TEXTNET: A Network-Based Approach to Text Handling

Randall H. Trigg

Mark Weiser

Department of Computer Science
University of Maryland
College Park, MD 20742

ABSTRACT

This paper describes a new system for text structuring called Textnet. The Textnet approach uses one uniform data structure to capture graph-like pools of text as well as embedded hierarchical structures. Using a semantic network formalism of nodes connected by typed links, the relationships between neighboring pieces of text are made explicit.

We also describe our partial implementation of the Textnet approach which makes use of an object-oriented window/menu-driven user interface. Users peruse the network by moving among object menus or by reading text along a path through the network. In addition, critiquing, reader linking, searching, and jumping are easily accessible operations.

Finally, the results of a short trial with users are presented.

1. Introduction

1.1. The Scientific Community and Document Composition

Computer technology has had a tremendous effect on the working lives of scientists over the last two decades. These effects have been most noticeable in the areas of data gathering, storage and analysis. Recently, however, with advances in computer networks and text processing, other activities are beginning to move online. Papers are written using word processing equipment or with text editors and document formatters on mainframes. Correspondence and information trading in the form of electronic mail occurs on various national networks. In addition, electronic journals and teleconferencing are becoming more commonplace.

In our view, the logical and inevitable result will be the transfer of all such activities to the computer, transforming communication within the scientific community. All paper writing,

critiquing and refereeing will be performed online. Rather than having to track down little-known proceedings, journals or unpublished technical reports from distant universities, users will find them stored in one large distributed computerized national paper network. New papers will be written using the network, often collaborated on by multiple authors, and submitted to online electronic journals.

Moving such paper writing activities online requires new concepts in text handling. In order to take fullest advantage of the computer's capabilities, we envision authors composing pools of text able to be organized and interconnected according to the tastes and interests of both author and reader. Indivisible pieces of text would range from notes to subsections of published documents. The computerized paper network would keep track of such pools of text and aid users in augmenting and manipulating them.

The research described here provides a uniform underlying structure for text at all levels. This novel strategy of text organization leads to natural implementations of many important capabilities for future online scientific networks.

1.2. Requirements

The text composition strategy we require and its accompanying computer implementation must be quite flexible. The system should not only help authors compose and interconnect pools of text. It must also be capable of generating traditional linear documents from these pools. Furthermore, online readers must be allowed convenient access to the pool and finally, these features must scale up to the national level. Specifically, our requirements include the following.

Text Perusal: The "simple" act of reading can become quite involved if user friendly interfaces are desired, especially in the case of non-sequential text.

Text Composition and Modification: Text composition is much more than word generation; text systems must actively help the user organize what has been written. This involves linking together primitive pieces of text to form a text network and then imposing various hierarchical structures (tables of contents) on this network to yield papers.

Critiquing and Reader Linking: Though these acts are rarely addressed by text accessing systems, we feel that they form an integral part of any reader's mental activity and we advocate designing systems with built-in critiquing capabilities. In particular, such systems must be able to link comments from readers (e.g. electronic journal referees) directly into the network of text comprising the paper.

Hard-copy Rendering: At least for the foreseeable future the "paper world" is still with us. Thus, computer text systems must be able to generate linear walks through portions of text and be interfaced to or even merged with document formatters.

Needs at the National Network Level: These include among others, a multiple authoring capability, personal literature monitoring for network users, and super-paper hierarchies forming scientific subdomains.

The reader will notice that a major activity of most computer text systems has not yet been mentioned: search and retrieval of portions of text. Certainly, data base retrieval is a necessary part of any such system. However, search issues have often been addressed in the past whereas we have chosen here to confront problem areas such as critiquing which have received to date much less attention in the literature.

1.3. The Textnet Solution

In order to investigate novel strategies for text organization and their ramifications for an online scientific community, we designed the Textnet system. Though presently implemented only at the local level (by a system hereafter called "TEXTNET"), it properly embodies our notion of "pools" of text and (we hope) will one day be scaled up to the multi-computer network level. The name "Textnet" can and should be read in two ways: as a local network of pieces of text, or *chunks*, making up a paper, and as a national network of linked documents comprising the online literature of a scientific discipline. One of the major contributions of this research is the integration of both into one approach. The same structural ideas apply on both levels. In fact, in a projected implementation, a chunk found in a paper and a super-paper node corresponding to an entire subdomain would both be nodes in one comprehensive distributed network.

In devising our approach to text organization we elected to diverge from work in natural language understanding. In spite of recent progress in this area of Artificial Intelligence, we feel that automated content analysis of documents is still many years away. Instead, we store text in such a way as to make its underlying structure explicit. Using such structure, meaning can be extracted from the relationships between chunks (small pieces of text) rather than from the words making them up. For example, one chunk can present support or a demonstration of the arguments in another. A set of three successive chunks can provide a parallel, but more detailed discussion of a topic than another set of similar chunks. An entire document composed of many chunks can serve as a refutation of another paper or of a single chunk.

To capture this inherent structure we have borrowed from well-established knowledge representation work in Artificial Intelligence. Chunks become nodes in a semantic network. The links between chunks carry the important typing information encapsulating relationships between neighboring nodes. A system employing such structures, however, must confront several issues. How are users aided in the composition of such linked text? How do readers browse through such a network? What is the proper user interface? How can hierarchical structures be integrated into the structure of linked chunks?

The underlying Textnet approach as described in this document goes a long way toward answering these questions. The TEXTNET system was originally implemented as a testbed for investigating novel forms of text storage. However, it has seen significant use as an aid for text manipulation. In particular, a Ph.D. dissertation was composed using TEXTNET [Trigg 83]. Parts of other existing documents have also been moved online and translated into Textnet-style networks. Some key features of this implementation follow.

Advanced user interface: The front end to TEXTNET employs overlapping windows to facilitate a friendly menu-driven interface.

Multiple access modes: Browsing a document can proceed in one of two ways. Users can move manually among the nodes stepping from one to the next by following links. In this mode, normally only header information is seen. Alternatively, a "scanning window" can be used which displays the text from several nodes along a linear "path."

Path definition and hard-copy generation: *Paths* allow multiple tracks through the same text depending on the interests of the reader. They can be saved, modified and passed to users for whom the given track is appropriate. Paths are also used by the system to generate hard-copy.

Suspendable, stacked tasks: At any point in the system's operation, the current task can be suspended and the point of view switched to some distant point in the document. This new location can be found using either a table of contents or via a keyword search. Later the task stack can be popped to recover the suspended job.

General purpose chunk format: The system's chunk objects contain the dated signature of the author, a list of keywords, status information, and a pointer to the text. Modification privileges are granted only to the author of the text represented by the chunk.

Convenient critiquing, refereeing and relinking: Through the use of labeled links between nodes, the system allows readers to attach critiques and other forms of commentary as new chunks onto existing nodes.

Integrated tables of contents: Using the same linked structures, hierarchical information in the form of tables of contents can be captured by the network. This enables multiple simultaneous organizations of the same paper.

The next section describes the underlying network structures of Textnet including chunks, tocs, links, and paths. The third section explains how such structures can and should be accessed. The results of a brief experiment with users are presented in Section 4. Section 5 briefly surveys related work in this area and the last section summarizes and concludes. In addition, the appendix shows snapshots of the TEXTNET implementation in action.

2. Textnet Organization

2.1. Overview

The Textnet network is organized using a semantic net-like formalism of labeled links and nodes. There are two types of nodes; *chunks* and *tocs* capturing the textual and hierarchical components respectively. Thus a chunk node represents a primitive piece of text while a toc node corresponds (loosely) to an entry in a table of contents. Effectively, toc nodes comprise the nonterminals of hierarchical DAGs embedded in the network.

Connecting the various nodes are typed (labeled) links. The type of the link is meant to capture the relationship between two nodes.

Note that although the internal structures of tocs and chunks are quite similar (as will be seen), it is important to differentiate between the two. The alternative would be to allow tocs to possess text (and perhaps chunks to have "children"). But then this text becomes bound inextricably to its toc (and thus to a particular hierarchical placement). In Textnet, such a binding, if needed, can be effected simply by linking the text to its toc. Then, later, alternative companion chunks can be proposed and linked to the same toc without disturbing its contents. Furthermore the text of such a companion chunk is critiquable apart from its toc node.

The fourth and final data object known to Textnet is the "path." Paths are simply ordered lists of nodes and are used to browse linear concatenations of text from several nodes, and to dump such scans to hard-copy.

In the following sections we describe in turn chunk nodes, toc nodes and links.

2.2. Chunk Nodes

Chunk nodes are the data objects corresponding to primitive pieces of text. In addition to providing storage for pointers to text, these contain several features. Included are the following:

Author: A unique identifier for the chunk's author.

Date: At least the creation date, but possibly also the dates of last access and modification.

Keywords: A list of words characterizing this chunk.

Name: The author's choice of name for this node.

Status: "Published," "Draft," etc.

Level of Treatment: "Technical," "General," "High School," etc.

Text: Pointer to the file containing text.

In-Links: Links entering this object.

Out-Links: Links emanating from this object.

The text slot of a chunk node contains a pointer to a file containing that chunk's text. We make no restrictions on the size of a chunk. Depending on the application, a chunk could range from one sentence to an entire document. However, we have found that from one to several paragraphs is usually a convenient size.

Often, a chunk that is too large prevents users from linking to one idea within the chunk rather than the whole. In such a case, it may be necessary to divide the chunk into several smaller chunks. On the other hand, chunks that are too small may prompt all linking to be done to a parent toc node. In this case, the author may decide to merge several chunks. Such division and merging could of course be suggested by readers in the form of critiques. In fact, one "quick" way to add an entire document into the network is to include it as a chunk and expect readers to dictate subdivisions[1].

The in-links and out-links fields make the graph structure of the text explicit. These links to other nodes (both tocs and chunks) can be viewed by readers and used to follow paths through the text.

As will be seen in Section 2.4, links can be tagged as *Prerequisite* or *Must-follow*. If one of the in-links is *Prerequisite* then it comes from a node thought to be a prerequisite for understanding the current node. Similarly, if one of the out-links is *Must-follow*, then it points to a node that should be perused eventually (after perusal of the current node).

2.3. Toc Nodes

The hierarchical DAG structures loosely corresponding to tables of contents embedded within the overall graph structure are composed of toc nodes. Toc nodes have the same structure as chunk nodes except that the text field is missing. The main difference between tocs and chunks is that rather than containing one piece of text, a toc can be said to "oversee" many chunks. A connected subnetwork of toc nodes actually forms a directed acyclic graph (DAG) since there can be cross-linking between the various toc hierarchies. The hierarchical relations

1. A better way is to initially divide the paper into tocs based on its table of contents (chapters and sections), and chunks by paragraph. We have implemented such a system which automatically converts Unix troff files into Textnet.

of parent and child among toc nodes are captured using *Child* link types. A toc must have an out-link for each child and, unless it is at the top of a hierarchy, at least one in-link connecting it to its parent.

The in-links and out-links fields are identical for tocs and chunks except that chunks may not have *child* out-links. Both fields can contain hierarchical and horizontal links, where the latter serve as pointers to chunks or tocs not hierarchically related. Depending on the link's type, these horizontal links can include critiques, comments, or links continuing a train of thought like *arguments* and *generalizations*.

For any toc, the child nodes form an ordered list. By making this ordering and the overall structure explicit, readers are able to comment not only on the content of a document but also on its organization. For example, a reader might approve of Chapters 1 and 2 of a paper individually, but feel they appear in the wrong order.

2.4. Links

As we have seen, links are normally used to connect chunk and toc nodes (though they may also point to other links) and as such, appear as in-links or out-links for those nodes. A link makes explicit the relationship between two nodes. This typing information is crucial when dealing with nonlinear text since it allows readers to choose between various "next" nodes to peruse based on the connecting link's type. The main features of a link follow:

Auth/Date: signature as for chunk/toc nodes.

Type: the type of the link (e.g. *Refutation*, *Continuation*, etc.).

Fromobj: the object at the "from" end of this link.

Toobj: the object at the "to" end of this link.

Prerequisite tag: if set, then reading the *fromobj* is a prerequisite for reading the *toobj*.

Must-follow tag: if set, then after reading the *fromobj* one should eventually read the *toobj*.

The real sense of a link is captured by its type. The system is aware of specialized semantics for only one link type, the *Child* link. This link is used to connect toc nodes to their children and thus together with the toc node forms the network's hierarchical structures.

A detailed discussion of link types is beyond the scope of this paper. For now, let us consider the effects of this linked structure on the reader's train of thought. We describe three types of motion: movement from one topic to another *along* the train of thought, a side trip *orthogonal* to the train of thought, and a *fork* to several alternatives.

(1) Movement along the train of thought. Examples include links from a topic to its generalization, from examples and movitating ideas to a formalization, from a theory to its application, or simply from one topic to the next.

(2) Side trips. Taking a side trip is not necessary for all readers, but may be desirable to acquire examples and explanations for novices in the area, and further details for experts. In fact, paths for the same paper often differ precisely in their choices of which side links to include.

(3) Forks. Forks are nodes at which the train of thought divides into one or more subpaths. The best choice of out-link to follow usually depends on the interests and expertise of the reader. Some forks divide only for the space of one node, allowing readers the choice between two versions of the same subject matter. At other times the trails leading from a

fork are more complex and extensive, perhaps never rejoining to form one main path.

All in all, readers move through the network following sequential, possibly forking paths and taking occasional side trips. However, it is not immediately clear how readers differentiate between train of thought links and side links. One solution is the use of fixed ordered lists of nodes, called *paths*. In general, authors create default paths for novice readers to follow (though paths can also be modified and created by readers). In this way, readers are assured that the intended train of thought link is exactly the next one on the path. Links not leading along the path are side links (at least as far as the path's author is concerned). In this way the train of thought is dictated by the path being followed at any given time.

Finally, there is one valuable, but slightly peculiar special case of a link. It is the link whose *fromobj* (or *toobj*) is another link. For example, suppose a reader disagrees with an author's assertion that chunk R refutes chunk C. This objection could be voiced by connecting to the link between R and C a *Refutation* link pointing to a chunk containing the objections. If this reader feels moreover that R provides support for C when viewed properly, he could, in addition, create a new *Support* link connecting C and R and pointed to with a *Comment* link containing an explanation as shown in Figure 2.1.

3. Textnet Access

3.1. Overview

There are at least four major activities performed by users accessing Textnet.

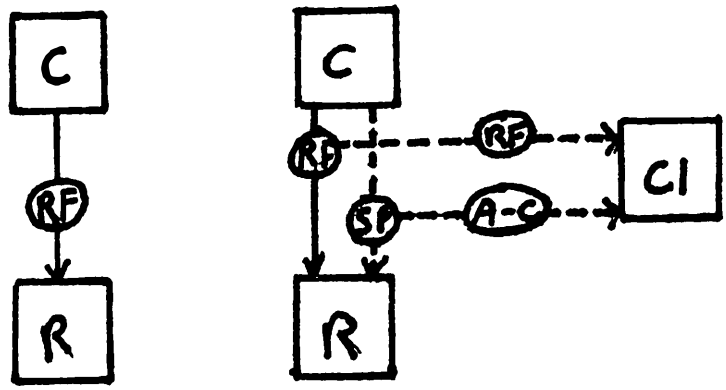


Figure 2.1: Example of links to a link.

Text perusal: This consists of a "manual" mode in which the user selects links to follow and a "path" mode in which either user or system-generated paths are used within a facility called the "scanning window."

Text and structure composition and modification.

Critiquing and Reader Linking: That is, linking comment chunks to nodes and linking together two existing nodes.

Hard-copy Rendering: or using paths to generate sequential walks over a subset of the network.

In the subsections that follow, each of the above areas are discussed in turn. Throughout, we attempt to provide a feeling for the complexities involved and how they lead to certain desirable features for a text system. In addition, we consider how such features are facilitated by the organizational strategy described in Section 2. Finally, the current implementation is briefly described at the end of this section and snapshots tracing the actions of a user perusing a Textnet network are displayed in the appendix.

3.2. Text Perusal

When perusing a printed document, a reader could begin in any of several ways, including:

(1) Start at the first page and work through sequentially.

(2) Start at a particular chapter or section whose location is found using a table of contents.

(3) Using an index of keywords, jump into the text of the document at some point.

And once at the given location:

(4) Skim, say by reading the first and last paragraphs of every chapter or the first sentence of every paragraph.

(5) Jump to other relevant points using the index.

(6) Proceed to a new topic using the table of contents.

(7) Variations and combinations of any of the above depending on the reader's current level of interest, understanding, impatience, frustration, etc. For instance, one quick way to move to an item of interest is to move top-down through a table of contents and then jump between pieces of text at a given level.

When the text is stored online and in a nonlinear format the possibilities increase further. In any case, however, we claim that all these modes of text perusal can be characterized as combinations of three general types of motion through text. We call these *horizontal*, *vertical*, and *non-local leaps*.

A reader electing to continue by moving to a neighboring piece of text (chunk) is perusing horizontally. This neighbor can only precede or follow the current chunk in the case of linear text. Thus, a text system in that case need only be capable of scrolling forward and backward. However, with nonlinear text the next chunk to peruse could be chosen from a potentially large set of neighbors. Systems incorporating nonlinear text must offer a convenient, efficient facility with which the user can specify such choices. Actually, this amounts to a device for interactively traversing a graph or network of nodes.

Those reader activities mentioned above that use a table of contents to move to higher or lower levels of abstraction constitute vertical motion. Readers move up the hierarchy to higher

levels and then down a new branch arriving at possibly distant text. Tables of contents impose hierarchical structures on the graph of text pieces and should be integrated into the system in such a way that their structures are convenient to the user at all times. Note that in printed documents this is not the case. Movement using a table of contents requires substantial page-turning and place-marking.

The third way of moving through text is by arbitrary discrete jumps. In a printed document the index provides a source of target points. As in the case of vertical motion, however, this indexing capability must be integrated conveniently into the system.

One of our primary goals was to make all three modes especially convenient to the reader. This was accomplished by designing a user interface that makes the overall network structure and its embedded hierarchies explicit. Users inspect and manipulate objects corresponding to chunks and tocs. A capability for making discrete jumps to either type of node is also provided. Additionally, the system keeps a history of suspended points in the text to facilitate returns from prior jumps.

Our approach to text organization makes such access modes and facilities feasible. Listed below are several facilities for perusal and a short description of how each follows naturally from our choice of organization.

Manual Movement: In this mode, the user selects at each node the link to follow. This link can lead horizontally to a chunk or toc, or vertically (if the chosen link is of type "child") to a node higher or lower in the hierarchy.

Movement Using Paths: Using "paths" (ordered lists of nodes), users peruse text arranged on the screen in the familiar linear fashion in a "scanning window." Scrolling and other browsing operations are permitted. In addition, the user can modify this path while perusing the text and save the modified version for future use. Default paths are generated by the system when needed. Flipping between this mode and manual mode is largely effortless.

Indexed Access: Indexing is accomplished via the keywords and names present on every node. The system can search through the space of nodes for given patterns of such keys.

Current Location: It is often the case that for systems containing large browsable data structures, users quickly become lost. It is therefore essential that they be aware of the current position in the document at all times. In TEXTNET, users can be shown their current position in the table of contents at any time.

Jumping: Jumping to distant nodes can be accomplished using either the indexing facility or the table of contents to locate the new node.

Suspending: It is often necessary to jump to a distant node temporarily and then resume processing at the original node. For example, one may wish to build a cross-paper link, but only after inspecting the other paper. Our system is able to suspend any current process keeping a stack of suspended processes for later resumption.

Skimming: This involves quick inspection of important features of a node without having to read the entire contents.

User Modeling: We claim that the semantic network formalism described in the last section is particularly well-suited to user modeling. User modeling requires mapping out some cross-section of the network as relevant to a given reader. This can be important when generating default perusal paths. Of course, users can always make other choices at each node, but the default path gives them a starting point and enables smooth perusal through most of the text. The system's choice of starting path can take into account any relevant features of the user. For instance, readers could provide information to the effect that they desire a quick walk-

through excluding extended examples and justification. By inspecting link types, the system could tailor the default path to those desires. TEXTNET currently does not do this.

It can be claimed that other organizational strategies also offer different perusal paths for different users. For example, in Figure 3.1, we compare Textnet's strategy with a hypothetical competitor. In the depicted scenario, following perusal of T1, the system moves the user to one of T2a or T2b, and finally to T3. In the approach on the right, the text for both T2a and T2b is stored together in a single node, while in the Textnet approach the two pieces are stored in separate nodes. While it is true that both approaches work in this case, future rearranging of the pieces, addition of a new T2c piece of text, or deletion of a piece of text from T2 causes problems in the alternative strategy. All these modifications are easily handled by Textnet, however, simply through appropriate manipulations of the relevant chunks. It should be noted that although our approach uses slightly more storage (for the extra node structure), this is small when compared to the storage required for the text itself. (For example, the total non-textual TEXTNET storage requirements for a large dissertation [Trigg 83], including names, keywords and signatures of nodes, amounts to approximately one fifth of the storage required for the raw text. This ratio, though it depends on the degree of linking activity, should be even further reduced in future implementations.)

3.3. Structure Composition and Modification

We expect authors to initially create and subsequently modify the text of their chunks using a standard text editor. Though perhaps the perfect text editor has yet to be designed, many of the existing ones are more than adequate for our purposes and so we don't address

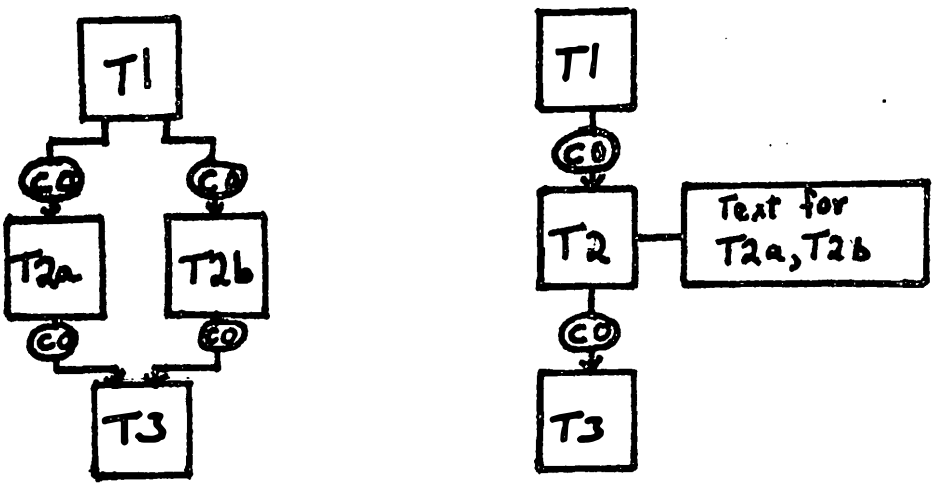


Figure 3.1: A forking path through text.

such issues. Rather, we are concerned with problems of structure composition and modification. Once the author has created some chunks, how can the system help organize those chunks hierarchically? In addition, how can the system help subdivide an existing chunk or merge several chunks together?

Overall, modes of structure modification consist of at least the following:

Node creation: New nodes can be created either during the process of critiquing or as children of an existing toc.

Node deletion: In fact we unlink nodes (remove an In-Link) rather than deleting them. Only if the unlinked node has no remaining In-Links does it become garbage and vanish.

Node reordering: The children of a toc node can be rearranged.

The above three operations are of the type required of any advanced structure editor. Of special interest though, is

Chunk subdivision: If it is found that over a period of time, a predominant number of links to a chunk are to particular places in the chunk's text rather than to the chunk as a whole, the author can elect to subdivide the chunk. One way to do this is by dividing the old chunk into several pieces connected with *Continuation* links (Figure 3.2a). Or a toc node T could be created with the new chunk pieces as children (Figure 3.2b). Old links pointing to the chunk as a whole now point to the new parent T (or to the first node in the *Continuation* chain), while other links point to appropriate children (or to later continuations). Alternatively, a more complicated structuring under T could be constructed. For example, certain of the subchunks could be linked to one another rather than as children of T. Also, further nesting using other new toc nodes under T might be desired (Figure 3.2c).

3.4. Critiquing and Reader Linking

One of the primary goals of this research is to capture a reader's critique of a document as an explicit part of the document's structure. Because of our choice of underlying structure, such encapsulation of critiques is quite simple. First, we define a few terms. By *critiquing*, we mean the act of creating a new node and linking it to an existing node. Thus critiquing is one way in which the network of nodes is augmented. *Reader linking*, rather than creating a link to a new node, creates a new link between two existing nodes. Thus reader linking adds structure to the network without adding nodes.

Though the act of augmenting the network (i.e. critiquing or reader linking), is simple, the purposes it can serve are numerous.

(1) *Comments on text*: These comments, both pro and con, function as feedback to the chunk's author. They are usually not intended to be included in the final published document as is. Rather, the author takes such feedback into account and may revise the text accordingly. Sometimes, as in the case of a *Rewrite* link, the critiquer will form a copy of the original chunk and edit that. This modified copy then serves as the critique.

(2) *Comments on structure*: These serve to offer criticism and possibly alternate formulations of a document's structure and are usually critiques of a toc node. By copying a toc node of a document and then rearranging its children or otherwise modifying it, the reader can convey an alternative organization of the text.

(3) *Links to side paths*: These paths can cover examples, asides, and other minor diversions from the current train of thought. The resulting chunks may or may not be included in

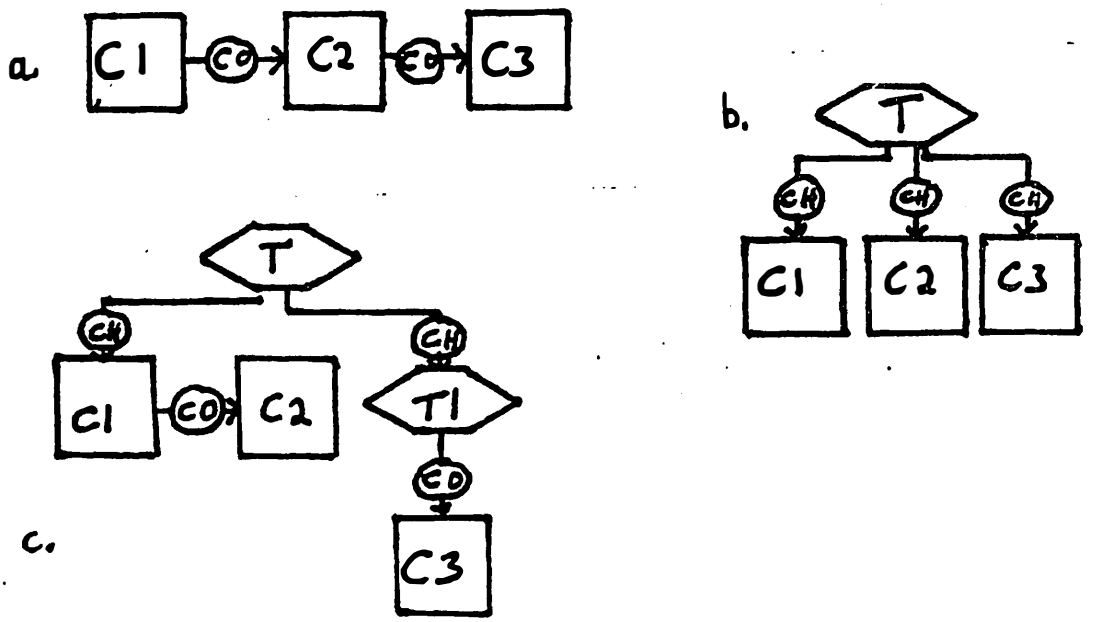


Figure 3.2: Chunk subdivision.

hard-copy versions, but are certainly part of the online document.

(4) *Reader linking*: Here, no new node is created, only a link between existing nodes. This is especially useful for conveying to future readers new relationships between existing nodes. For example, a user reading one document who is reminded of another can capture that recollection instantly by explicitly linking the two documents together.

(5) *Critiques of links*: These allow feedback to the author of a link. For example, a reader might feel that a link labeled as a refutation actually serves as support when viewed properly. (For an example, see Section 2.4.)

The important aspect of all forms of critiquing is that the critique is embedded in the network immediately. Rather than readers jotting down notes that may later turn into mail to the author (or simply be forgotten), here users state their case at the time of reading and within the proper context.

3.5. Paths, Hard-Copy, and User Modeling

The notion of a "path" or recorded trace of a reader's movement through online text has been present in the literature for some time [Bush 45]. For us, a path is simply an ordered list of network nodes. In addition, paths have signatures and names. There are two types of paths: those created implicitly by users as they peruse text, and those created by the system.

Before perusing text in the scanning window, a reader is always given a starting path. This avoids the need for explicit link selection when moving between nodes. Rather a reader must explicitly choose a link only when the default path is undesirable at that point. The current path is modified accordingly as new choices are entered. At any time, the user can save this current path to reuse later or pass to other readers desiring a similar walk through the given material.

Another important use of these path objects is for generating printed copy. Because a path records a linear walk through a portion of the text, it is ideal as a source for hard-copy. The path (often a toc's default path possibly modified by the user) can be turned over to a document formatter to convert toc nodes into table of contents entries and chunk nodes into portions of text. Thus, the same underlying set of nodes can be used to generate several papers depending on the path chosen. This is particularly useful when tailoring a document to a given readership.

The system's choice of default starting path in the absence of an overriding choice by the reader is quite important. If the system possesses partial models of its readers it could choose an appropriate starting path for each user. Such models include level of interest and expertise in the relevant area as well as available time. These features are then used to select the quality and quantity of side paths to follow.

The flexibility offered by the above approach does present one problem, however. If the text of a chunk can appear in several places depending on the path chosen, readability may suffer. How can one provide smooth transitions between pieces of text when the pieces may appear in other arrangements? We don't have a complete answer to this at the present time, but it should be noted that the problem is far less noticeable when reading the document online. Here, the type of the link connecting two nodes provides a kind of transition between them. "Prerequisite" and "Must-follow" links (see Section 2.4) can assist in constructing meaningful chunk orderings for a path.

3.6. The TEXTNET Implementation

The current implementation of TEXTNET is in Franz Lisp on a Vax 11/780 running Unix(tm). Both the programming style and the user interface are object-oriented. By using a window package designed at the University of Maryland [Weiser et al 83], we can associate one or more windows with most internal objects such as chunks and tocs. (See the appendix for snapshots of the user interface.)

Most of the features discussed in this paper have been implemented. Future versions of TEXTNET will additionally feature increased efficiency, improved facilities for scanning text along a path, chunk subdivision, the ability to link to a link, and possibly some rudimentary user modeling.

4. Experience with Users

In order to test the Textnet approach and the TEXTNET system, we performed a short trial with users. The trial lasted 12 days using 16 users. In all, almost 25 hours were logged online with TEXTNET. Each user was required to fill out a questionnaire and logs of time

spent in various TEXTNET activities were kept automatically. Users were largely free to play with TEXTNET as they wished and all had experience on Unix. However, the percentage of online time they normally spent in paper writing varied widely between 0 and 90% (with mean 36%, median 25%).

For the purposes of the trial, TEXTNET was augmented in several ways.

(1) One (changing) pool of text was kept. Each user could thus browse all existing nodes including new nodes created by users in previous sessions.

(2) Because of (1) and the need to prevent corruption of the pool, TEXTNET permitted only one user at a time.

(3) Logs were automatically kept of every TEXTNET session. Logon and logoff times were captured as well as the amount of clock time in each of several TEXTNET activities (scanning, chunk browsing, editing, etc.).

(4) A new command was implemented allowing users to edit their online copy of the questionnaire at any point in the operation of TEXTNET. Users could also work on their questionnaire after exiting the system.

(5) Users were able to suspend their TEXTNET session at any time and send electronic mail to the TEXTNET authors. Such mail usually consisted of comments and complaints inappropriate for the questionnaire such as bugs in the implementation or confusion about the documentation.

(6) The system's pool of text was initially seeded with three "papers." The first was a user's guide to TEXTNET geared for the critiquer. It contains a brief overview of the Textnet design philosophy and about 10 pages of instructions detailing the use of the various commands. (This guide was composed on TEXTNET.) Several nodes outlining more advanced topics omitted from the guide were gathered together to form the second "paper." Finally, a portion of Weizenbaum's book "Computer Power and Human Reason" [Weizenbaum 76] was included as the third "paper." The text we chose comprised in large part Weizenbaum's portrayal of the "compulsive programmer."

Before signing on to TEXTNET for the first time, users were asked to attend a short demonstration by the author. They were also given a hard copy of the critiquer's guide. The only requirement placed upon them was that by the end of the trial period they should have signed on at least once and filled out their questionnaire.

Most users rated the online help facility of Textnet very highly, although the number of uses of '?' per hour varied from 0 to 21.2 with mean 3.9 and median .9.

Use of the scanner dominated virtually all of the sessions. This is natural since it was usually the only way to view a chunk's text. In their questionnaires, users often reacted to the scanning window, citing its lack of flexibility. Some simply wanted more commands for moving about in the scrolling area. Others felt a need to inspect the text of critique nodes without necessarily splicing them into the current path, which the current system does not permit[1]. Also confusing to first time users were the links lines appearing in the scanning window separating nodes in the path. Ideally, these should be removed to enable a smooth rendering of the text appearing along a path. The links information for the current node (in whose text the

1. One solution to the problem uses a new companion window appearing next to the scanner, into which the text of the node to be inspected can be temporarily placed (along with header information). The node can then be spliced into the path exactly when it's explicitly needed.

cursor resides) could be kept up to date in a separate window. The problem of displaying and modifying linear walks through graph structures is inherently difficult and continues to be an interesting research topic.

Among users who created new nodes there was an unexpected clear division between those whose new nodes came from critiquing and those who created new papers (and thus whose new nodes were all either tocs or child chunks of tocs). As it turned out about half of the users, including some of the heaviest, created no new nodes. The remainder either critiqued or wrote papers, but not both. We suspected at first that perhaps the demonstrations had been misleading, implying to some that certain operations were somehow preferred. This turned out not to be the case. We asked follow-up questions of the users who didn't critique and found that (as in preferences for help facilities) this was a matter of personal preference. Some users wished only to peruse what others had written, some perused and commented, and others created anew.

Among specific positive points cited by users in their questionaire comments and mail were the advantages of toc nodes for text organization, the critiquing facility, the pushdown stack of tasks and windows, <esc> as the universal bailout, and command consistency across different menus.

Negative comments about screen format appeared at each extreme: one user wanted at most one window visible at any time to reduce clutter, while another wanted all active windows visible. Both styles of user interface could be accomplished through some form of user modeling.

Several capabilities we knew were missing from the current implementation (and had planned for the future) were also noted by our users. These include the need for a chunk division/merging facility and a smoother interface to a document processor. One common complaint concerned the inefficiency of the searching mechanism. Other suggestions included: keyboard macros, automatic division of non-textnet files into chunks (and tocs),[1] user-defined links, and graphical representations (on suitable terminals) of node structures.

5. Related Work

5.1. Advanced Text Processing

In the last few years there has been a surge in the number of text processing programs and systems available and a renewed interest in text processing as a valid area of research [Meyrowitz & van Dam 82]. These new systems range from text editors and document formatters to integrated programs formatting documents dynamically during the editing session. There are also syntax directed editors for programs and general hierarchy editors for documents, programs and graphics. Finally, there are a few systems which encourage the user to organize his text in a truly novel way. The system we are proposing fits into this latter class. Nonetheless our network accessor has traits in common with general hierarchy editors and we mention these first.

1. Automatic division of non-textnet troff files into chunks and tocs has since been implemented.

Hierarchy editors are interactive programs that peruse and modify structures organized in a tree-like fashion. Some are meant specifically for document editing such as Walker's Document Editor [Walker 81] and PEN [Allen et al 81]. Others like Fraser's sds [Fraser 81] are general purpose tree editors. Some of these systems ([Walker 81] and [Reid & Walker 80]) also include facilities for cross referencing. Though cross referencing implies inter-structural linking, the links themselves are not explicitly present in the hierarchical structure. Two other tree-oriented systems are ZOG [Robertson et al 79] and INFO which is part of the EMACS editor [Stallman 81].

Perhaps surprisingly, two major systems emphasizing structural linking along with tree manipulation were conceived and at least partially implemented well before any of the above editors and document formatters. Reports in the literature from both Engelbart working at SRI's Augmented Knowledge Workshop on NLS (now Augment), and Carmody, Nelson and others on a Hypertext editing system appeared as early as the 1960's ([Engelbart & English 68, Carmody et al 69]). Both of these groups acknowledge their debt to an earlier landmark paper by Vannevar Bush [Bush 45] introducing the concept of a "memex." This mechanized private file would contain various "associative trails" recording paths its owner had taken through a set of records and would allow both browsing and new document creation. Nelson's hypertext as described in [Nelson 67] and later extended in the Xanadu project ([XOC 83]) is in many ways a computerized implementation of Bush's memex.

Engelbart's Augment system ([Engelbart et al 73] and summarized in [Uhlig et al 79] and [Seybold 78]) though also acknowledging the importance of Bush's work, emphasizes the tree structure of its textual information and in this way resembles the hierarchy editing systems already discussed. But Augment goes beyond such editors to incorporate facilities for cross-linking to other documents and saving of traces or paths.

The Textnet system as proposed here incorporates elements of both Hypertext and Augment. Unlike Hypertext, we do have a tree-like component, but unlike Augment, a graph-like network is at the core. Rather than adding a linking capability to a hierarchy of nodes, both linking and hierarchical organizing become natural products of one uniform underlying structure. Child nodes in the DAG (directed acyclic graph) are joined to their parents by links (of a certain type) in the same way that critiques or side paths (or hypertext "branches") are linked together. Though linking to a document from other documents can be accomplished in both Hypertext and Augment, Textnet's uniform linking facility works equally well for bibliographic references, collaboration of multiple authors, referee critiquing, and reader feedback. Furthermore, because the hierarchical structure is built on the network of nodes, a reader can critique the structure (or organization) of the document as easily as he critiques the contents of a text node. In fact, several organizations for the same underlying text can exist simultaneously. A more detailed comparison of Textnet with Hypertext and Augment is provided in Table 5.1.

5.2. User Interfaces

Though designing an advanced user interface has not been a major thrust of this research, we nonetheless confronted the interface problem when designing the TEXTNET front-end. We chose to use a menu-driven window'ed user interface. This work grew directly out of the ISPACE project [Rieger et al 81] and makes use of the "cluttered desktop" format of overlapping windows [Weiser et al 83] similar in many ways to the window interface used by the Lisp Machine [Weinreb & Moon 81]. This reduces somewhat the classic problem of "getting lost"

Feature	Hypertext/Xanadu	NLS/Augment	Textnet
Hierarchies	None	Single	Multiple
Graph-based	Yes		Yes
Link-types	planned		Yes
Links to subparts	Yes	Yes	Proposed
Built-in critiquing	?	Yes	Yes
Reader linking	?		Yes
Paths	Proposed		Yes
Versioning	Yes		
Identifiers	Tumblers	Hierarchy #'s	Names/Keywords
Keyword Searching	?		Yes
Special purpose editor	Yes	Yes	
Distributed network	Proposed	Yes	Proposed
Multi users	Proposed	Yes	Proposed
Teleconferencing	Proposed	Yes	Proposed

Table 5.1: Comparison of three systems.

often encountered by users when browsing large databases of textual information. The ZOG system for one, ran into this problem partly because only one text window ("frame") could be on the screen at a time [Robertson et al 79]. For further justification of this window/menu interface style see [Wood 82].

Another important example of an object-oriented interface is the Smalltalk system [Ingalls 78]. In fact, some of the original Smalltalk authors are now proposing "Dynabook," a futuristic computer the size of a notebook serving all the information needs of its user [Kay & Goldberg 77].

5.3. The Automated Encyclopedia

An idea closely paralleling the notion of an online scientific community is the automated encyclopedia. Though it has been investigated mostly by those in the field of library and information science, the automated encyclopedia researchers share many of the same needs, problems and ambitions as those computer scientists investigating national document networks.

One of the first major proponents of these ideas was H. G. Wells. In 1936, he gave a landmark talk on the "World Encyclopedia" [Wells 38]. Bohnert and Kochen extended Wells' ideas, outlining plans and requirements for a centralized "master encyclopedia" [Bohnert & Kochen 63].

A further in-depth look at the need for an automated encyclopedia together with a set of functional requirements was presented by Soergel [Soergel 77]. His encyclopedia consists of "data elements" representing the text of actual documents, but in a condensed form in order to reduce redundancy. We feel that an environment for users in which they can freely submit articles, commentaries and critiques would more likely be accepted by the scientific community.

Textnet's capability for linking to existing text would reduce redundancy in any case.

5.4. Electronic Journals and Teleconferencing

Work in electronic (or online) journals has begun in several places, though widespread acceptance is still to come. Two extensive experiments in electronic journaling and teleconferencing are of special interest because their results came from testing implemented systems on "real" users. These are EIES (Electronic Information Exchange System) at the New Jersey Institute of Technology [Hiltz & Turoff 81], and BLEND (Birmingham and Loughborough Network Development) in Great Britain [Shackel 82].

EIES provides a Notebooks subsystem which bears a resemblance to our Textnet composition and critiquing facility. However, we provide a uniform approach to such personal and group writing, from "notebooks" through drafts and to final published articles.

The BLEND system is primarily geared toward exploring the use of a computer network for electronic journals. Our emphasis with the Textnet project has been on developing adequate data structures for critiquing and refereeing rather than addressing the (important) logistical details of bringing up such a system in the large.

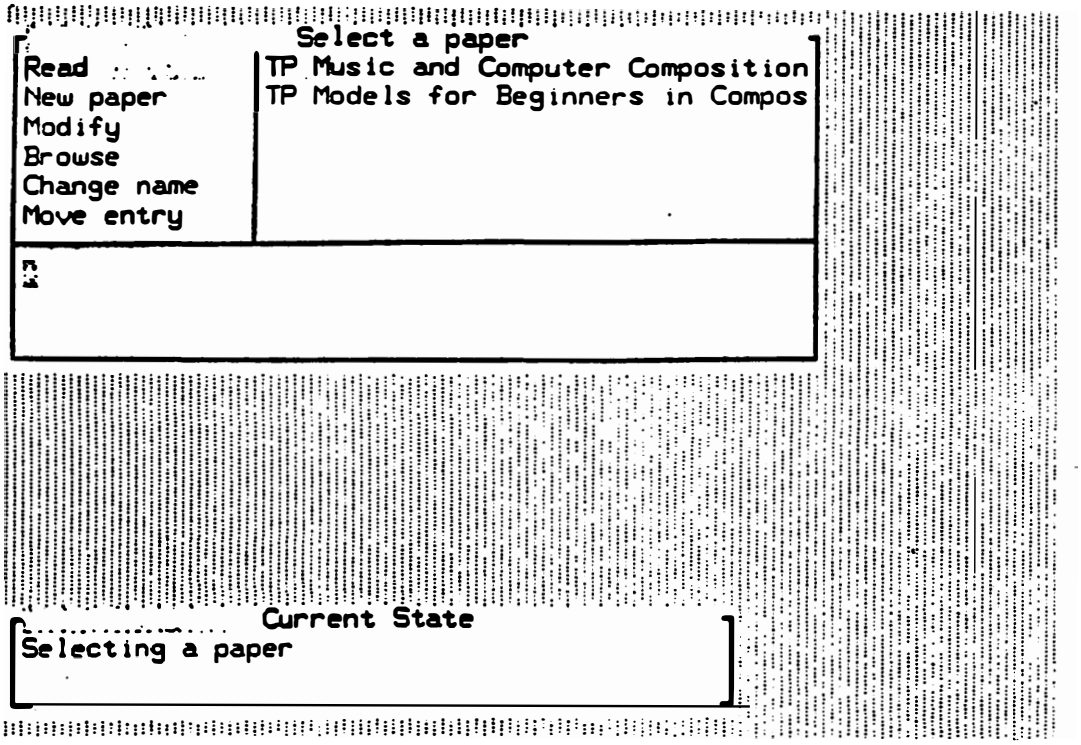
6. Conclusion

The day of the online scientific community is fast approaching. To pave the way, we must develop systems that meet our future communication and text handling needs. This paper has described one promising approach to text organization and manipulation providing a framework for meeting those needs.

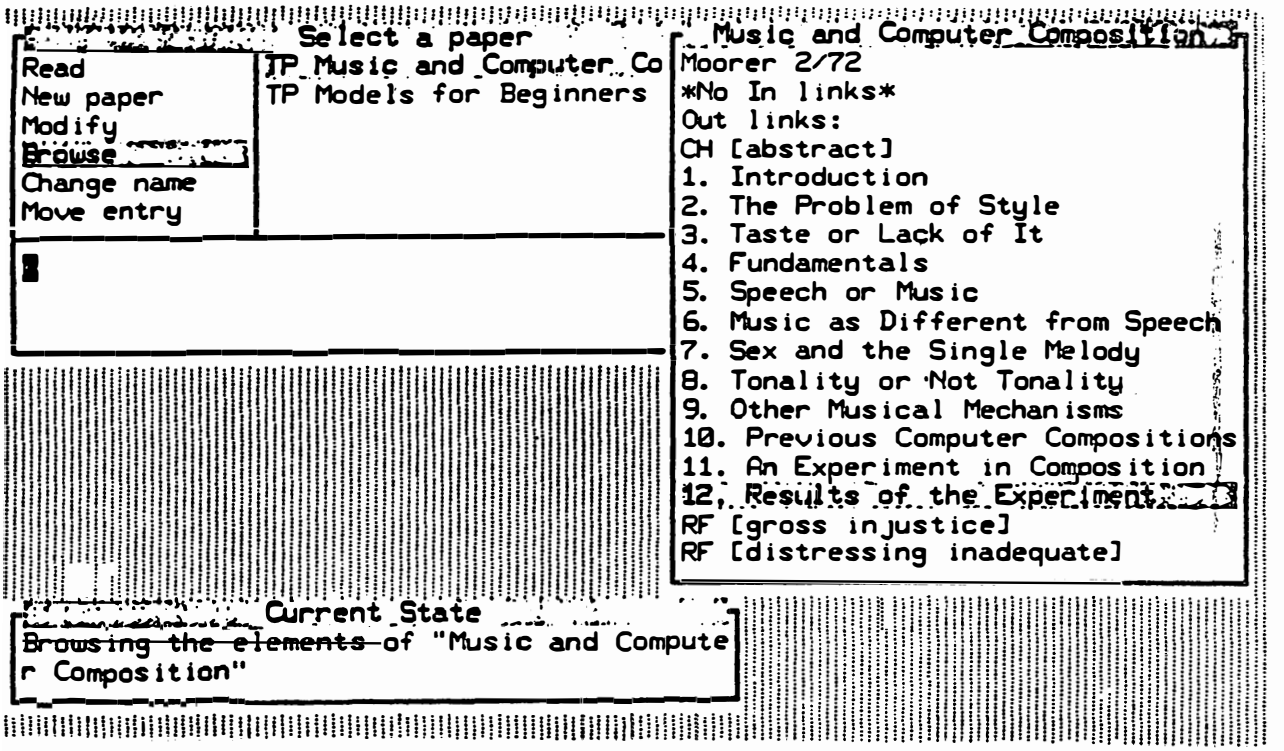
Though not covered in this paper, the structuring ideas inherent in Textnet scale up naturally to a distributed document network setting. This in turn leads to important capabilities for the future online scientific community including paper supersets, personal literature monitors, online publishing and multiple authoring. For further discussion of these ideas and others, see [Trigg 83].

Appendix The following figures trace the actions of a TEXTNET user. The network of text was taken from a paper by Moorer, a critique by Smoliar and Moorer's subsequent reply [Moorer 72a, Smoliar 72, Moorer 72b].

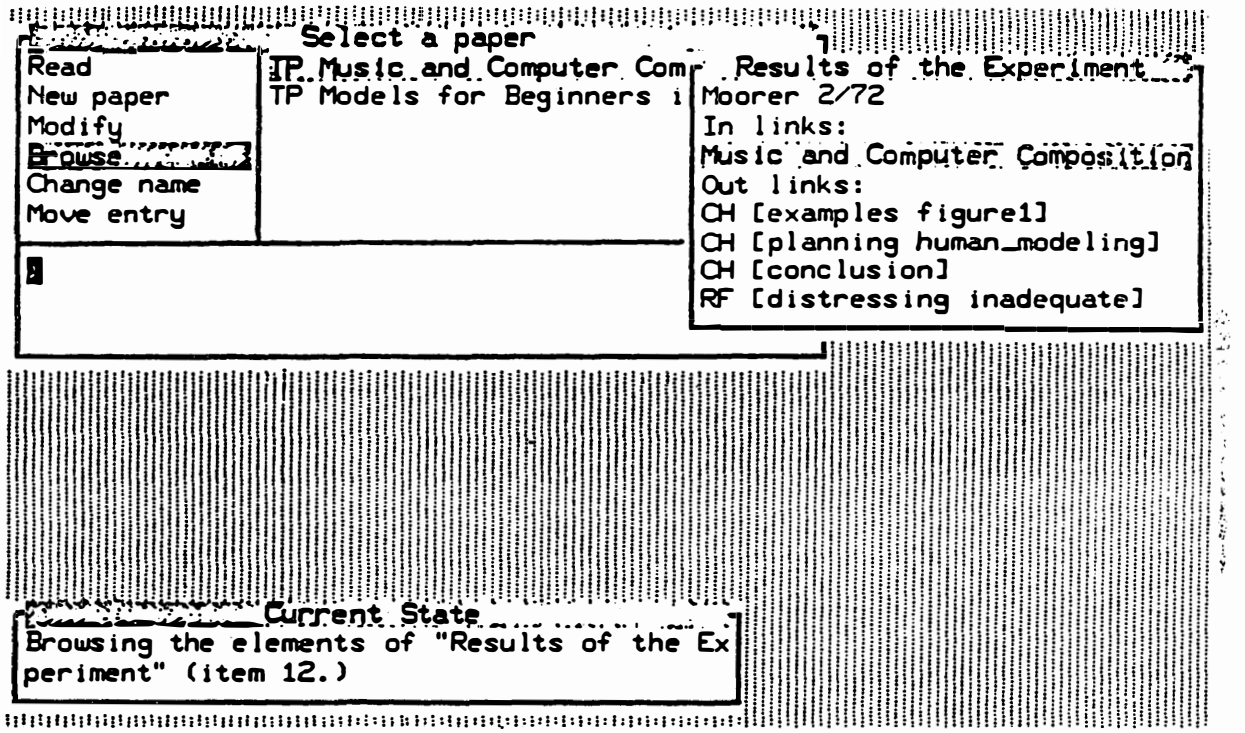
The initial state of TEXTNET showing two papers.



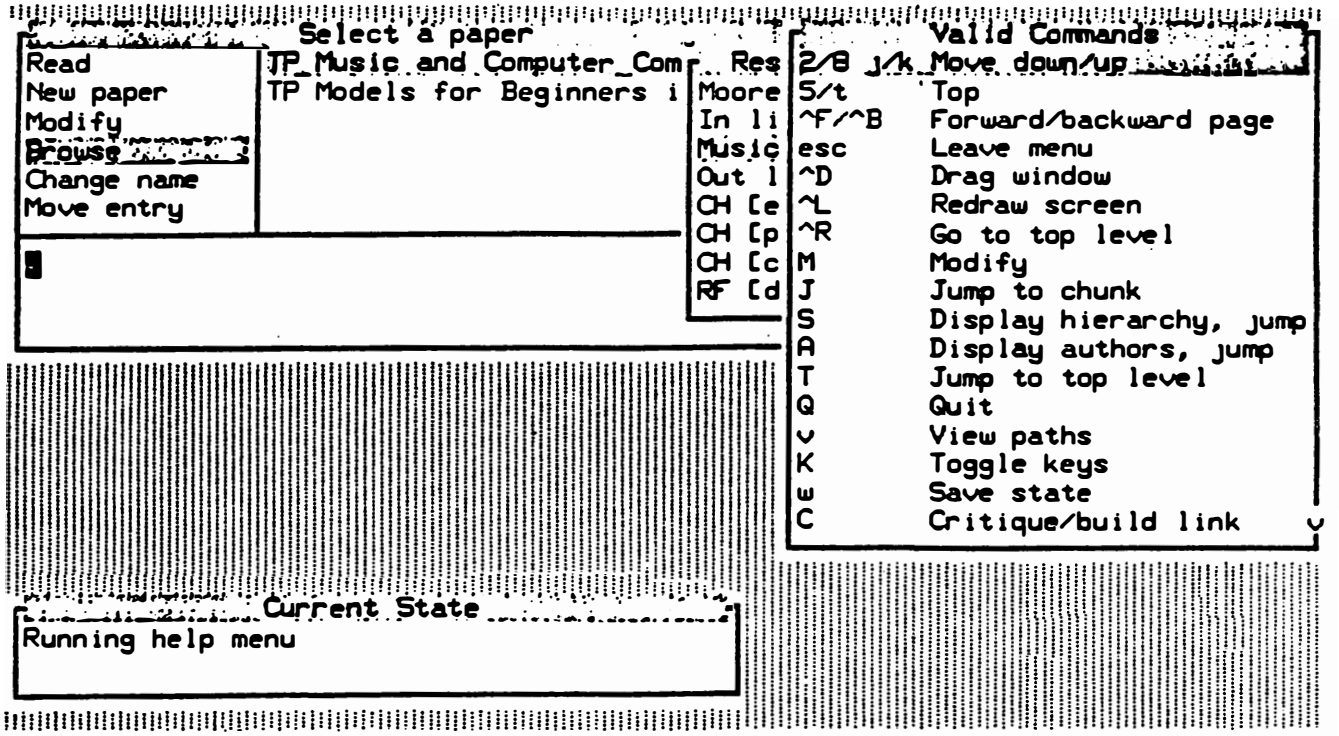
Browsing the first paper.



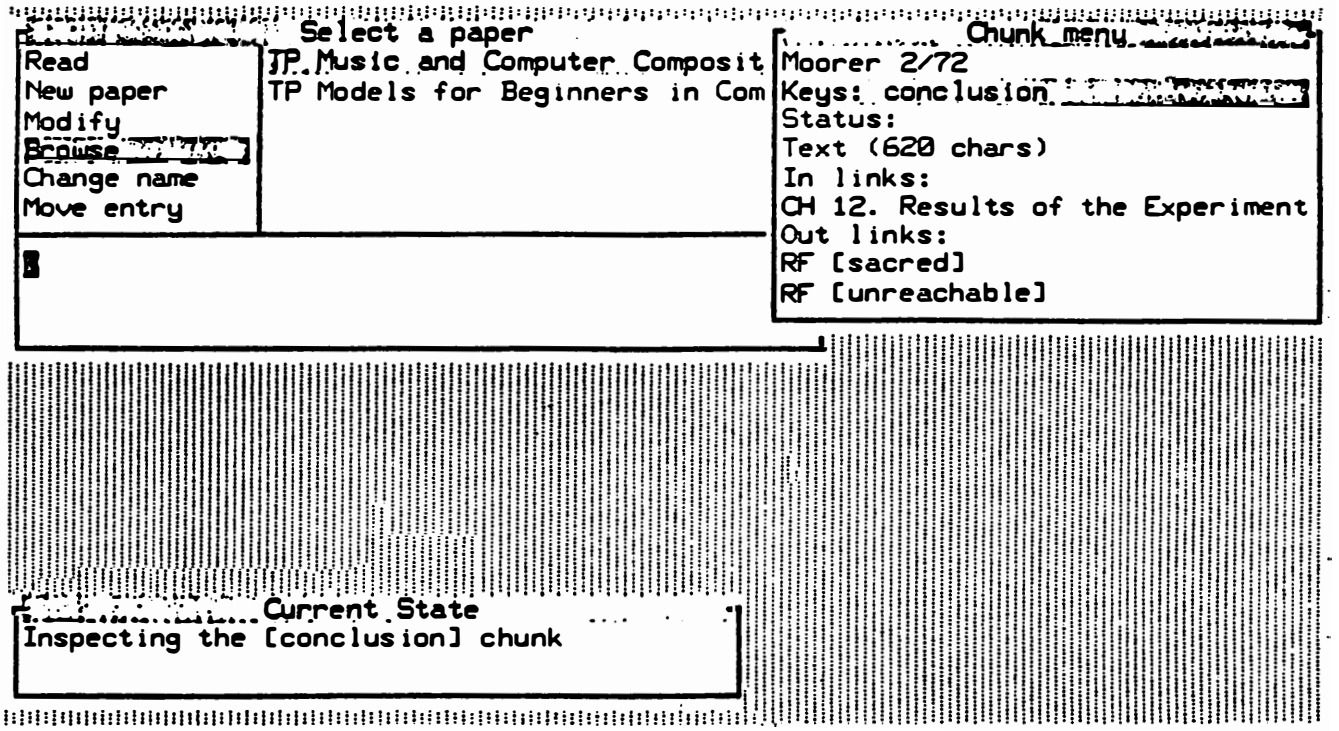
Browsing Section 12's toc node.



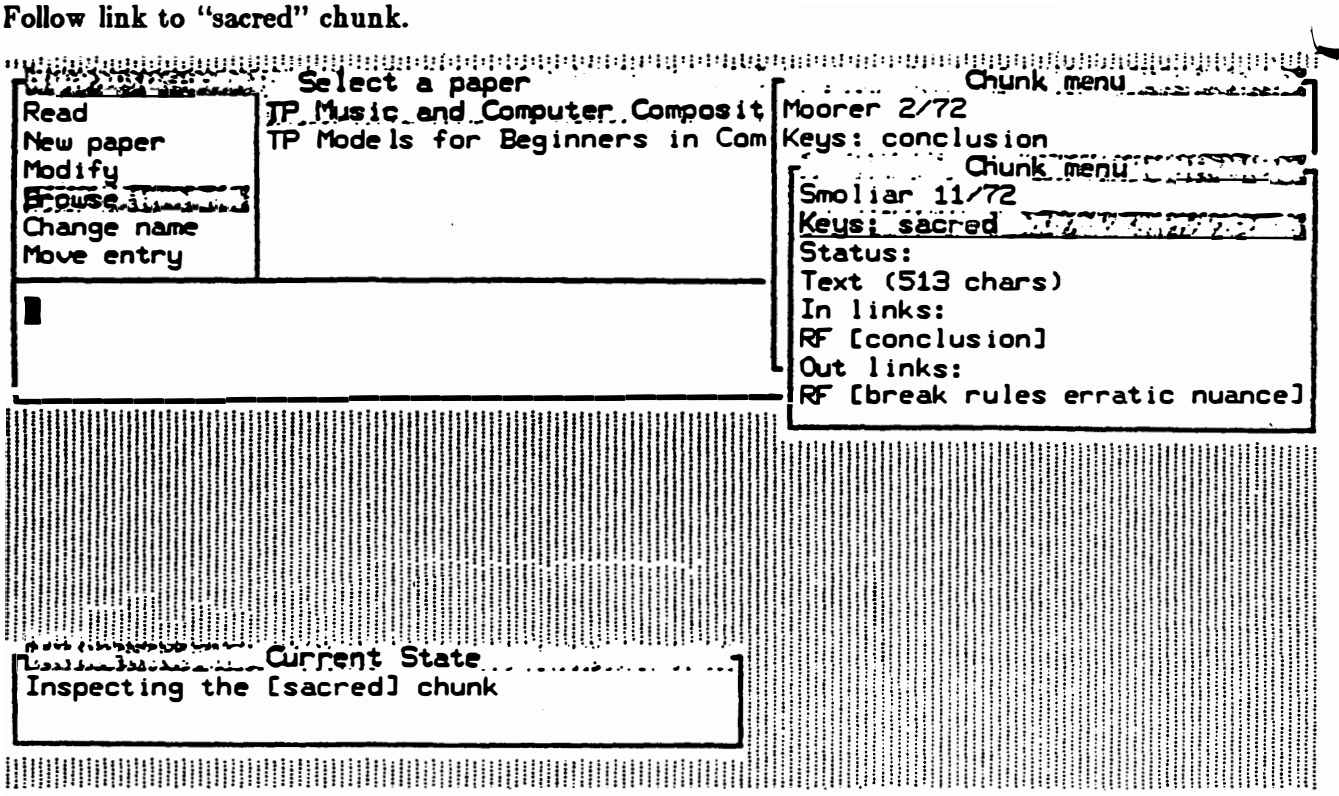
Inspecting the help menu.



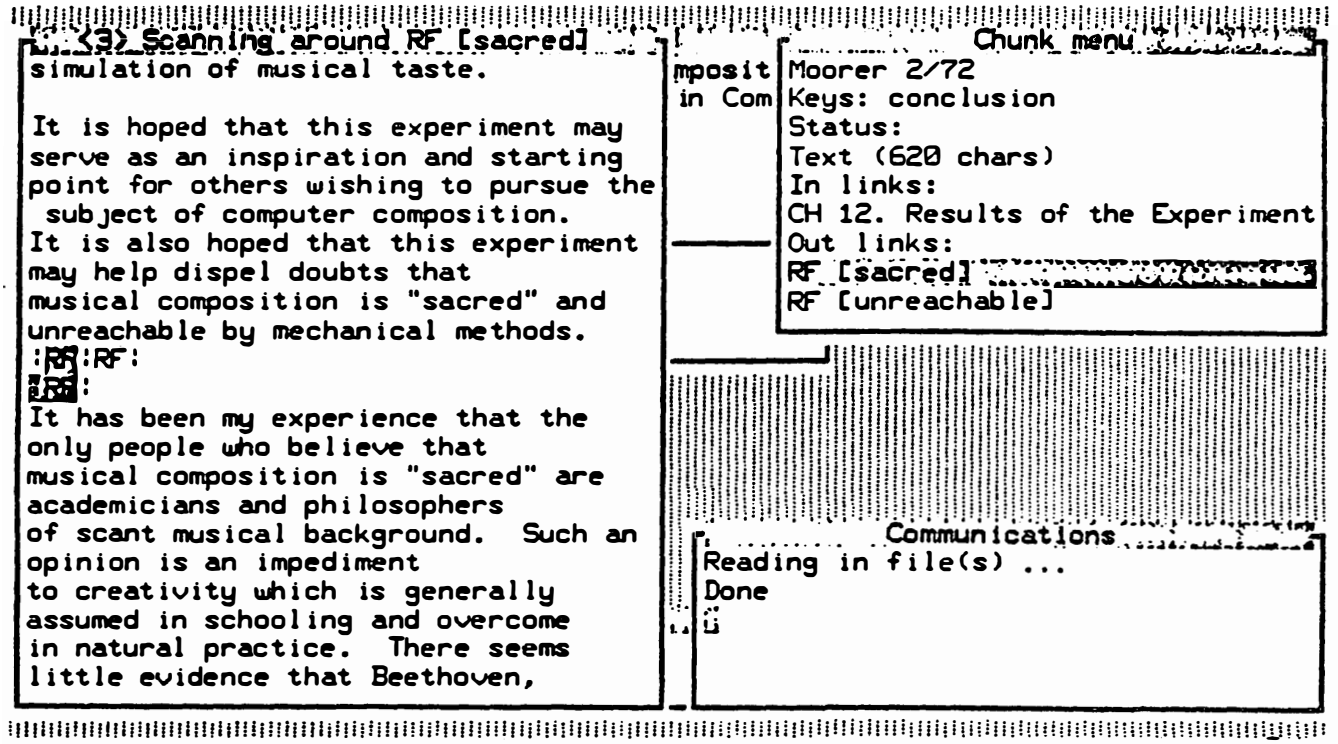
Browsing the "conclusion" chunk.



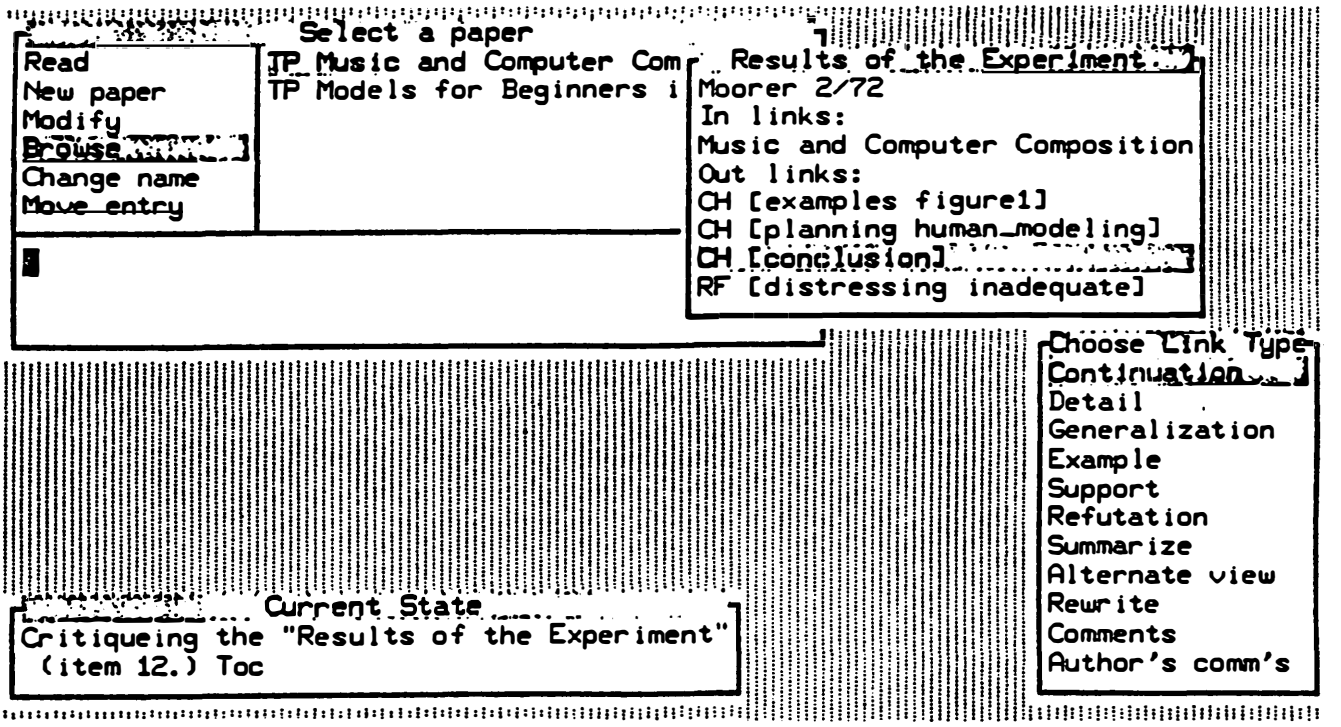
Follow link to "sacred" chunk.



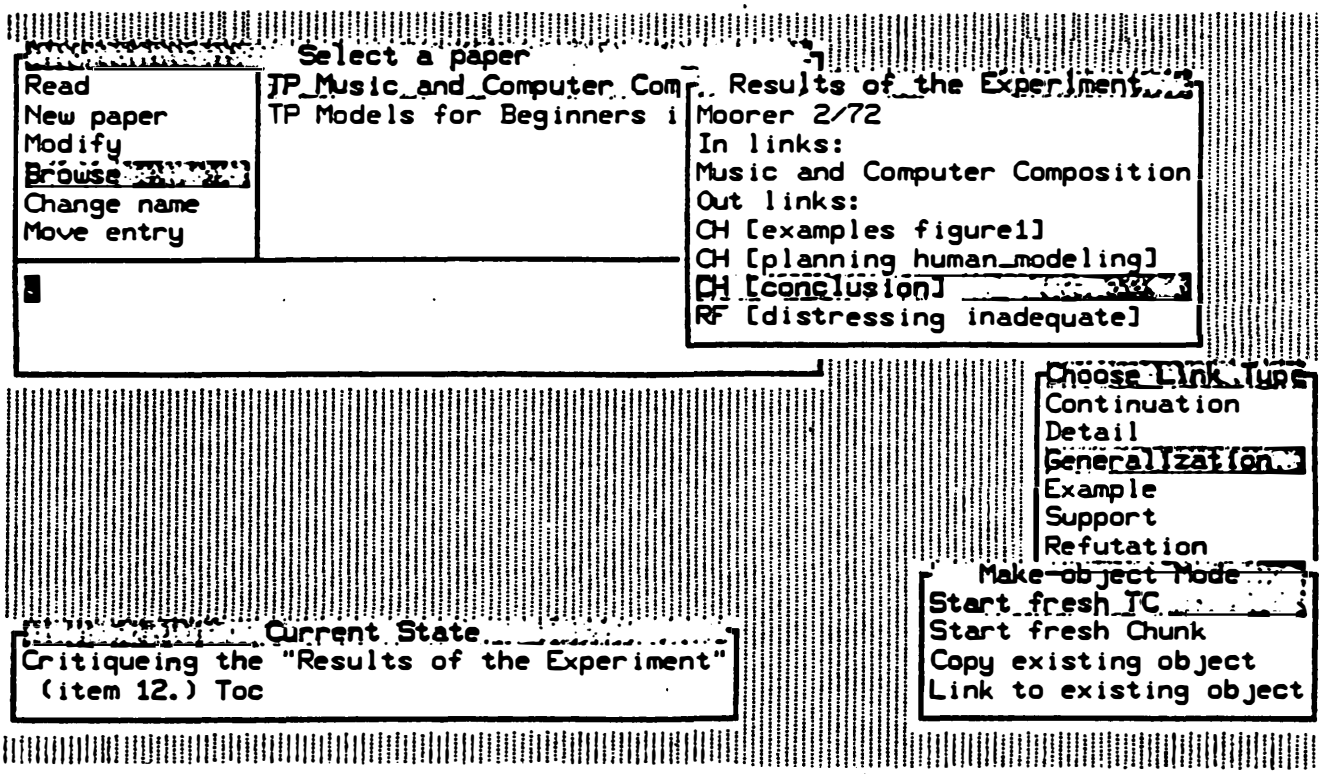
Scanning text around "sacred" chunk.



Building a new link: choosing its type.



Choosing what to link to:



REFERENCES

[Allen et al 81]
Allen, T., R. Nix, and A. Perlis, PEN: A Hierarchical Document Editor, *Proc. ACM SIGPLAN/SIGOA Conf. Text Manipulation*, Portland, Ore., pp. 74-81, June 1981.

[Bohnert & Kochen 63]
Bohnert, H.G. and M. Kochen, The Automated Multilevel Encyclopedia as a New Mode of Scientific Communication, *Proc. of the American Documentation Institute*, pp. 269-270, Oct. 1963.

[Bush 45]
Bush, V., As We May Think, *Atlantic Monthly* **176**, 1, pp. 101-108, July 1945. (Reprinted in: Kochen, M. (ed.) *The Growth of Knowledge*, John Wiley & Sons, Inc., New York, 1967, pp. 23-35.)

[Carmody et al 69]
Carmody, S., W. Gross, T. Nelson, D. Rice, and A. van Dam, A Hypertext Editing System for the /360, in *Pertinent Concepts in Computer Graphics: Proc. 2nd Annual Conference on Computer Graphics*, ed. M. Faiman & J. Nievergelt, U. of Illinois Press, Urbana, Ill., 1969.

[Engelbart & English 68]
Engelbart, D. C. and W. K. English, A Research Center for Augmenting Human Intellect, *Proc. Fall Jt. Computer Conf.* **33**, Arlington, Va., pp. 395-410, 1968.

[Engelbart et al 73]
Engelbart, D. C., R. W. Watson, and J. C. Norton, The Augmented Knowledge Workshop, *Proc. National Computer Conf.* **42**, Arlington, Va., pp. 9-21, 1973.

[Fraser 81]
Fraser, C.W., Syntax-Directed Editing of General Data Structures, *Proc. ACM SIGPLAN/SIGOA Conf. Text Manipulation*, Portland, Ore., pp. 17-21, June 1981.

[Hiltz & Turoff 81]
Hiltz, S. R. and M. Turoff, The Evolution of User Behavior in a Computerized Conferencing System, *CACM* **24**, 11, pp. 739-751, November 1981.

[Ingalls 78]
Ingalls, D.H.H., The Smalltalk-76 Programming System Design and Implementation, in *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, 1978.

[Kay & Goldberg 77]
Kay, A. and A. Goldberg, Personal Dynamic Media, *Computer* **10**, 3, pp. 31-43, March 1977.

[Meyrowitz & van Dam 82]

Meyrowitz, N. and A. van Dam, Interactive Editing Systems, *Computing Surveys* **14**, 3, pp. 321-415, Sept. 1982.

[Moorer 72a]
Moorer, J.A., Music and Computer Composition, *CACM* **15**, 2, pp. 104-113, Feb. 1972.

[Moorer 72b]
Moorer, J.A., Reply by Moorer to Smoliar's Comments on Music and Computer Composition, *CACM* **15**, 11, p. 1001, Nov. 1972.

[Nelson 67]
Nelson, T.H., Getting It Out of Our System, in *Information Retrieval: A Critical View,* ed. G. Schecter, Thompson Books, Washington, D.C., 1967.

[Reid & Walker 80]
Reid, B. K. and J. H. Walker, *Scribe User's Manual, 3rd ed.,* Unilogic, Ltd., Pittsburgh, Pa., 1980.

[Rieger et al 81]
Rieger, C., R. Wood, and E. Allen, Large Human-Machine Information Spaces, *Proc. of IJCAI-81,* Vancouver, BC (also Univ. of Maryland CS TR-1024), pp. 985-991, Aug. 1981.

[Robertson et al 79]
Robertson, G., D. McCracken, and A. Newell, The ZOG Approach to Man-Machine Communication, Tech. Rep. CMU-CS-79-148, Dep. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, Pa., Oct. 1979.

[Seybold 78]
Seybold, P., Tymshare's Augment: Heralding a New Era, *Seybold Report on Word Processing* **1**, 9, Oct. 1978.

[Shackel 82]
Shackel, B., Plans and Initial Progress with BLEND - An Electronic Network Communication Experiment, *Int. J. Man-Machine Studies,* 17, pp. 225-233, 1982.

[Smoliar 72]
Smoliar, S.W., Comments on Moorer's Music and Computer Composition, *CACM* **15**, 11, pp. 1000-1001, Nov. 1972.

[Soergel 77]
Soergel, D., An Automated Encyclopedia - a Solution of the Information Problem?, *International Classification* **4**, 1,2, 1977.

[Stallman 81]
Stallman, R.M., EMACS: The Extensible, Customizable Self-Documenting Display Editor, *Proc. ACM SIGPLAN/SIGOA Conf. Text Manipulation,* Portland, Ore., pp. 147-156, June 1981.

[Trigg 83]
Trigg, R., A Network-Based Approach to Text Handling for the Online Scientific Community, Ph.D. Thesis, Dept. of Computer Science, Univ. of Maryland CS TR-1346, Nov. 1983.

[Uhlig et al 79]
Uhlig, R.P., D.J. Farber, and J.H. Bair, *The Office of the Future,* North-Holland, 1979.

[Walker 81]
Walker, J., The Document Editor: A Support Environment for Preparing Technical Documents, *Proc. ACM SIGPLAN/SIGOA Conf. Text Manipulation,* Portland, Ore., pp. 44-50, June 1981.

[Weinreb & Moon 81]
Weinreb, D. and D.A. Moon, Introduction to Using the Window System, MIT A.I. Laboratory, Working Paper 210 (pre-print of MIT Lisp Machine Manual Chapter), May 1981.

[Weiser et al 83]
Weiser, M., C. Torek, R. Trigg, and R. Wood, The Maryland Window Systems, University of Maryland CS TR-1271, January 1983.

[Weizenbaum 76]
Weizenbaum, Joseph, *Computer Power and Human Reason,* W.H. Freeman and Company, San Francisco, 1976.

[Wells 38]
Wells, H. G., *World Brain,* Doubleday, Doran & Co., Inc., Garden City, N. Y., 1938. (Reprinted in: Kochen, M. (ed.) *The Growth of Knowledge,* John Wiley & Sons, Inc., New York, 1967, pp. 11-22.)

[Wood 82]
Wood, R.J., Computer Aided Program Synthesis, University of Maryland CS TR-1219, Sept. 1982. (PhD. Thesis)

[XOC 83]
XOC, Inc., Xanadu(tm) Marketing and Technical Documentation. 1983.