object) according to their relative weights. Specifically, a given query term t receives activation,

$$a_t = A * (w_{Qt} / W_Q),$$

where w_{Qt} equals the weight from the query to term t and W_Q equals the sum total of all the query's associated term weights.

In the second stage, the activation received by a given term t in the first stage is further partitioned among each of its associated files so that a given file receives activation,

$$a_F = a_t * (w_{tF} / W_t),$$

where w_{tF} equals the weight from term t to file **F** and W_t equals the sum total of all of term t's associated file weights. The activation a given file receives in this manner from several different query terms will sum together. Files can then be ordered in a partial match listing according to their activation levels.

This spreading-activation matching algorithm, when applied to the ME system's bi-directional network of links, realizes two important benefits

- Changes in the discriminatory values of terms (see Salton & Yang, 1973; Sparck Jones, 1972), resulting from the addition or modification of files, are dynamically and automatically included in the calculations of the matching process.

- The ability to traverse links in the term-to-file direction permits the ME system to limit the scope of the matching process to a subset of the files in the database.

With respect to both of these benefits, the ME system compares favorably with a class of vector similarity measures used in document retrieval systems (Jones, 1985a, 1985b; also see Salton & McGill, 1983, for a review of vector similarity measures). The ME system realizes further gains in computational efficiency by considering only those links, at each stage of the matching process, whose weights exceed a threshold (currently set at .1).

An obvious extension to the matching process is to allow activation to spread further. It can be shown (Jones, 1985a) that this provides the ME system with a limited ability to induce term synonymy.

3. Discussion

In summary, the ME system possesses the following features:

- A multiple-term indexing of files that improves both recallability and recognizability.
- A multiple-term representation of context that is used to minimize the user input necessary to disambiguate a file.
- Decay mechanisms that reduce representational inertia and remove outdated term links.
- A mechanism of information exchange that adaptively alters the ME system's representations of the current context and retrieved files.
- A spreading-activation matching process whose actions are automatically and dynamically influenced by changes in the representational structure of the database.

The evaluation of the ME system is at a very early stage. The ME system has been successfully applied to a moderately sized UNIX file structure of about 600 files and the experience with its use thus far has been very positive. Particularly striking is the ease with which files can be retrieved, even when these files come from widely separate subdirectories in the underlying UNIX file hierarchy. Since files can be related in so many different ways - limited only by their representational richness - the storage of files is also very easy. It is no longer necessary, for example, to give careful consideration to a file's placement within the underlying file hierarchy. During actions of both storage and retrieval, the use of context has the intended effect of greatly reducing the requisite user input. The specification of from one to three terms is generally sufficient, for example, to place any desired file among the top ten of a partial match listing.

Performance of the ME system depends upon the values of its parameters in ways that are not yet fully understood. What should be the strength of the current context or the average file? If these strengths are too low, then representations are unduly influenced by the current interaction. If these are too high, then representations may become too resistant to change. Should these strengths be permitted to increase with the size of the database? If so, how? To what degree should the representations of retrieved files or the current context be modified by the mechanism of information exchange?

Some of these issues may be resolved through an iterative evaluation of ME system performance. It should also be noted that components of the ME system and their associated parameters have their analogues in theoretical models of human memory (see Anderson, 1976, 1983; Collins & Quillian, 1969, 1972). It is possible that a general set of considerations may apply to the design of any information retrieval system - whether this system is housed in human memory or in electronic storage. Our understanding of these considerations may then be enhanced through a free interplay between applied attempts to build better information retrieval systems and theoretical attempts to build better models of human memory.

Acknowledgment

I would like to thank Steve Poltrock and George Furnas for their helpful comments on earlier drafts of this article.

References

Anderson, J. R. (1976). *Language, Memory, and Thought.* Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Anderson, J. R. (1983). *The Architecture of Cognition.* Cambridge, Massachusetts: Harvard University Press.

Collins, A. M., & Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior, 8*, 240-247.

Collins, A. M., & Quillian, M. R. (1972). Experiments on semantic memory and language comprehension. In Gregg, L. (Ed.), *Cognition and Learning.* New York: Wiley and Sons.

Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1983). Statistical Semantics: Analysis of the Potential Performance of Key-Word Information Systems. *The Bell System Technical Journal, 62*, 1753-1806.

Ide, E. & Salton, G. (1971). Interactive search strategies and dynamic file organization. In Salton, G. (Ed.), *The SMART Retrieval System - Experiments in Automatic Document Processing.* Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Jones, W. P. (1985a). *On the Applied Use of Computer Models of Memory - A Proposal For a Large-scale Personal Filing System* (Tech. Rep. TR-85-103-04). Austin, TX: Microelectronics and Computer Technology Corporation,

Jones, W. P. (1985b). *A Geometric Analysis of Four Information Retrieval Similarity Measures* (Tech. Rep. HI-138-85). Austin, TX: Microelectronics and Computer Technology Corporation,

Salton, G. & Yang, C. S. (1973). On the specifications of term values in automatic indexing. *Journal of Documentation, 29*, 351-372.

Salton, G. & McGill, M. J. (1983). *Introduction to Modern Information Retrieval.* New York: McGraw Hill.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation. 28*, 11-21.

Wickelgren, W. A. (1976). Memory storage dynamics. In Estes, W. K. (Ed.), *Handbook of Learning and Cognitive Processes.* Hillsdale, New Jersey: Lawrence, Erlbaum Associates.