SEARCHING IN A HYPERLIBRARY

Bruce R. Schatz
Michael A. Caplinger

Bell Communications Research
445 South St, Morristown, NJ 07960-1910 USA

Abstract
Proposed hypermedia systems rely mainly on the interlinked structure of their information to allow users to locate the information they are interested in. While such links are enormously powerful, link-following as a way to find specific information from a collection breaks down as the amount of information becomes large and eclectic, as would happen in a system intended to store all the literature of a community—a hyperlibrary. We propose a new facility in hypermedia systems to provide search via queries. The search mechanism is shown to complement linking, making creation of rich linked structures easier. Finally, we describe a prototype hypermedia system that provides search, and discuss our user experiences to date and some future directions.

Keywords: information retrieval, very large databases, hypertext, hypermedia, multimedia databases, information spaces, telesophy

1. Introduction

The original visionaries of what we now call hypertext systems were concerned with access to the universal library, the single source containing all the world’s knowledge. Vannevar Bush described interacting with a "memex" to find trails of information on a particular topic from many different sources [1]. Ted Nelson wrote about the "docuverse" containing large numbers of interconnected hypertext documents [2]. Douglas Engelbart’s NLS system pioneered the notion of an "information space" containing paragraph-sized units of information connected hierarchically [3]. The ultimate goal of these systems was provide access to all information, regardless of media type or physical location, in a uniform way.

The recent wave of interest in hypertext systems has displaced this goal with that of interacting with a single electronic book [4]. These systems concentrate on reading or writing a document with a rich structure of interconnections, where the interconnections might be used for purposes like linking related items or constructing an argument. In the literature on these systems, there is much discussion about browsing interfaces and link types, but almost none about how links come into being in the first place, or how information might be located in a very large collection.

In an attempt to remedy this lack, we have chosen to concentrate primarily on the mechanism of search, by which a hypertext system for single documents can be scaled up to a hypermedia library, or hyperlibrary, containing many interconnected documents.

2. Some terminology

Hypermedia systems usually build their structures out of atomic units, which we will call nodes, connected by links. Nodes might have internal structure (an entire paper might be stored as a single node, for example) but that structure is invisible to the hypermedia system as a whole.

We will call a connected subgraph of nodes a hyperdocument, or simply a document. Existing systems, such as Notecards and Intermedia, manipulate a single connected graph.

As a hypermedia system grows larger and larger, it is likely that its graph structure will not be connected; that is, there will isolated subgraphs not reachable from outside nodes. Such a graph structure could be called a hyperlibrary, with each connected subgraph a "hyperbook" in the hyperlibrary. The distinction between hyperdocuments and a hyperlibrary is shown in Figure 1.

3. Current hypermedia paradigms

Current and proposed hypermedia systems have enforced definite paradigms on document composition and viewing. There are six basic mechanisms for how nodes and the links among them are formed and subsequently examined. A description of many existing hypermedia systems and their features can be found in [5].
Hierarchies

Since hierarchical documents are common in ordinary literature, it should not be surprising that most systems support hierarchical arrangements of links well. An author might use such a system like today’s commercial “outline processors,” designing the basic structure of a document and then fleshing it out section by section (either depth-first or breadth-first). A reader could view a high-level outline of the document first, looking at interesting sections in greater and greater detail (depth-first) or scanning the entire document in progressively greater detail (breadth-first). The NLS/Augment system has primarily been used for such hierarchical documents (although it supports general links).

Nonlinear authoring

In nonlinear authoring, the writer makes little attempt to linearize the presentation of material. Instead, he or she simply writes about an initial topic, making diversions to other topics as they occur in the “stream of consciousness,” possibly returning to the original chain of thought later. Such non-linear writing might be especially valuable for composing the documentation for large, complex systems, or in more artistic endeavors (such as children’s “interactive novels”). There might be many possible reading orders, and whole portions of the network might be skipped entirely, as would happen if the author allowed readers to make decisions about alternatives at choice points in the narrative.

Pedagogical ordering

Pedagogical ordering is designed to allow a student unfamiliar with a topic to explore it in a less rigid way than a single linear presentation would allow. Typically nodes are constructed independently, and then links are added to emphasize particular relationships between nodes. For example, an INFLUENCE link might be added from a node describing Yeats’ work to one describing Eliot’s, because Eliot was influenced by Yeats. In general, a student would be expected to eventually traverse the entire network, looking at all nodes, but the exact order of traversal would vary depending on the varying interests of individual readers. The Brown University Intermedia system [4] is a good example of this usage.

Argumentation structure

In this approach, the author collects a large number of facts and deduces some conclusions from them, then adds typed links between the facts and the conclusions in order to compose an argument. Supporting such an argumentation structure is a major use of the Xerox Notecards hypermedia system [6]. A reader would typically be expected to begin at the first conclusion, possibly examining the supporting and contradicting evidence, and then move on to later conclusions in a single well-defined order; there is a single path through the document graph, with optional branching side paths.

Versions and annotations

In Nelson’s descriptions of the Xanadu system, most documents contain large portions from earlier work as quotes, or are revisions of earlier documents. Links are used to eliminate redundant material; a new document contains only new text, with links to the older incorporated material. When the document is read, the links are followed and the new document constructed in its full form dynamically. While less compositional links describing relationships are also supported, a majority of link following in the system is this automatic form.

Trails

Trails, first proposed by Bush, are left by a reader as he or she browses through an existing collection of nodes. Previously unnoticed connections are made between nodes, building up a series of associations that might represent an ordering like historical development or less

---

It is interesting to note that Bush’s memex had no ability for automatic search, but came equipped with a mechanism — the "code book" — to be used for classification. This was something like a library card catalog.
obvious interdependencies like those described in [7]. Later readers may wish to follow the trails, which may also contain annotations describing their significance, as added by the original author.

4. Problems with current paradigms

Current paradigms for the use of links in hypermedia systems almost uniformly make the assumption that the links are constructed by the author at about the same time as the nodes themselves, either between node creations as with nonlinear writing, or immediately afterward as in argumentation root node from which a reader can start to explore the document network. Both of these assumptions are viable in a relatively small network of a few hundred or perhaps a few thousand nodes. The author is able to remember some underlying structure for a network of that size (at least long enough to get links into place) and the topic of the document as a whole is likely to be narrow enough that a single starting node makes sense.

However, if we want to extend hypermedia notions to much larger networks and ultimately to all available literature, both assumptions break down. While individually-authored subgraphs (the "hyperbooks" of the hyperlibrary) are still small enough to be written and read as above, the global structure of interrelationships is not. When the library is first set up, each book exists in isolation and there are no links between any books. As a result, there is no way to find possibly relevant works by following links, since there are no across-book links and no root node at which to start. We propose that in order to manage a very large collection of distinct documents, a new mechanism for document location must be provided.

Figure 2: A typical telopephy screen
5. The searching paradigm

A typical hypermedia system provides link-following as the only mechanism for obtaining new units for viewing. We provide the additional mechanism of search. The search operation accepts a query from the user and locates all units that match that query, making them available for viewing. We call a set of units matched by a query a region, because it can be viewed as a subspace of the entire information space of units.

The definition of what a query is and the criteria for how queries match units is very general and need not be fixed for the entire collection of units. (Consider, for example, the features of a pattern recognition method for querying image units versus a speech recognition technique for searching voice recordings.) However, we acknowledge the fact that query matching is inherently inaccurate, in that what the user actually wanted to retrieve and what the system actually matched may not be the same. We relax the demands on the searching mechanism by allowing fast browsing on the matching units. The user performs a query, then rapidly looks at the results in the form of one-line unit descriptions. The complete contents of a unit can be seen by selecting its description; we call this zooming.

Some of the items will be obviously irrelevant even at the one-line level of detail; some may appear relevant until examined more closely, and some will actually be useful. The useless items can be deleted from the region. Examination of the useless items may suggest reasons the original query was imperfect, so that additional queries can be issued and the results added to the region. Once the user is satisfied with the region's contents, it can be annotated and stored back into the system, such that the next similar query will match the region directly, as well as other items. If a region so located proves useful to the next user, the effort of a second search can be reduced or eliminated.

Figure 2 contains a typical screen from our prototype hypermedia searching system. A new region has just been constructed by examining and selecting units matched by a query. The region's contents are shown in the window on the upper left; displayed just below is the classification information for the region. The results of the original query are shown in the lower left. In the course of filtering those items, the user has zoomed into a magazine article; the text of that article is shown in the lower right. This text article is actually one piece of a hyperdocument; another piece is the picture which is shown in the upper right.

This searching paradigm closely resembles a literature search in a physical library. A card catalog or index is consulted to determine possibly useful books; those books are located and examined; if they prove useful other books on the same topic (shelved nearby) may also be examined. Reading book titles and opening books that seem interesting is like scanning through one-line descriptions in a window and zooming into the interesting ones. Note that any links that are present between books are still valuable; once a useful unit has been located its links can be followed, much as a bibliography in a useful article may yield other articles.

Searching complements linking by making it possible to find items that should be linked before the links exist. The region resulting from a filtered query can itself be viewed as a collection of links to its contained items.

5.1. The relationship between search and links

In some sense, searching and linking are isomorphic. One could imagine implementing search by creating new nodes, one for each possible query, and then linking each query node to all the items it matched. Conversely, many uses of links could be as easily implemented by search; in a unit where each occurrence of a word was linked to a node containing that word's definition, a search for the word (with the restriction that only "definition-type" nodes should be matched) would serve as well as following a link. Some of the similarities between links and search were pointed out by Conklin:

We propose that in hypertext initiating a search is done for the same reason that one follows a link, i.e. to get from one node to another. The difference is that explicit hypertext links typically connect one node to another, whereas keyword search links a virtual set of nodes. Therefore, search of the hypertext graph amounts to definition of a network of implicit, virtual links, and the keywords can be regarded as a kind of implicit, computed link. [5, p38]

it should be noted, however, that search does not require locating some initial node first.

Links and search are not completely equivalent, though. Links can be split into two classes: those links that are implied by contents and can be found automatically or implemented by search, and those that are made explicitly and require some reasoning to uncover. This first class of links includes full-text matches, where all units whose text contains a specified word or phrase are implicitly connected. For example, all documents whose text mentions "cloud formation" can be located automatically by string search. Many uses of traditional links are simply a compact search. Many uses of traditional links are simply a compact search.
The second class of link is impossible to create automatically, and therefore much more meaningful and information-rich. It takes an act of thought to recognize the connections between, say, cloud formation and the structure of matter. Such connections must be manually specified with explicit links.

Of course, the line between the two classes of links depends on the state of the art in information understanding. Currently, only the purely syntactic level of extracting words and phrases actually appearing in a piece of text is feasible. More semantic scanning, such as the automatic assignment of a small set of keywords that accurately describe an item, is not yet possible on any large scale and must be manually performed by human reviewers.

By providing a search mechanism which makes constructing links of the first class easy, we hope to enhance users' abilities to find the more profound links of the second class. Filtering the results of a query in a region is easier than explicitly creating a relevant link between two unrelated units since it relies on automatic matching by the system to perform the initial part of the grouping process. Region creation, as a natural product of search, is not commonly found in hypertext systems and is useful in many circumstances as an analog to the production of annotated bibliographies of literature in a given field.

5.2. Time-varying information

Another area poorly addressed by current systems is the management of time-varying information or information added over time. Nearly all systems make the assumption that once a document has been constructed, it will remain fixed (except possibly for links added by readers) over its lifetime.

In a system that attempts to store all documents of interest, this is a bad assumption. For example, new items will constantly be added from news services, electronic conversations, and the natural output of authors at work. Such additions might be linked in the existing structure at many points. How is a reader to know that new, possibly interesting material has appeared?

One approach, based only on links, maintains for each user a set of links to items not yet examined. As the user reads, items mentioned in the list are taken off, and as items are added into the system, links to them are put on the end. Periodically, a user goes through the new list to see if anything interesting has appeared.

This approach has a large flaw, however; it fails to take the user's interests into account. Since every new item is linked in, the user is deluged with a mass of irrelevant items. If the user took the time to reexamine areas of the network that interested him, the new items would be located, but only at the expense of looking at a lot of material again.

The answer to this problem seems obvious; simply maintain a list of items the user has already seen, and suppress their display when looking for new items. However, this list may become unreasonably large if many items are examined. We solve this problem with search by allowing a query to be made based on the creation time of units. For example, an "electronic newspaper" can be obtained by issuing a query of the form "creation-time = today and <user's interest profile>". If the system keeps track of when this query is issued, it can be modified to return all items that were added since the last time it was issued (with a clause of the form "creation-time > last-time-of-query").

6. Telesophy Systems

Our interest in search inside a hypermedia graph grew out of an attempt to take a fresh look at the original visionaries' goal of accessing the universal library. We wanted to build a system which could eventually store and manage a single information space containing all the world's literature. Such a system could be called a telesophy system. "Telesophy" is a word coined by one of us (Schatz). Literally meaning "wisdom at a distance," it is a fusion of the words telephony and philosophy. The intent of a telesophy system is to provide the same level of transparency for manipulation of knowledge as a telephony system provides for transmission of sound. As the original feasibility study [8] argues, it is possible to build a telesophy system by combining technology from several existing areas into a single new kind of system. See [9] for an overview of telesophy systems.

A telesophy system provides uniform access to large amounts of distributed, multimedia information. The user interacts with a local terminal connected via a fast network to remote information sources. A hypermedia system runs in the local terminal to handle the manipulation, such as display, editing, and link construction. A hierarchical network routes queries to the appropriate sources and routes results back to the requesting terminals. The remote machines use information retrieval technology to provide fast search for large amounts of data.

In a telesophy system, the user sees a single large collection of nodes called information units or IUs. Each IU has three components: contents, the structure of the unit itself; connections, links (possibly with annotations and types) to other IUs; and classifications, additional information (such as keywords, assignments to an indexing scheme like the Dewey
manipulating "knowledge" while the system reflects this by eventually manipulating "data".

Although the user sees just a flat space of IUs, the system is actually distributed, with IUs being stored on many different IU servers connected via a fast network. Each IU in the system is assigned a unique identifier that can be used to fetch or modify it; these identifiers are the basis of links in the system.

As described above, the user can issue a query in order to select all IUs that match it. Queries are transmitted to indexing servers distinct from the IU servers. Because query processing is separated from IU storage, many different indexing schemes can be employed, depending on the needs of a particular IU's contents. For example, text IUs will appear in indexes providing retrieval by string match, word proximity, and Boolean combination; a picture IU might appear in a (hypothetical) pattern-matching index. Classifications are always textual attribute-value pairs, so even without sophisticated visual indexing, it is possible to search for a picture with a particular title or subject (assuming appropriate classification).

There are potentially many indexing servers of each type available; an IU may be indexed in more than one, and no index must store all IUs. This allows partitioning and load balancing as the system grows.

6.1. Current prototype

We have constructed a working prototype of a telesophy system, called TSI. It uses a notion of "distributed objects" coupled with indexing servers providing full-text information retrieval. The architecture of the prototype is described in more detail in [10].

The current prototype is accessible from hundreds of computers, mostly Sun Microsystems workstations, connected by a large virtual network extending across several Bellcore locations in New Jersey. Logical access is transparent, unaffected by the physical location of the user's workstation. Within the Bellcore Morristown building LAN, speed of access is transparent as well, independent of the physical location of the user's workstation. Additional trials have been run over a simulated 10 mile MAN and an actual 50 mile WAN (wide area network comprised of 2 building LANs connected by a T1 link) with the speed of access being roughly the same as the intrabuilding case. Presently, there are about twenty IU server processes storing some 300,000 items, and about ten indexing servers for query processing, spread across three large-configuration file server machines.
The information stored represents our attempts to obtain as large a cross section of useful information for our community as we could. It is in fact a representative sample of what is currently available electronically. The majority of the items are journal citations in computer science and electrical engineering, from the INSPECT™ database of the IEE. These are not full-text items, but contain one- or two-paragraph abstracts and classification information from articles in a wide selection of technical journals. INSPEC serves the needs of researchers doing bibliographic searches in their fields.

Information of more general interest includes about a year’s worth of full text from a range of monthly magazines from Scientific American to The Whole Earth Review, as well as a year’s worth of the text from the weekly’s Business Week and Electronics. These represent some 5000 articles.

Internal Belcore information is also available. These include the library catalog and the technical memoranda (citations with abstracts; approximately 5000).

Daily, stories from the Associated Press and New York Times wire services are added, as are incoming messages from Usenet and ARPAnet newsgroups. There are also a number of special-purpose databases like the Magill’s Survey of Cinema, which contains plot synopses or critical essays for some 8000 motion pictures.

We have also added a modest collection of digitized photographs and videodisc frames and clips. (It has been considerably harder to obtain a large collection of such non-textual items because of their scarcity in the commercial market. We have not yet found a suitable pictorial archive with an accompanying classification database to index upon.)

Finally, the system stores the personal notes, regions, and links created by its users.

6.2. Implementing search

Since the vast majority of the items stored in TSI are textual, we have concentrated on fast textual search. A discussion of the searching implementation can be found in [11]. The system uses several indexing servers, each running the same indexing software. IUs are assigned to a particular server based on source; for example, all IUs generated from INSPEC abstracts are indexed in the same server. The indexing software provides full-text word matching on the contents and classification of each IU, as well as boolean combination (and, or, and not) and word proximity. The latter function can locate phrases such as “information retrieval” efficiently by performing the query “information before retrieval”. The system also allows word truncation, so it is possible to search for “comput*” and match occurrences of “computer” and “computation”. Stemming is also performed, so that singular queries such as “computer” will match plural occurrences (“computers”).

Since classifications are text attribute-value pairs, words can be restricted to be part of particular attributes (as in the query “title contains laser”). This trivially provides the features of “semi-structured messaging” [12].

The word indexing system uses well-established techniques from information retrieval, such as those described in [13]. Each index maintains its total list of words in a binary tree data structure for rapid search. Associated with each word’s entry is a list of which IUs contain that word; additional information about where the words occur is stored for use by the proximity operators. An index is stored in multiple levels so that all of it need not be resident in memory to perform a query; this allows the indices to become extremely large. In the current system, indices of up to a few hundred megabytes (few hundred thousand units) can be searched in under 2 seconds.

6.3. Finding appropriate indices

We mentioned that there were multiple indexing servers, partitioned by source and for load. This means that when a user issues a query, the front end must decide which of the available servers should process it. Since the indices are partitioned by source, a user can also request that the query be sent to only a subset of the possible servers; for example, a query intended to cover only popular literature can be sent explicitly to the “magazines” index. With the amount of information and number of indices in the current system, it is feasible to simply send the query to all servers, and this is typically done. This solves the index choice problem common to many commercial searching systems; if a user is unsure of what indices might be correct, the system simply searches all of them. Of course, as the size of the system grows, this will become less and less possible. We discuss this and other scaling issues in the following section.

7. Scaling up

We expect our system to handle a few million items comfortably. In comparison, ten years of INSPEC contains about 1 million items; today’s Dialog™ Information Service has closer to 100 million. The Library of Congress also contains some 100 million items. However, many of these are books, which unlike the citations in Dialog are themselves composed of many interconnected items. So a large library could easily contain a billion information units.
There are two areas where increasing the number of items by two or three orders of magnitude impacts searching: the efficiency of the search algorithms, and the filtering process on a query result. The first is well handled by the already distributed architecture of the indexing servers; as more items are added, more and more server machines can be added to index them, using parallel processing to keep the time to perform a given query constant. (Since the search time per index only increases logarithmically with its size, new machines need only be added infrequently.)

Filtering is more seriously affected. Suppose that queries are sufficiently imprecise that they match a constant percentage of the total number of items, say 0.1%; so that in a space of a million items, a query always matches a thousand. These are typical numbers for coarse queries in the commercial bibliographic information services of today. If we increase the number of items to a billion, then the same query matches a million items. Scanning through a thousand items' one-line descriptions is possible via page flipping on a workstation, but scanning a million items is not.

Two solutions are possible. One could use graphical techniques to scan items at an even less detailed level than one-line text descriptions. Such a technique is analogous to browsing through the library shelves looking for books based on color, size, or position rather than looking at the titles. It involves the assignment of visual and positional attributes to items based on classification information, then displaying the results graphically. For example, we might represent books as points along a Dewey Decimal System axis, immediately recognizing that items about neurobiology, matched by a query for "fiber" when "fiber optics" was intended, are inappropriate. If placement along multiple axes could be assigned, higher-dimensional displays would be possible. The problems with this scheme are largely those of determining useful axes and then positioning individual items along them. The topic is more fully described in [14].

7.1. Metaqueries

The alternate solution is to make queries more specific, such that unscannable numbers of items are never produced. At the simplest level, this might be done by restricting the query to a subset of indices most likely to contain relevant material. This assumes the flavor of a "metaquery", where the indices themselves are classified and the user initially identifies what the general subject area of a search is to be; that general area is used to search the classified indices to choose appropriate ones.

Suppose that efficient search is possible when an index handles 100,000 items. This is a reasonable assumption for current word indexing technology. Then suppose that there is a space of one billion items with the items uniformly and independently distributed across the indices. This situation requires 10,000 indices. If an initial metaquery is efficient enough to cut the number of possible indices to 0.1% as before, to ten, then we are reduced to the previous case of searching a million items from those indices. The metaquery thus restricts the subspace searched sufficiently to result in returning one thousand items, which is easily scannable. It might even be possible to make metaqueries more efficient and restrictive than queries — because there are many fewer indices than items, more work could be devoted to classifying them.

Producing a metaquery in the first place could be done in several ways. One, obviously, is to have the user specify it, perhaps at the beginning of a research session in a particular area. Once suitable indices had been located, subsequent queries would be sent only to them. The disadvantage with this approach is that it requires the user to know something about the structure of indices, and for some kinds of research, a single fixed set of indices might not be appropriate. (One sees this problem with database selection on Dialog already.)

Another technique would automatically produce a metaquery from each query. Such a metaquery processor might resemble an expert system, using knowledge about a user's interests and past behavior, along with the content of a query, to choose appropriate indices. Because of the relatively small number of indices, current AI technology might be capable of metaquery processing. An intermediate technique is to have the user first specify a broad subject area, invoking a specialized command script which automatically restricts the queries to certain indices and certain words. Such scripts serve as "search strategies", and could be manually constructed by subject matter experts.

Instead of reducing the number of indices to be searched, we could attempt to make queries more discriminating. Simple word matching cannot cope with multiple word senses, context sensitivity, or other ambiguities — capturing more of the semantics present in the text to be matched would be highly desirable. A complete solution of this nature would require total natural language understanding, but an approximate solution might be possible using more mechanical methods of sense disambiguation. For example, in the "fiber" query mentioned before, the system would detect the word's ambiguity and ask the user what sense was intended; items would be indexed by word and sense rather than just by word.
A technique for sense disambiguation is presented in [15].

We are also investigating ways to automatically classify the subject matter of text, using word matching against a classified lexicon. The approach is similar to that described in [16], but uses phrases rather than single words.

8. Granularity

Our terminology has highlighted a problem with the granularity of information units. At one level, every IU is a node in a single graph structure, and search just selects matching nodes. Hyperbooks are just subgraphs of the total space. However, regions themselves can be stored back into the system as new IUs, and since IUs can be defined as arbitrary objects, they can have an arbitrarily complex internal structure of links.

This raises a question about the behavior of search. If an atomic IU (a piece of text, say) is matched by a query, should the system retrieve just that IU or any region that contains it? Put another way, do regions inherit the searching properties of their components?

In the prototype system, most IUs tend to be article-sized pieces of text — linear collections of paragraphs, too small to have an internal linking structure. We have not yet attempted to build large interlinked subgraphs, where individual elements do not stand alone. Therefore, our treatment of regions so far has been to match them only using the keywords assigned directly to them; they do not inherit the keywords of their children. However, this means that for a given matching IU, it may not be possible to find all the regions that contain it, since links are typically unidirectional. While this has not been a problem so far, we anticipate further investigation of granularity issues, especially when large collections of interlinked IUs are added.

9. Previous work

Obviously, search of this kind is nothing new, since commercial information retrieval systems have been providing similar functionality for years. However, the application of searching to hypertext systems is surprisingly absent. While a number of systems (Notecards, TextNet, ZOG) have provided search, none seem to have emphasized it as a major system feature. Searching was usually an afterthought, implemented by linear search, and painfully slow even with the very small numbers of nodes such systems manipulated in comparison with TSI.

One hypermedia-related work that does emphasize search was Weyer's thesis on dynamic books [17]. He built a system that provided browsing and search functions for a single history book stored in electronic form, then tested this system versus a conventional paper book on a number of eighth-grade students. It is worth noting that search in this system was by the term hierarchy provided by the book's table of contents alone; no full-text searching through the entire book was possible. In some cases, it is clear that performance on the question-answering tests used would have been better if a less restrictive search technique had been provided.

10. Conclusions

The telesophy prototype has been running for over two years. Within Bellcore, there are some 15 registered users in the community and perhaps 50 occasional users. The software is also available as a technology transfer package (free to non-commercial organizations). There are currently some 20 external sites; other interested parties are encouraged to write to the first author.

One good example of the prototype's usage was at the month-long Matrix of Biological Knowledge Workshop in summer 1987 [18]. There, some 10 professional biologists used it with an information space of biomedical literature abstracts to generate regions corresponding to biological concepts. For example, one region contained abstracts relevant to "hormonal regulation in sea urchin" while another contained those relevant to "functions of peptide bombesin". Typically, several searches were required to discover all the appropriate phrases for the topic under consideration. A common technique was to page flip through the scanlines resulting from a query, zoom into a few abstracts, save away some of these in the new region, and use others to extract new phrases to search for (there is a system facility for extracting text from an abstract Contents and feeding it into a query). Compared to their existing tools (traditional information retrieval over telephone lines), the telesophy prototype received enthusiastic marks for its browsing speed, intra-database transparency, and sharing composition.

Our experience has indicated that search is an essential feature of a hypermedia system with a large, diverse set of documents. Regions containing annotated collections of filtered query results have been the primary way new link-like information is added into the system. We have found that the use of general node-to-node links is relatively infrequent (as have architects of more conventional hypertext systems [19]). We believe that for the purposes our system is used (general literature browsing, scanning new information periodically, personal information management), locating items via query and forming regions are more natural paradigms than simple
link-following and explicit link construction. (Of course, it must be noted than our initial document collection has only a small number of links, generated automatically. In a single hyperdocument with a carefully-constructed link structure, links would be far more important.)

Put another way, we argue that the authoring process is composed of two activities: research and composition. When doing research, fast location of relevant material is required, and the system should support this directly; regions are a natural way of capturing research results for latter use by the author and others. After a collection of useful material has suggested the directions that the new work is to take, priorities change. Convenient ways of establishing and following links are likely to be more important while actually writing a new hyperdocument.

We expect that in some applications, very little new material will be added by a user. Annotated collections of old material may be adequate and quite common; consider a bibliography in a specified field, or a digest of bulletin board messages.

In the next year we hope to increase the number of IUs by another order of magnitude and to include some entire hyperdocuments with local link structure (such as an encyclopedia or system documentation). This should help us to better understand the interaction between search and links, and the system facilities required to support hyperbooks within a hyperlibrary.

11. Acknowledgements

The indexing and searching system used is TSI 1 was designed by Steve Bulick and implemented by Steve Miller. We also thank Bob Amster for insight into the field of natural language understanding.

References

1. BUSH, V. “As We May Think,” Atlantic Monthly 176, 1, pp. 101-108 (July 1945).
2. NELSON, T. Literary Machines, The Distributors, South Bend, IN (1983).
19. HALASZ, F. personal communication