Chapter 14

Hypertext Perspectives: Using Pathfinder to Build Hypertext Systems

James E. McDonald, Kenneth R. Paap, and Deborah R. McDonald

Hypertext is a "hot" topic. Interest in hypertext systems has burst upon the computing scene—which is a bit surprising given that the basic concepts have been around for more than forty years (Bush, 1945). Why this sudden interest in "nonlinear text?" In this chapter we examine some of the claims being made for hypertext and attempt to identify those that show promise of producing real gains in peoples' ability to acquire information, as distinct from purely technological embellishments. As with most technological advances, there seem to be costs associated with the benefits of hypertext and we will discuss how knowledge elicitation and representation techniques might be used to minimize the impact of some of these costs. In particular, we will show how Pathfinder network scaling can be used to construct hypertext systems in order to facilitate information access.

Hypertext

Vannevar Bush (1945), the father of hypertext, began his seminal work by identifying a problem that has become ever more acute: There is simply too much information for anyone to assimilate, and it's not readily available. Furthermore, even if all the information in the world were available, it would not all be relevant to any given task. Although technology has certainly facilitated the accumulation of knowledge, it has done little to make it more accessible. The trick, as Bush saw it, is to devise a system that allows users to access relevant information without forcing them to wade through a lot of irrelevant facts.

Bush proposed the *memex* as a way of mechanizing the scientific literature, placing it within reach of individual researchers. He envisioned this contraption as containing a very large library, along with personal notes, photographs, sketches, and so forth. Although not technically feasible in 1945 (many would question whether it is feasible even today), the design of the *memex* was inspired by Bush's belief in how the mind works:

> The human mind...operates by association. Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to be able to learn from it. One cannot hope to equal the speed and flexibility with which the mind follows an associative trail, but it should be possible to beat the mind decisively in regard to the permanence and clarity of the items resurrected from storage.

Thus, researchers would use the *memex* to build *trails* by identifying links between items of interest. Trails could later be followed by the builder, or other users, to recreate episodes. Over time, networks of trails would organize knowledge in an efficient, useful way. The *memex* was intended to complement human abilities, not replace them. Bush

was motivated by a perceived problem and formulated a solution based on his understanding of human strengths and weaknesses. Although a technological solution, the *memex*; was not inspired by technology.[1]

What is Hypertext?

One of the first tasks in identifying and evaluating the claims being made for hypertext should be to figure out exactly what it is. Unfortunately, perhaps because of the current hyperbole surrounding the topic, hypertext doesn't lend itself to such analysis. Although hypertext systems do seem to share a number of common characteristics, such as the ability to manage nonhierarchically organized information and to allow interactive branching among information "nodes," there doesn't seem to be any necessary or defining features. For example, the use of "windows" containing chunks of information is often viewed as important, as is the ability to provide users with a single, coherent interface. However, these are neither necessary nor sufficient criteria for classifying systems as hypertext.

In his excellent hypertext review article, Conklin (1987) identifies machine-supported links as an essential feature of hypertext. Without additional qualification, however, this requirement is not specific enough since a definition based on machine-supported links would encompass virtually all existing database management systems. To truly qualify as hypertext, Conklin contends that a system must support direct references from one information chunk to another and allow users to easily establish new relationships between chunks. In other words, it must be possible to link any information node to any other node without restrictions (e.g., the structure need not be hierarchical). Furthermore, if a user wishes to establish new links, a good hypertext system should have features that support their construction. Alternatively, if the user is simply engaged in information retrieval by browsing along a pathway of nodes, a true hypertext system should have features that support the selection of relevant nodes and minimize the likelihood of getting lost. Therefore, although windowing systems, outline processors (e.g., ThinkTank), text formatters (e.g., Troff, Scribe), database management systems, and even file systems each possess some features of hypertext, they all fail to qualify for various reasons.

A walk in the woods. In order to make the rather fuzzy concept of hypertext a bit more concrete, imagine a wilderness in which there are numerous crisscrossing and overlapping trails. Within this network of trails there are also many starting points and destinations for journeys through the wilderness, as well as numerous points of interest along the way. Travelers typically follow established trails, but may choose to blaze new ones depending on their knowledge and daring.

This analogy depicts the opportunities for exploration and discovery afforded by hypertext systems. It also suggests some of the potential problems, such as disorientation and confusion, are consequences of wandering around in hypertext without a good map and accurate trail signs. However, it fails to capture some important aspects of the information wilderness faced by users of hypertext systems as well. First, hypertext eliminates most of the limitations inherent in physical space. Specifically, hypertext removes the restrictions of traditional information systems consisting of printed books and journals, since information nodes are all potentially equidistant. Second, the ability to get from point A to point B efficiently is less important in hypertext than what's learned along the way. The content of information nodes and the order in which they are visited are what counts, not necessarily how quickly one arrives at the destination. Finally, hypertext systems often do a poor job of signaling established trails. In fact, the kinds of trails that Bush envisioned and that

[1]This is in contrast to the recent interest in hypertext, which often seems to be motivated more by technological feasibility than any perceived need.

exist in our hypothetical wilderness aren't typically represented in hypertext systems at all. Instead, users are presented with sign posts indicating nearby points of interest. Rather than trail signs such as "Rim Trail" or even "Trail #117," the user is more likely to encounter references such as "Dead Man's Gulch" or "Sourdough Hill." The point is that little indication of the relationship between trail segments and well-established trails is provided. A potential solution to this problem is to provide orienting aids, and the hypertext browser described later uses maps based on various perspectives for this purpose. Extending our walk in the woods analogy, one perspective may show trails relevant to botanical study while another map shows trails leading to the best fishing spots. Although segments of the two maps may overlap, confusion is reduced by selecting and viewing the map that is appropriate to the current intention or goal.

In spite of the difficulty of precisely defining hypertext, the excitement over this new technology continues, primarily because of the widespread belief that a linear organization of information is inadequate for many applications. Since information embedded in computers can be represented and used in qualitatively different ways than on paper, it seems reasonable to believe that these new representations will aid in the process of knowledge acquisition in unforeseen ways (Fischer, 1987). For many of us, however, the nagging question remains, "Why hypertext ?"

Text Comprehension

Reading and Writing

Although a number of hypertext systems are now in use and interest in hypertext is intensifying, the psychological ramifications of hypertext as a tool for acquiring knowledge have not been empirically validated—even worse, we claim that the important psychological variables have yet to be identified. In order to make the topic more manageable, we will explicitly distinguish between features of hypertext systems that are important for reading (or trail following) and those that are more concerned with writing (or trail blazing).[2] We will begin with a discussion of reading and go on to suggest a methodology for building networks of trails appropriate for hypertext systems. One area of psychological research that is clearly relevant to acquiring information from hypertext systems is text comprehension.

According to the van Dijk and Kintsch (1983) model of text comprehension, there are three levels of representation: surface memory, text-based memory, and the situation model. The *surface memory* representation for a text is the verbatim encoding of words and phrases in the text. The surface memory provides the basis for the *text-based memory* representation. Text-based memory is for the propositional content and structure (micro- and macrostructure) of the text. The *situation model* not only involves memory for the meaning of the text and the unique situation described, but also incorporates the reader's goals and prior knowledge. Situation models are sometimes referred to as *conceptual* or *mental models*, and it is this level that is most relevant to our discussion of hypertext.

Considerable evidence has accumulated for the validity of distinguishing between textbases and situation models (Dellarosa, 1983; Fletcher, 1984; Perrig & Kintsch, 1985; Schmalhofer & Glavanov, 1986; Weaver & Kintsch, 1985). The consensus of these

[2]It should be noted that Bush (1945) did not distinguish between reading and writing and, in fact, felt that much of the power of the *memex* would come from the ability to combine new knowledge with existing information. Current authors often blur this distinction as well.

researchers is that people do not always remember the surface form or even the syntactic deep structure of a text when they remember what the text was about. Furthermore, the experiments in this area show that people's representations of a written text go beyond information directly presented in the text. For example, many pragmatic inferences are drawn as the propositional content of the text interacts with the reader's world knowledge. Similarly, reading nonlinear text might influence the development of conceptual models, although to our knowledge no empirical studies have examined such effects of hypertext.

Unlike traditional text, hypertext represents multiple relationships among various "chunks" of text simultaneously. Traditional linear text encourages the reader to follow a single sequence of paragraphs (although readers may choose to jump around), whereas hypertext provides alternative organizations for the same information, encouraging readers to wander. In terms of our walk in the woods analogy, this would be comparable to restricting one traveler to a given trail while allowing another to travel any route he chooses. The mental representations that these two travelers would form of the wilderness are likely to be quite different, even for a particular trail they both know. The second traveler, free to explore the wilderness, selecting points of interest to visit at will, is likely to posses a more elaborate mental model and, if the trails are carefully laid out, a more accurate model as well.

The Benefits of Multiple Perspectives

What benefits might we expect from providing readers with nonhierarchical, richly interconnected networks of information as opposed to linearly organized text? hypertext proponents seem to assume that the benefits are obvious, although the only benefit that is obvious to us is that having access to necessary information is better than not having access to it, a tautology. At the same time, these proponents acknowledge the problems of confusion and disorientation associated with hypertext systems (Charney, 1987; Conklin, 1987; van Dam, 1988), which makes even the promise of increased information availability suspect. Postponing a discussion of disorientation, there has been some recent interest in the effects of providing readers with more than one perspective on a given topic. In particular, researchers have been interested in the ability of readers to remember details and draw inferences, based on the way in which information is presented.

Mannes and Kintsch (1987) were interested in the notion that learning from text is different from memory for text. Using outlines as advance organizers, they showed that when subjects studied advance organizers whose structure differed from the succeeding text they created more richly elaborated mental representations of the textual materials. In a related study, McDonald (1987) found that requiring readers to form a more complex conceptual model by having them read two different versions of the same text enhanced their learning of the content, compared to readers who read the same version of the text twice.

In her study, McDonald devised two factual texts that both described the violin. These texts contained approximately the same information, but one text described the violin from the perspective of the historical development of the instrument (the *historical* version), whereas the other described the same aspects of the violin from a structural/functional perspective (the *categorical* version). Subjects were either in the Inconsistent condition and read both the *historical* and *categorical* texts, or they were in the Consistent condition and read the same text twice (either *historical* or *categorical*).

After reading the texts, subjects were given a true/false test containing both memory items (either verbatim excerpts or statements that had been paraphrased from the texts) and inference items (conclusions of either formal inductive or deductive arguments), as well as statements that were extraneous to the texts (foils). For inference items, the premises were presented in the body of each of the texts, but the conclusions were not. The subjects' task

was to decide whether each statement was true or false according to the information they had read in the texts. The important results for the present discussion were that, as expected, subjects in the Consistent condition were better able to recognize verbatim and paraphrased items than subjects in the Inconsistent condition. However, subjects in the Inconsistent condition were better able to draw correct inferences than subjects in the Consistent condition.

The results of this study are intriguing. Earlier we suggested that separate maps (perspectives) should facilitate following the trail relevant to the current goal. But suppose our adventurer pursues flora on one outing and fish on the next. If McDonald's results transfer to the wilderness, we would expect the eclectic outdoorsman to have made more correct inferences about the local terrain than the dedicated botanist (or fisherperson) who travels the same trail every time.

There does appear to be some support for the notion that providing readers with several perspectives on the same topic is beneficial, at least for drawing inferences, implying that their conceptual models of the topic are more complete. However, rather than several discrete perspectives (or well-marked trails), hypertext systems typically provide users with tangled, poorly marked perspectives. Although Eileen Kintsch (1988) has speculated that forcing readers to work hard at uncovering structure may produce even more elaborate mental representations, neither the Mannes and Kintsch (1987) study nor the McDonald (1987) study investigated the consequences of flipping back and forth from one perspective to another. Subjects in the Inconsistent condition first read one coherent passage (or outline) and then read a different coherent passage. Unrestricted access and frequent switchbacks between perspectives may be harmful rather than beneficial. This is an empirical issue and one that McDonald is currently pursuing. We will now turn our attention to how hypertext networks might be built and represented in order to facilitate navigation and the development of useful conceptual models.

Empirically Derived Conceptual Models

A major theme of our work in the area of interface design has been that effective human-computer interaction depends on communicating system structure and organization to users via the interface (McDonald, Dearholt, Paap, & Schvaneveldt, 1986; McDonald & Schvaneveldt, 1988). The premise upon which this work is based is that users develop conceptual models of systems and that, in general, the more accurately these conceptual models correspond to the organization of the system the better. As a consequence, this line of research has focused on developing and evaluating methods for eliciting, representing, and communicating appropriate models of systems.

We believe that users' conceptual models can be usefully characterized as "schemata" in which parts of the system are associated because they are functionally related or because they have common features (Young, 1983). One of our objectives has been to develop an empirical methodology for eliciting and representing such conceptual models. The general methodology consists of (1) obtaining estimates of relatedness for all pairs of relevant system concepts (e.g., objects and actions), (2) analyzing the obtained proximity data using one of several scaling techniques, and (3) mapping the resulting representations onto interface organization (e.g., menu layout). A number of scaling techniques have been employed, such as hierarchical clustering (McDonald, Stone, Liebelt, & Karat, 1982; Tullis, 1985), multidimensional scaling (McDonald, Dayton, & McDonald, 1988), and simulated annealing (McDonald, Molander, & Noel, 1988). In this chapter, however, we will focus

on the use of Pathfinder network scaling because we believe that it is most appropriate for hypertext systems.

Using Pathfinder to Blaze Trails

As we said, it is common for users to become disoriented when navigating through systems that require them to search for desired information by traversing machine-supported links. This may be due to the nature of the "trail signs," as suggested, or because of a more fundamental problem associated with the complexity of such systems. The type of disorientation common to hypertext systems is also found in many "deep" menu-based systems (cf., Paap & Roske-Hofstrand, 1988, for a discussion of the breadth-depth tradeoff in menu design).

Roske-Hofstrand and Paap (1986a) compared several different prototypes for a Control-Display Unit (CDU), the primary interface for the complex flight management system being developed for NASA's advanced concepts simulator (Chappell & Sexton, 1985). The CDU is a fairly large menu-driven system that can be used to both acquire knowledge and take actions. In the prototypes evaluated, the pilot is shown a display which contains some text, sometimes a prompt for information that is to be entered, and a menu of options. The pilot selects options from the menu in order to display new panels of information. Pilots with limited CDU experience often have difficulty finding their way from where they are (the current panel) to where they want to go (the target panel). The goal of this study was to compare menu organizations based on Pathfinder networks to an organization based on the specifications of the original design team.

A subset of 34 panels from the original specifications for the CDU were selected. The information contained in each panel of the CDU was treated as a single, coherent concept. In the initial phase, four experienced pilots were familiarized with the panels, then rated the similarity between each of them. Various Pathfinder analyses were performed on the resulting proximity matrix and the Pathfinder networks (PFNETs) were mapped onto menu organization.

Sixteen experienced pilots participated in the validation test. The test was a training session that consisted of four blocks of 34 trials. On each trial the pilot read a scenario that described a set of current conditions, a general goal, and a specific question that could only be answered by accessing the appropriate CDU panel. The total task time for each trial was measured from the onset of the question to the pilot's response. Each of the 34 menu panels contained the target information for one trial in each block.

The results of this study support the use of PFNETs as a basis for interface design: Prototypes based on cognitive networks were consistently easier to learn than the prototype based on the original design specifications. In particular, this study shows how effective networks of hypertext trails might be constructed. Nevertheless, the CDU is relatively simple compared to most hypertext systems, and complex systems may require accurate maps as well as good route planning. A methodology based on user knowledge might be particularly appropriate for complex systems that are the products of evolution, rather than any systematic design process. In such cases, the "blueprints" which could be used to describe the systems to users simply don't exist. A classic example of this process of evolutionary growth, rather than controlled design, is the UNIX operating system and its associated documentation.

Using PFNETs to Build a Hypertext Browser

We have constructed a prototype of an interactive, graphic-based, hypertext Browser (HyBrow) for the UNIX online documentation system (the *man* system) using the general methodology outlined above. HyBrow enhances the user's ability to access documents,

particularly in a nonlinear fashion, by providing machine-supported links among *man* entries. Our objectives for HyBrow are to facilitate the ability of UNIX users to locate appropriate information and to encourage the development of useful conceptual models of UNIX (commands and utilities). The design of the HyBrow system was based on data obtained from experienced UNIX users and information extracted from the UNIX documentation itself.

In line with the methodology described, we employed empirical techniques to elicit and represent the knowledge of experienced UNIX users. We began the HyBrow development project by obtaining estimates of relatedness among UNIX concepts using two different techniques. Estimates of functional relatedness were obtained by having experts sort commands into related categories, whereas estimates of procedural relatedness were derived from co-occurrence data, that is, the probability that particular commands will be used together (i.e., temporal proximity).

The Functional Perspective: Elicitation of Command Knowledge

The functional networks used in the HyBrow prototype were constructed from data obtained from 15 experienced UNIX users. Seven were classified as expert or intermediate-to-expert users, whereas eight were classified as intermediate users. The knowledge elicitation procedure used during this phase of the project was card sorting. An index card was prepared for each of the 219 UNIX *man* Section 1 functions (Berkeley UNIX version 4.2 running on SUN minicomputers). Subjects were instructed to sort commands according to functional relatedness. In an attempt to reduce the "hierarchical filtering" associated with sorting, subjects were encouraged to use duplicate cards when they felt a particular command belonged in more than one pile. They were also instructed not to sort commands they didn't know.[3]

Deriving the Distance Matrix. A proximity matrix was created for the subset of 152 UNIX commands that were known by at least half of the 15 experienced users. The matrix was constructed by subtracting the average conditional probabilities associated with each pair of commands from one. It should be noted that computing proximity in this manner tends to increase the relative distances between pairs in which both items are not sorted by all of the raters (see McDonald, Plate, & Schvaneveldt, Chapter 11, this volume, for a more complete discussion of relatedness functions).

Scaling: Hierarchical Clustering & Pathfinder Networks. A hierarchical cluster analysis of the 152 commands was performed using the minimum, or single-link, method (Johnson, 1967). The minimum method was selected because of its relationship to Pathfinder with $r = \infty$ and $q = n-1$, which facilitates mapping the resulting subnets onto clusters. The minimum method also tends to produce fewer distinct clusters, reducing the amount of effort required during the category labeling phase. The hierarchical cluster analysis produced a total of 83 distinct clusters. The Pathfinder analysis resulted in a network with 184 links.

Establishing the Category Structure. The design of HyBrow requires us to graphically represent "subnets" (connected portions of the Pathfinder network) that correspond to high-level concepts. Figure 1 shows one approach to representing subnets that involves the combined use of hierarchical clustering, MDS, and Pathfinder. First, clusters were obtained from the hierarchical cluster analysis. Then the appropriate subsets of distance estimates for each cluster were subjected to a two-dimensional, nonmetric MDS.

[3]Early experience with the initial set of 219 commands suggested that we had been too restrictive. Therefore, the command set was expanded from 219 to 373 by including new commands that were commonly used (see the Procedural Perspective).

Finally, the MDS coordinates were used to guide node (command) layout, and the links specified by the Pathfinder solution were added. Comparing the hierarchical and network representations of these clusters is informative. In Figure 1, for example, not only are the two clusters that consist of directory-level and file-level commands evident in the PFNET, but the "bridge" between these two clusters, from *rmdir* (remove directory) to *rm* (remove file), can be seen as well. It is tempting to speculate that this connection represents more than the simple fact that both of these commands remove things. Most UNIX users will recognize that these commands have another fundamental relationship as well; in order to remove a directory, all of the files in the directory must be removed first.

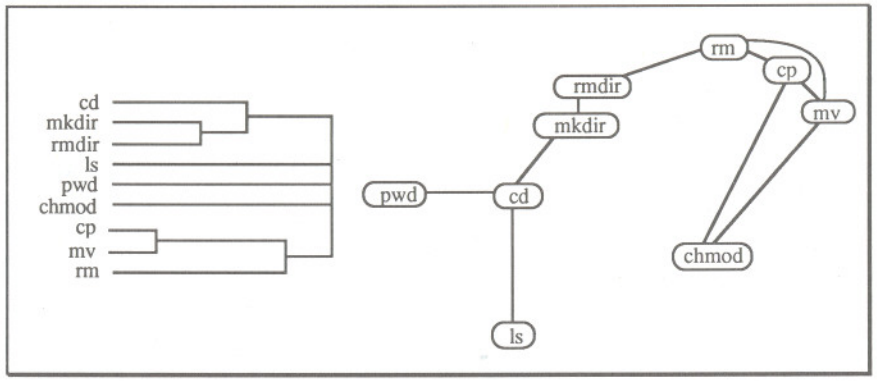

Figure 1. A comparison of hierarchical clustering (Minimum Method) and network analysis [PFNET($r = \infty$, $q = n-1$)] for UNIX directory commands.

A similar comparison of hierarchical cluster analysis and Pathfinder networks is shown in Figure 2. In this example, the subset of commands used in programming (e.g., compilers, interpreters, etc.) was selected. Rather than relying on an MDS analysis to help lay out the network, only the constraints inherent in the Pathfinder solution were used. This procedure is somewhat more subjective than the MDS approach, although many of the layout rules, such as minimizing link crossings, are relatively well-specified. For those familiar with programming, but unfamiliar with UNIX, the central role of the *C* compiler (*cc*) may seem a bit strange. An experienced programmer might reasonably expect the assembler (*as*) to occupy such a role, but probably not a particular language compiler. However, those familiar with UNIX will recognize the central importance of *C* to UNIX (e.g., UNIX is written in *C*). Other details in the Pathfinder representation can be observed in this comparison as well, such as the connections between the *C* utilities (*cb, cpp*, and *lint*) and the *C* compiler, and the particular way in which the Pascal functions (*pc, pi, pix*, and *px*) are linked. In both Figures 1 and 2 the PFNET provides more information than the corresponding hierarchical representation. Indeed, the minimum hierarchical cluster solution can be derived from the PFNET, but not vice versa (cf., Dearholt & Schvaneveldt, Chapter 1, this volume).

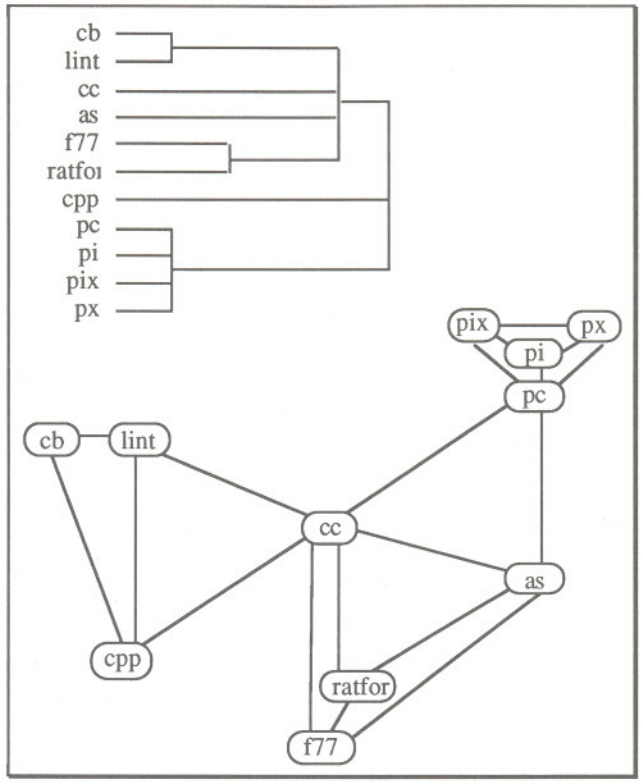

Figure 2. A comparison of hierarchical clustering (Minimum Method) and PFNET($r = \infty$, $q = n-1$) for UNIX programming functions.

Labeling Categories. The complete functional perspective requires not only valid categories, but appropriate category labels as well. Judgments of relatedness contain error, and the attempt to identify subnets, described above, is subject to the criticism that some of the clusters correspond to conceptual categories, whereas others are artifacts of the procedures employed. One approach to resolving such questions is to obtain more data from experts. Therefore, four of the most experienced UNIX users in the group were asked to judge the goodness of the clusters obtained from the hierarchical cluster analysis and to name them. Judges were shown all 83 clusters in a random sequence and asked to rate the goodness of each cluster on a five-point scale, ranging from 1 (Very Bad) to 5 (Very Good), and to provide appropriate names for all but the Very Bad clusters.

The results of the cluster-rating phase show a general tendency for the average cluster goodness to decrease as cluster size increased ($r = -.50$), with the ratings for clusters ranging in size from two to five commands averaging 4 or better (Good to Very Good). In the next step, the ratings and names supplied by the judges were used to eliminate artifactual clusters. As an example, Figure 3 shows the cluster corresponding to communications

functions, with the average ratings of the original nine clusters on the left and the reduced version with four categories on the right. Reduction was accomplished by comparing the names given by the judges for each of the smallest clusters with those given for the clusters above them in the hierarchy, and collapsing smaller clusters into larger ones when the names were the same.

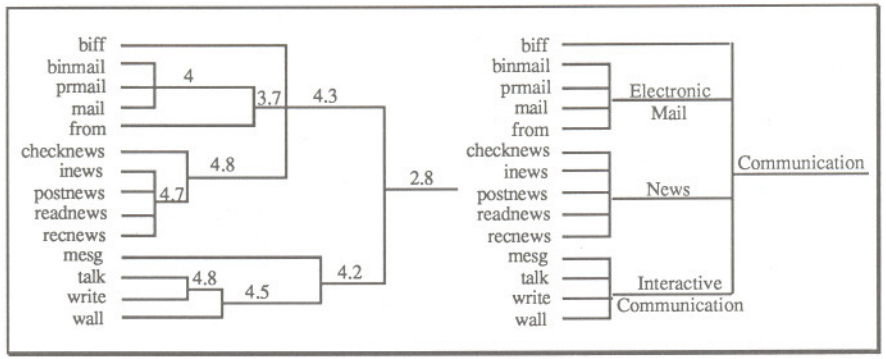

Figure 3. A hierarchical cluster analysis (Minimum Method) for UNIX communication commands with average "goodness" ratings and after reduction based on naming.

The final result of using Pathfinder to complement the hierarchical cluster analysis was 10 categories and 30 subcategories, with only 5 terms left in the miscellaneous category. It is our belief that this category structure resembles the schematic mental organization available to an experienced UNIX user. A common way of using HyBrow is to begin with the top-level category network and make successive selections first from the top-level category (see Figure 4) and then from any subcategories until the appropriate subnet of command nodes is revealed.

The Procedural Perspective: Obtaining Command-Event Protocols

Our approach to developing a procedural perspective was to obtain a number of protocols from nine experienced UNIX users. Users' *.login* files were modified to increase the size of their *history* records so that all of the events that occurred during a session were automatically captured and, at the end of each session, mailed to us. Event records were collected for a period of approximately six months and provided a database of over 75,000 command-line entries. A summary of the event-record data for the first of our two samples is shown in Table 1.

A total of 41,372 commands was obtained. Of these, approximately 58% came from section 1 of the *man* system. The number of unique commands used by individuals ranged from 42 to 133. Interestingly, as shown in Table 2, the top 10 commands accounted for 66% of all commands issued—the top three commands accounted for 32%! Although these statistics would undoubtedly change somewhat from one group of users to another (e.g., *rsh* is commonly used only when several computers are networked together), they are probably fairly typical of command usage.

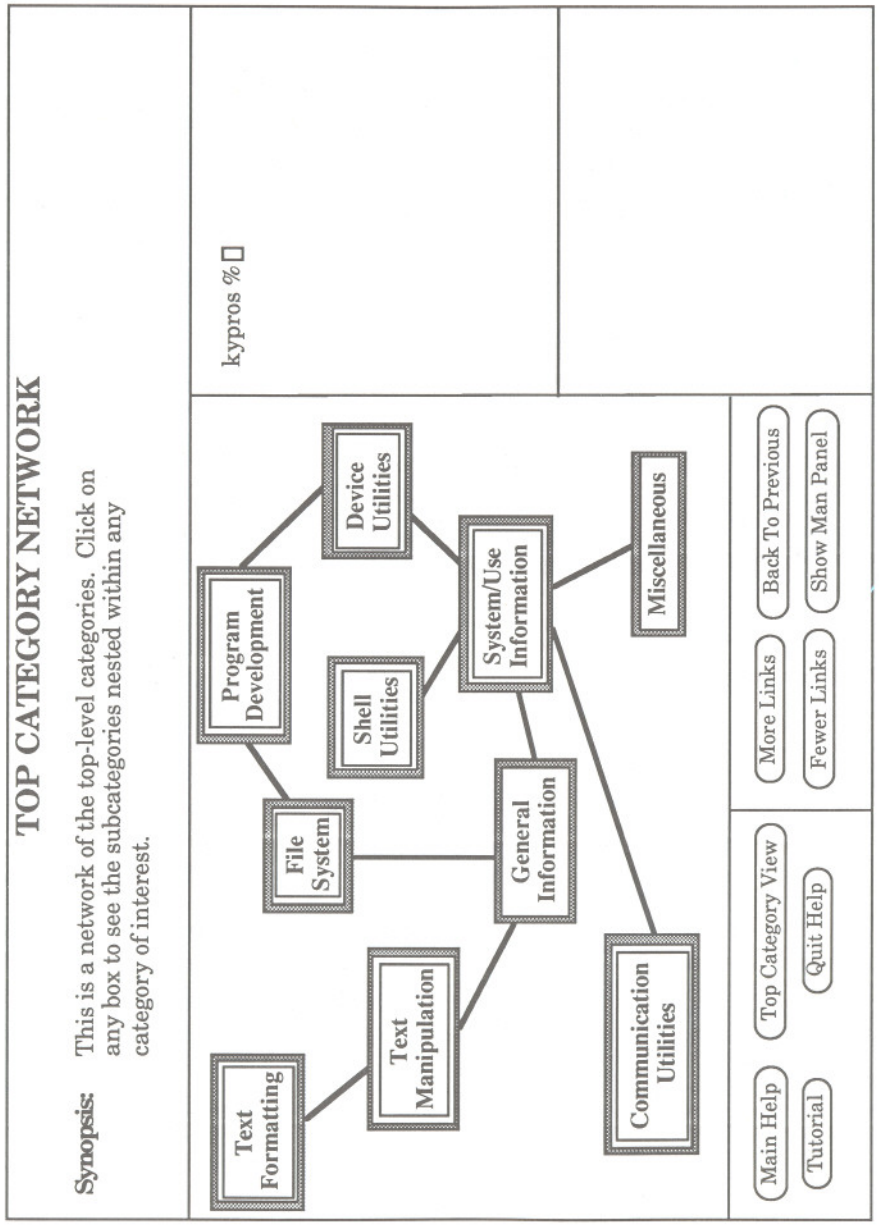

Figure 4. The top-level category network for HyBrow. Selecting any node will reveal the subcategories nested within any category of interest.

Table 1. A summary of the UNIX event record data showing the number of commands recorded for each user and the number and percentage of these commands that come from Section 1 of the *man* system.

	All Commands		*Section 1 Commands*		
User	*Unique*	*Total*	*Unique*	*Total*	*Percent*
1	58	2,124	39	1,197	56%
2	72	1,139	41	685	60%
3	47	1,370	32	814	59%
4	97	2,786	57	1,728	62%
5	42	863	26	592	69%
6	84	9,499	51	4,994	53%
7	122	15,894	75	11,225	71%
8	49	2,196	27	771	35%
9	133	5,501	84	3,178	58%
Average	78	4,597	48	2,798	58%

Table 2. The 10 most frequently used commands from the UNIX event record study.

Rank	*Command*	*Frequency*	*Percent*
1	ls	4,719	11.41%
2	cd	4,420	10.68%
3	more	3,961	9.57%
4	| (pipe)	3,372	8.15%
5	mail	2,346	5.67%
6	emacs	2,078	5.02%
7	fg	2,060	4.98%
8	pwd	1,781	4.30%
9	rsh	1,588	3.84%
10	rm	815	1.97%
Total		27,140	65.59%

Deriving a Proximity Matrix and a Procedural Network. The event records from all of the users were combined into a composite frequency of co-occurrence matrix by incrementing the appropriate cells each time a particular command followed another in an event record.[4] The data were then converted to conditional probabilities such that large conditional probabilities produced small proximities in the matrix, whereas small conditional probabilities produced large proximities.

After converting the frequency of co-occurrence data into conditional probabilities, a subset of 49 commands were submitted to Pathfinder analysis ($r = \infty$, $q = n–1$). The network of commands selected consisted of those that occurred in the event records of at least half of the users. The purpose of this analysis was to identify task *sequences*. Figure 5 shows a portion of this network that includes all commands within two links (arcs) of *kill*. As an example of the type of information contained in the network—and its potential utility—suppose several programs were executing "simultaneously." Further, suppose that one of these programs was a particularly time-consuming analysis that had to be terminated for some reason (e.g., to reduce processor load). This is one of the few cases where the name of the appropriate UNIX command is fairly easy to remember (i.e., *kill*). However, in order to *kill* a program, you have to know its job number, and the name of the command that returns job numbers is not so easy to remember. From the network in Figure 5 it is apparent that only one command, *ps*, frequently precedes *kill* (i.e., there is only one arc leading to *kill*). Indeed, *ps* (program status) is the command that returns job numbers, along with some other useful information.

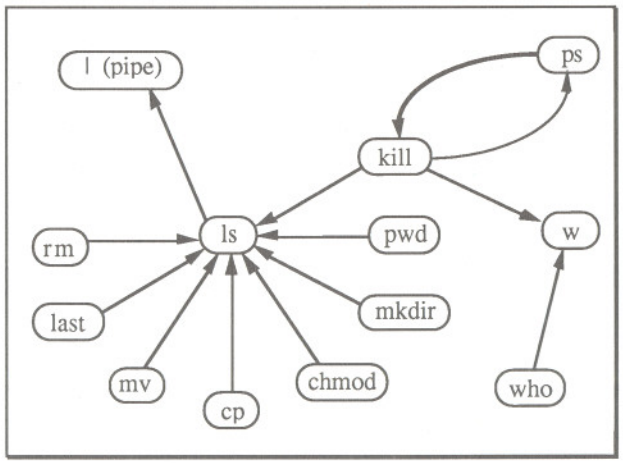

Figure 5. A Pathfinder analysis [PFNET($r = \infty$, $q = n–1$)] of UNIX event-record data, including all commands within two links (arcs) of *kill*.

[4]This is a relatively shallow method for extracting information from these data because only two-step sequences are considered.

The PFNETs derived from the two sets of proximity measures generate two different perspectives on the UNIX system. HyBrow attempts to provide users with a consistent documentation interface, relying on contextual cues provided by the network representation to facilitate the development of conceptual models. Either the functional or procedural perspective can be presented to the user in the form of networks with concepts as nodes and relationships among the concepts as links or arcs. In addition, there can be multiple levels of abstraction within perspectives, as described for the functional perspective. For example, Figure 4 shows the highest level of abstraction for the functional perspective.

A user can call upon HyBrow to display a network of commands that are functionally associated with any command of interest (see Figure 6). When a specific command is selected, a brief synopsis and description of the syntax for the command-in-focus appear in the window above the network. As shown in Figure 6, the user has selected the *cmp* command. The complete *man* panel for the selected command can be retrieved and displayed in the interactive window to the right of the network by clicking on the *show man panel* button. Technical terms that appear in the synopsis are highlighted, indicating Hypertext options. Selecting a highlighted term leads to the display of its glossary definition in the interactive window. This is illustrated in Figure 6 for a user who wants additional information about *standard input*.

The user can easily shift HyBrow's focus to any of the related commands in the network by clicking on a visible command node. The user has control over the network density through the use of the *more links* and *fewer links* buttons. HyBrow can also identify those commands that belong to the same category or the same supercategory. The Hypertext engine maintains a record of execution that permits the user to backtrack in stepwise fashion or to jump back to the beginning of the session.

A command subnet, like the one shown in Figure 6, can be accessed in two different ways. If the user wants help for a command he already knows, then the command subnet can be accessed directly by entering the command's name on any prompt line. Commands can also be accessed from a top-down search of the category network shown in Figure 4.

Conclusions

We are still a long way from Vannevar Bush's *memex*—but technology *has* provided us with the means to organize information in new and potentially useful ways. Considerably more research needs to be conducted into the utility of hypertext, both basic and applied, but the preliminary indications are promising: Providing readers with multiple perspectives facilitates their ability to draw inferences, and even superficially hierarchical systems may have naturally nonhierarchical representations (e.g., UNIX).

We have presented an interface design methodology that seems particularly suited to building hypertext networks, along with accurate "maps." It remains to be seen whether or not this methodology leads to better interfaces, but a number of advantages are already clear. First, the methodology is formal, meaning that it is well-specified enough that different designers following essentially the same set of procedures will arrive at essentially the same results. Second, it is empirically based, meaning that the method does not rely on the subjective impressions of designers but the mental representations of users.

In this chapter we have attempted to identify some of the potential and challenge associated with presenting information in hypertext systems. Although the current enthusiasm for hypertext may wain, there do appear to be legitimate reasons to believe that collections of information should be represented nonhierarchically.

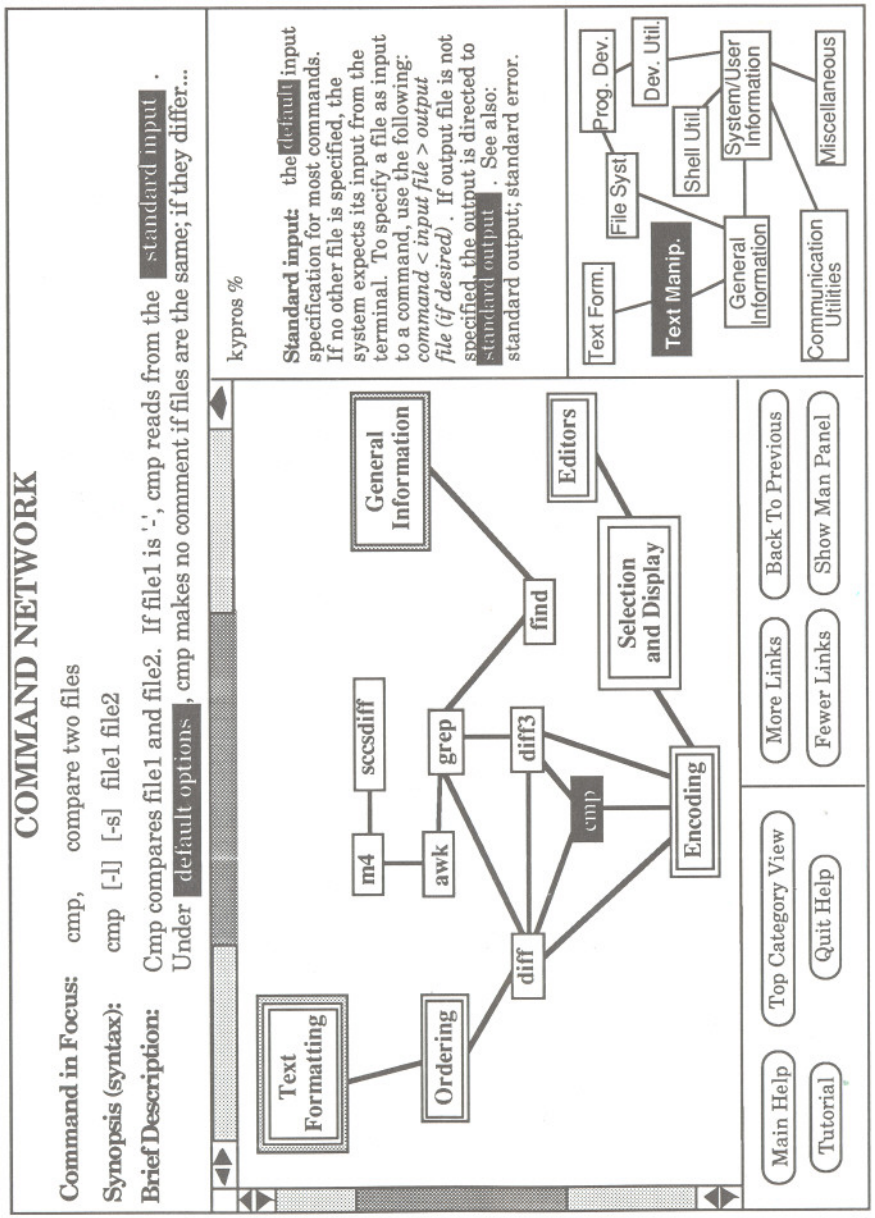

Figure 6. A bottom-level command subnet for HyBrow. Selecting any command node automatically retrieves a brief description of the command-in-focus and makes other information available.