Chapter 10

User Interfaces and Visualization

by Marti A. Hearst

10.1 Introduction

This chapter discusses user interfaces for communication between human information seekers and information retrieval systems. Information seeking is an imprecise process. When users approach an information access system they often have only a fuzzy understanding of how they can achieve their goals. Thus the user interface should aid in the understanding and expression of information needs. It should also help users formulate their queries, select among available information sources, understand search results, and keep track of the progress of their search.

The human-computer interface is less well-understood than other aspects of information retrieval, in part because humans are more complex than computer systems, and their motivations and behaviors are more difficult to measure and characterize. The area is also undergoing rapid change, and so the discussion in this chapter will emphasize recent developments rather than established wisdom.

The chapter will first outline the human side of the information seeking process and then focus on the aspects of this process that can best be supported by the user interface. Discussion will encompass current practice and technology, recently proposed innovative ideas, and suggestions for future areas of development.

Section 10.2 outlines design principles for human-computer interaction and introduces notions related to information visualization. Section 10.3 describes information seeking models, past and present. The next four sections describe user interface support for starting the search process, for query specification, for viewing retrieval results in context, and for interactive relevance feedback. The last major section, Section 10.8, describes user interface techniques to support the information access process as a whole, and Section 10.9 concludes the chapter. Figure 10.1 presents the flow the chapter contents.

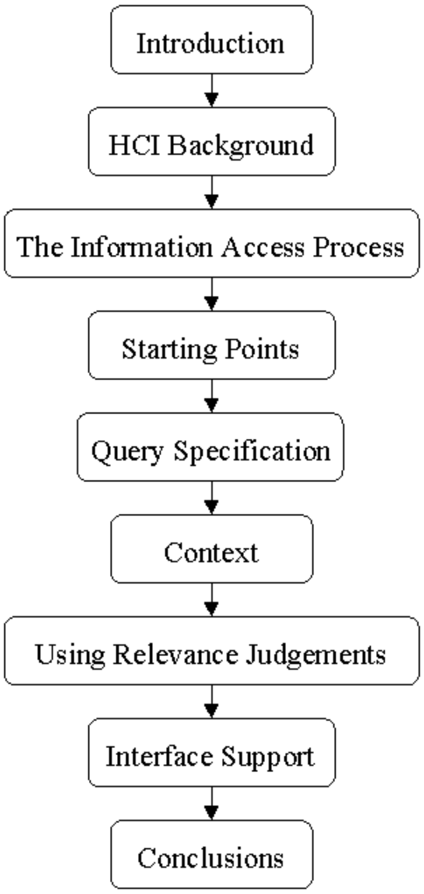

Figure 10.1: The flow of this chapter's contents.

10.2 Human-Computer Interaction

What makes an effective human-computer interface? Ben Shneiderman, an expert in the field, writes [168, p. 10]]:

> Well designed, effective computer systems generate positive feelings of success, competence, mastery, and clarity in the user community. When an interactive system is well-designed, the interface almost disappears, enabling users to concentrate on their work, exploration, or pleasure.

As steps towards achieving these goals, Shneiderman lists principles for design of user interfaces. Those which are particularly important for information access include (slightly restated): provide informative feedback, permit easy reversal of actions, support an internal locus of control, reduce working memory load, and provide alternative interfaces for novice and expert users. Each of these principles should be instantiated differently depending on the particular interface application. Below we outline the impact of those principles that are of special interest to information access systems.

10.2.1 Design Principles

Offer informative feedback. This principle is especially important for information access interfaces. In this chapter we will see current ideas about how to provide users with feedback about the relationship between their query specification and documents retrieved, about relationships among retrieved documents, and about relationships between retrieved documents and metadata describing collections. If the user has control of how and when feedback is provided, then the system provides an **internal locus of control**.

Reduce working memory load. Information access is an iterative process, the goals of which shift and change as information is encountered. One key way information access interfaces can help with memory load is to provide mechanisms for keeping track of choices made during the search process, allowing users to return to temporarily abandoned strategies, jump from one strategy to the next, and retain information and context across search sessions. Another memory-aiding device is to provide browsable information that is relevant to the current stage of the information access process. This includes suggestions of related terms or metadata, and search starting points including lists of sources and topic lists.

Provide alternative interfaces for novice and expert users. An important tradeoff in all user interfaces design is that of simplicity versus power. Simple interfaces are easier to learn, at the expense of less flexibility and sometimes less efficient use. Powerful interfaces allow a knowledgeable user to do more and have more control over the operation of the interface, but can be time-consuming to learn and impose a memory burden on people who use the system only intermittently. A common solution is to use a "scaffolding" technique [159]. The novice user is presented with a simple interface that can be learned quickly and that provides the basic functionality of the application, but is restricted in power and flexibility. Alternative interfaces are offered for more experienced users, giving them more control, more options, and more features, or potentially even entirely different interaction models. Good user interface design provides intuitive bridges between the simple and the advanced interfaces.

Information access interfaces must contend with special kinds of simplicity/power tradeoffs. One such tradeoff is the amount of information shown about the workings of the search system itself. Users who are new to a system or to a particular collection may not know enough about the system or the domain associated with the collection to make choices among complex features. They may not know how best to weight terms, or in the case of relevance feedback, not know what the effects of reweighting terms would be. On the other hand, users that have worked with a system and gotten a feeling for a topic are likely to be able to choose among suggested terms to add to their query in an informed manner. Determining how much information to show the user of the system is a major design choice in information access interfaces.

10.2.2 The Role of Visualization

The tools of computer interface design are familiar to most computer users today: windows, menus, icons, dialog boxes, and so on. These make use of bit-mapped display and computer graphics to provide a more accessible interface than command-line based TTY displays. A less familiar but growing area is that of **information visualization**, which attempts to provide visual depictions of very large information spaces.

Humans are highly attuned to images and visual information [178, 100, 108]. Pictures and graphics can be captivating and appealing, especially if well-designed. A visual representation can communicate some kinds of information much more rapidly and effectively than any other method. Consider the difference between a written description of a person's face and a photograph of it, or the difference between a table of numbers containing a correlation and a scatter plot showing the same information.

The growing prevalence of fast graphics processors and high resolution color monitors is increasing interest information visualization. Scientific visualization, a rapidly advancing branch of this field, maps physical phenomena onto two or three-dimensional representations [92]. An example of scientific visualization is a colorful image of the pattern of peaks and valleys on the ocean floor; this provides a view of physical phenomena for which a photograph cannot (currently) be taken. Instead, the image is constructed from data that represent the underlying phenomena.

Visualization of inherently abstract information is more difficult, and visualization of textually represented information is especially challenging. Language is our main means of communicating abstract ideas for which there is no obvious physical manifestation. What does a picture look like that describes negotiations over a trade agreement in which one party demands concessions on environmental policies while the other requires help in strengthening its currency?

Despite the difficulties, researchers are attempting to represent aspects of the information access process using information visualization techniques. Some of these will be described later in this chapter. Aside from using **icons** and **color highlighting**, the main information visualization techniques include **brushing and linking** [54, 181], **panning and zooming** [16], **focus-plus-context** [112], **magic lenses** [19], and the use of **animation** to retain context and help make occluded information visible [157, 29]. These techniques support dynamic, interactive use. Interactivity seems to be an especially important property for visualizing abstract information, although it has not played as large a role within scientific visualization.

Brushing and linking refers to the connecting of two or more views of the same data, such that a change to the representation in one view effects the representation in the other views as well. For example, say a histogram is shown in one window representing the years of publication for a set of documents, and another window depicts this document set as a list of titles in order of relevance. An example of brushing and linking is a mechanism by which the user's assigning the color red to a bar of the histogram (brushing it with red) causes the titles of the documents that were published in the corresponding year to also be colored red.

Panning and zooming refers to the actions of a movie camera that can scan sideways across a scene (panning) or move in for a closeup or back away to get a wider view (zooming). For example, text clustering can be used to show a top-level view of the main themes in a document collection, and zooming can be used to move "closer," showing individual documents represented as icons, and then move in closer still to see the text associated with a single document.

When zooming is used, the more detail that is visible about a particular item, the less can be seen about the surrounding items. Focus-plus-context is used to partly alleviate this effect. The idea is to make one portion of the view – the focus of attention – larger, while simultaneously shrinking the surrounding objects. The farther an object is from the focus of attention, the smaller it is made to appear, like the effect seen in a fisheye camera lens (also in some door peepholes).

Magic lenses are directly manipulable transparent windows that, when overlayed on some other data type, cause a transformation to be applied to the underlying data, thus changing its appearance. The most straightforward application of magic lenses is for drawing tasks, and it is especially useful if used as a two-handed interface. For example, the left hand can be used to position a color lens over a drawing of an object. The right hand is used to mouse-click on the lens, thus causing the appearance of the underlying object to be transformed to the color specified by the lens.

Additionally, there are a large number of graphical methods for depicting trees and hierarchies, some of which make use of animation to show nodes that would otherwise be occluded (hidden from view by other nodes) [62, 81, 89, 106, 157].

It is often useful to combine these techniques into an interface layout consisting of an **overview plus details** [64, 149]. An overview, such as a table-of-contents of a large manual, is shown in one window. A mouse-click on the title of the chapter causes the text of the chapter itself to appear in another window, in a linking action. Panning and zooming or focus-plus-context can be used to change the view of the contents within the overview window.

10.2.3 Evaluating Interactive Systems

From the viewpoint of user interface design, people have widely differing abilities, preferences, and predilections. Important differences for information access interfaces include relative spatial ability and memory, reasoning abilities, verbal aptitude, and (potentially) personality differences [49, 168]. Age and cultural differences can contribute to acceptance or rejection of interface techniques [129]. An interface innovation can be useful and pleasing for some users, and foreign and cumbersome for others. Thus software design should allow for flexibility in interaction style, and new features should not be expected to be equally helpful for all users.

An important aspect of human-computer interaction is the methodology for evaluation of user interface techniques. Precision and recall measures have been widely used for comparing the ranking results of non-interactive systems, but are less appropriate for assessing interactive systems [105]. This results in part because of a strong topic-system interactions; the kind of question being asked can have a strong effect on the performance of the system. Also, the standard measures of precision and recall evaluate the system at high recall levels; in the TREC tasks systems are compared to see how well they return the top 1000 documents (see chapter **??**. In many interactives settings, users require only a few relevant documents.

For evaluating highly interactive information access systems, useful metrics beyond precision and recall include: time required to learn the system, time required to achieve goals on benchmark tasks, error rates, and retention of the use of the interface over time. Throughout this chapter, empirical results of user studies are presented whenever they are available.

Empirical data involving human users is time-consuming to gather and difficult to draw conclusions from. This is due in part to variation in users' characteristics and motivations, and in part to the broad scope of information access activities. Formal psychological studies usually only uncover narrow conclusions about restricted contexts. For example, interaction moves such as the length of time it takes for a user to select an item from a fixed menu under various conditions have been characterized empirically [28], but variations in interaction behavior for complex tasks like information access are difficult to account for accurately. Nielsen [138] advocates a more informal evaluation approach (called heuristic evaluation) in which user interface affordances are assessed in terms of more general properties and without concern about statistically significant results.

10.3 The Information Access Process

A person engaged in an information seeking process has one or more **goals** in mind and uses a search system as a tool to help achieve those goals. Goals requiring information access can range quite widely, from finding a plumber to keeping informed about a business competitor, from writing a publishable scholarly article to investigating an allegation of fraud.

Information access **tasks** are used to achieve these goals. These tasks span the spectrum from asking specific questions to exhaustively researching a topic. Other tasks fall between these two extremes. A study of business analysts [140] found three main kinds of information seeking tasks: monitoring a well-known topic over time (such as researching competitors' activities each quarter), following a plan or stereotyped series of searches to achieve a particular goal (such as keeping up to date on good business practices), and exploring a topic in an undirected fashion (as when getting to know an unfamiliar industry). Although the goals differ, there is a common core revolving around the information seeking component, which is our focus here.

10.3.1 Models of Interaction

Most accounts of the information access process assume an interaction cycle consisting of query specification, receipt and examination of retrieval results, and then either stopping or reformulating the query and repeating the process until a perfect result set is found [162, 169]. In more detail, the standard process can be described according to the following sequence of steps (see Figure 10.2):

1. Start with an information need
2. Select a system and collections to search on
3. Formulate a query
4. Send the query to the system
5. Receive the results in the form of information items
6. Scan, evaluate, and interpret the results
7. Either stop or
8. Reformulate the query and go to step 4.

This simple interaction model used by web search engines is the only model that most information seekers see today. This model does not take into account the fact that many users dislike being confronted with a long disorganized list of retrieval results that do not directly address their information needs. It also contains an underlying assumption that the users' information need is static and the information seeking process is one of successively refining a query until it retrieves all and only those documents relevant to the original information need.

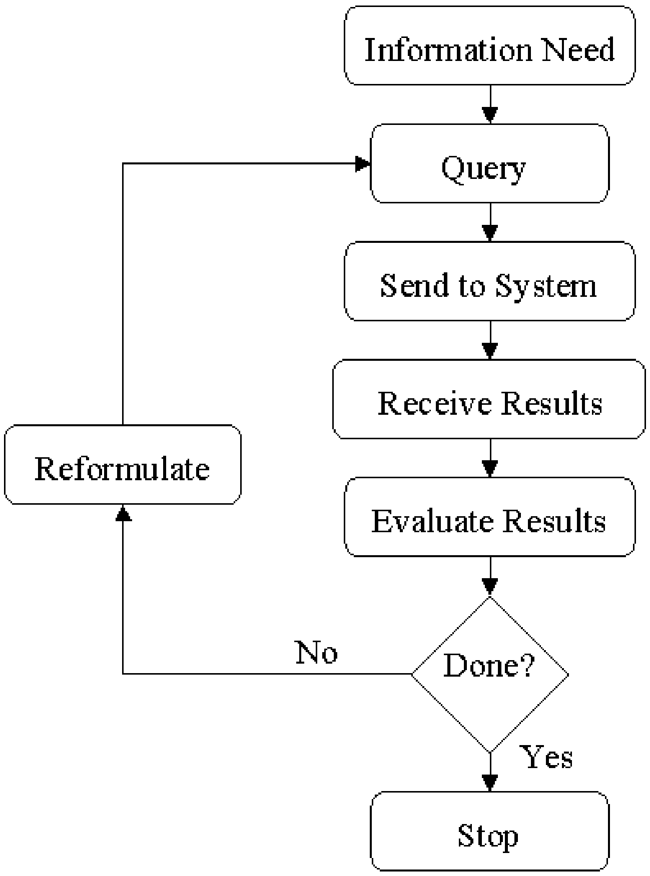

Figure 10.2: Models of information access processes.

One way users learn during the search process is by scanning information, such as reading the titles of result sets of documents, reading retrieved documents themselves, viewing lists of topics related to their query terms, and navigating within hyperlinked web sites. The recent advent of hyperlinks as a pivotal part of the information seeking process makes it no longer feasible to ignore the role of scanning and navigation within the search process itself. In particular, today a near-miss is much more acceptable than it was with bibliographic search, since an information seeker using the web can navigate from a near miss in the hopes that a useful page will be a few links away.

The standard model also downplays the interaction that takes place when the user scans terms suggested as a result of relevance feedback, scans thesaurus structures, or views thematic overviews of document collections. It de-emphasizes the role of source selection, which is increasingly important now that, for the first time, tens of thousands of information collections are immediately reachable for millions of people.

Thus while useful for describing the basics of information access systems, this simple interaction model is being challenged on many fronts [14, 140, 22, 82, 40]. Bates [14] proposes the "berry-picking" model of information seeking, which has two main points. The first is that, as a result of reading and learning from the information encountered throughout the search process, the users' information needs, and consequently their queries, continually shift. Information encountered at one point in a search may lead in a new, unanticipated direction. The original goal may become partly fulfilled, thus lowering the priority of one goal in favor of another. This is posed in contrast to the assumption of "standard" information retrieval that the users' information need remains the same throughout the search process. The second point is that users' information needs are not satisfied by a single, final retrieved set of documents, but rather by a series of selections and bits of information found along the way. This is in contrast to the assumption that the main goal of the search process is to hone down the set of retrieved documents into a perfect match of the original information need.

The berry-picking model is supported by a number of observational studies [55, 22], including that of O'Day and Jeffries [140]. They found that the information seeking process consisted of a series of interconnected but diverse searches on one problem-based theme. They also found that search results for a goal tended to trigger new goals, and hence search in new directions, but that the context of the problem and the previous searches was carried from one stage of search to the next. They also found that the main value of the search resided in the accumulated learning and acquisition of information that occurred during the search process, rather than in the final results set.

Thus, a user interface for information access should allow users to reassess their goals and adjust their search strategy accordingly. A related situation occurs when users encounter a "trigger" that causes them to pursue a different strategy temporarily, perhaps to return to the current unfinished activity at a later time. An implication of these observations is that the user interface should support search strategies by making it easy to follow trails with unanticipated results. This can be accomplished in part by supplying ways to record the progress of the current strategy and to store, find, and re-load intermediate results, and supporting pursuit of multiple strategies simultaneously.

The user interface should also support methods for monitoring the status of the current strategy in relation to the users' current task and high-level goals. One way to cast the activity of monitoring the progress of a search strategy relative to a goal or subgoal is in terms of a cost/benefit analysis, or an analysis of diminishing returns [161]. This kind of analysis assumes that at any point in the search process, the user is pursuing the strategy that has the highest expected utility. If, as a consequence of some local tactical choices, another strategy presents itself as being of higher utility than the current one, the current one is (temporarily or permanently) abandoned in favor of the new strategy.

There are a number of theories and frameworks that contrast **browsing, querying, navigating,** and **scanning** along several dimensions [18, 34, 123, 183]. Here we assume that users scan information structure, be it titles, thesaurus terms, hyperlinks, category labels, or the results of clustering, and then either select a displayed item for some purpose (to read in detail, to use as input to a query, to navigate to a new page of information) or formulate a query (either by recalling potential words or by selecting categories or suggested terms that have been scanned). In both cases, a new set of information is then made viewable for scanning. Queries tend to produce new, ad hoc collections of information that have not been gathered together before, whereas selection retrieves information that has already been composed or organized. Navigation refers to following a chain of links, switching from one view to another, toward some goal, in a sequence of scan and select operations. Browsing refers to the casual, mainly undirected exploration of information structures, and is usually done in tandem with selection, although queries can also be used to create subcollections to browse through. An important aspect of the interaction process is that the output of one action should be easily used as the input to the next.

10.3.2 Non-Search Parts of the Information Access Process

The O'Day and Jeffries study [140] found that information seeking is only one part of the full work process their subjects were engaged in. In between searching sessions many different kinds of work was done with the retrieved information, including reading and annotating [142] and analysis. O'Day and Jeffries examined the analysis steps in more detail, finding that 80% of this work fell into six main types: finding trends, making comparisons, aggregation, identifying a critical subset, assessing, and interpreting. The remainder consisted of cross-referencing, summarizing, finding evocative visualizations for reports, and miscellaneous activities. The Sensemaking work of Russell et al. [161] also discusses information work as a process in which information retrieval plays only a small part. They observe that most of the effort made in Sensemaking is in the synthesis of a good representation, or way of thinking about, the problem at hand. They describe the process of developing a way to formulate and crystallize the important concepts for a given task.

From these observations it is convenient to divide the entire information access process into two main components: search/retrieval, and analysis/synthesis of results. User interfaces should allow both kinds of activity to be tightly interwoven. However, analysis/synthesis are activities that can be done independently of information seeking, and for our purposes it is useful to make a distinction between the two types of activities.

10.3.3 Earlier Interface Studies

The bulk of the literature on studies of human-computer information seeking behavior concerns information intermediaries using online systems consisting of bibliographic records (e.g., [124, 164, 21]), sometimes with costs assessed per time unit. Unfortunately, many of the assumptions behind those studies do not reflect the conditions of modern information access [68, 46]. The differences include the following:

- The text being searched now is often full text rather than bibliographic citations, and thus in many cases questions can be answered directly. Because users have access to full text, rather than document surrogates, it is more likely that simple queries will find relevant answers directly as part of the search process.
- Most studies were performed on Boolean systems whereas modern systems use statistical ranking (which is more effective when abstracts and full text are available than when only titles and citations are available).
- Much of modern searching is done by end users, many new to online searching, rather than professional intermediaries, which were the focus of many of the earlier studies.
- Tens of thousands of sources are now available online on networked information systems, and many are tightly coupled via hyperlinks, as opposed to being stored in separate collections owned by separate services. Earlier studies generally used systems in which moving from one collection to another required prior knowledge of the collections and considerable time and effort to switch. A near miss is much more useful in this hyperlinked environment than in earlier systems, since hyperlinks allow users to navigate from the near miss directly to the source containing information of interest. In a card catalog environments, where documents are represented as isolated units, a near miss consists of finding a book in the general area of interest and then going to the bookshelf in the library to look for related books, or obtaining copies of many issues of a journal and scanning for related articles.
- Finally, most users have access to bit-mapped displays allowing for direct manipulation, or at least form fillin. Most earlier studies and bibliographic systems were implemented on TTY displays, which require command-line based syntax and do a poor job of retaining context.

Despite these significant differences, some general information seeking strategies have been identified that seem to transfer across systems. Additionally, although modern systems have remedied many of the problems of earlier online public access catalogs, they also introduce new problems of their own.

10.4 Starting Points

Search interfaces must provide users with good ways to get started. An empty screen or a blank entry form does not provide clues to help a user decide how to start the search process. Users usually do not begin by creating a long, detailed expression of their information need. Studies show that users tend to start out with very short queries, inspect the results, and then modify those queries in an incremental feedback cycle [7]. The initial query can be seen as a kind of "testing the water" to see what kinds of results are returned and get an idea of how to reformulate the query [183, 14]. Thus one task of an information access interface is to help users select the sources and collections to search on.

For example, there are many different information sources associated with cancer, and there are many different kinds of information a user would like to know about cancer. Guiding the user to the right set of starting points can help with the initial problem formulation. Traditional bibliographic search assumes that the user begins by looking through a list of names of sources and choosing which collections to search on, while web search engines obliterate the distinctions between sources and plunge the user into the middle of a web site with little information about the relationship of the search hit to the rest of the collection. In neither case is the interface to the available sources particularly helpful.

In this section we will discuss four main ways to get started in search: **lists, overviews, examples,** and **automated source selection**.

10.4.1 Lists of Collections

Typical online systems such as Lexis-Nexis require users to begin any inquiry with a scan through a long list of source names and guess which ones will be of interest. Usually little information beyond the name of the collection is provided online for these sources (see Figure 10.3). If the user is not satisfied with the results on one collection, they must reissue the query on another collection.

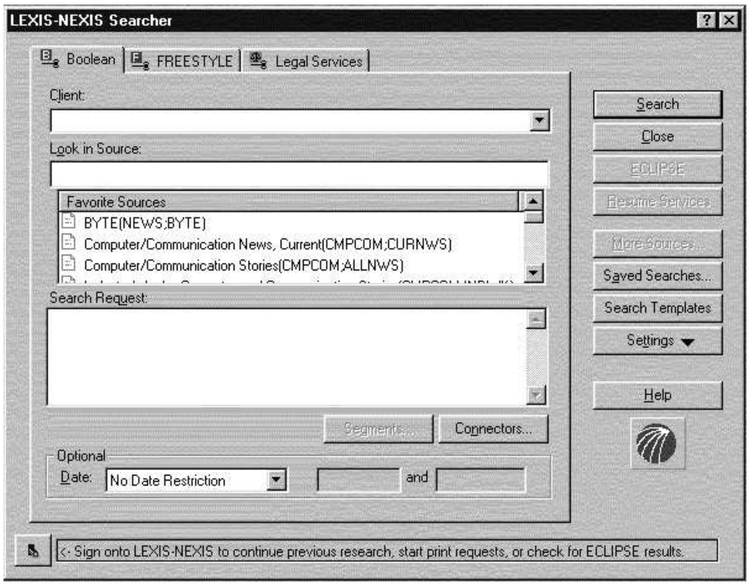

Figure 10.3: Lexis-Nexis source selection screen.

Frequent searchers eventually learn a set of sources that are useful for their domains of interest, either through experience, formal training, or recommendations from friends and colleagues. Often-used sources can be stored on a "favorites" list, also known as a bookmark list or a hotlist on the web. Recent research explores the maintenance of a personalized information profile for users or workgroups, based on the kinds of information they've used in the past [60].

However, when a user wants to search outside their domains of expertise, a list of familiar sources is not sufficient. Professional searchers such as librarians learn through experience and years of training which sources are appropriate for various information needs. The restricted nature of traditional interfaces to information collections discourages exploration and discovery of new useful sources. However, recently researchers have devised a number of mechanisms to help users understand the contents of collections as a way of getting started in their search.

10.4.2 Overviews

Faced with a large set of text collections, how can a user choose which to begin with? One approach is to study an overview of the contents of the collections. An overview can show the topic domains represented within the collections, to help users select or eliminate sources from consideration. An overview can help users get started, directing them into general neighborhoods, after which they can navigate using more detailed descriptions. Shneiderman [167] advocates an interaction model in which the user begins with an overview of the information to be worked with, then pans and zooms to find areas of potential interest, and then view details. The process is repeated as often as necessary.

Three types of overviews are discussed in this subsection. The first is display and navigation of large topical **category hierarchies** associated the documents of a collection. The second is automatically derived overviews, usually created by unsupervised **clustering techniques** on the text of documents, that attempt to extract overall characterizing themes from collections. The third type of overview is that created by applying a variant of **cocitation analysis** on connections or links between different entities within a collection. Other kinds of overviews are possible, for example, showing a bookshelves or piles of books graphically [158, 8].

Category or Directory Overviews

There exist today many large online text collections to which category labels have been assigned. Traditional online bibliographic systems have for decades assigned subject headings to books and other documents [173]. MEDLINE, a large collection of biomedical articles, has associated with it Medical Subject Headings (MeSH) consisting of approximately 18,000 categories [115]. The Association for Computing Machinery (ACM) has developed a hierarchy of approximately 1200 category (keyword) labels.[1] Yahoo[2], one of the most popular search sites on the World Wide Web, organizes web pages into a hierarchy consisting of thousands of category labels.

The popularity of Yahoo and other web directories suggests that hierarchically structured categories are useful starting points for users seeking information on the web. This popularity may reflect a preference to begin at a logical starting point, such as the home page for a set of information, or it may reflect a desire to avoid having to guess which words will retrieve the desired information. (It may also reflect the fact that directory services attempt to cull out low-quality web sites.)

The meanings of category labels differ somewhat among collections. Most are designed to help organize the documents and to aid in query specification. Unfortunately, users of online bibliographic catalogs rarely use the available subject headings [68, 46]. Hancock-Beaulieu and Drabenstott and Weller, among others, put much of the blame on poor (command line-based) user interfaces which provide little aid for selecting subject labels and require users to scroll through long alphabetic lists. Even with graphical web interfaces, finding the appropriate place within a

[1] http://www.acm.org/class/1991/cr91.html
[2] http://www.yahoo.com

category hierarchy can be a time-consuming task, and once a collection has been found using such a representation, an alternative means is required for searching within the site itself.

Most interfaces that depict category hierarchies graphically do so by associating a document directly with the node of the category hierarchy to which it has been assigned. For example, clicking on a category link in Yahoo brings up a list of documents that have been assigned that category label. Conceptually, the document is stored within the category label. When navigating the results of a search in Yahoo, the user must look through a list of category labels and guess which one is most likely to contain references to the topic of interest. A wrong path requires backing up and trying again, and remembering which pages contain which information. If the desired information is deep in the hierarchy, or not available at all, this can be a time-consuming and frustrating process. Because documents are conceptually stored "inside" categories, users cannot create queries based on combinations of categories using this interface.

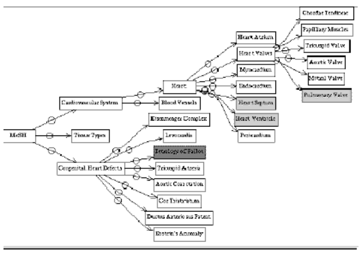

Figure 10.4: The MeshBrowse interface for viewing category labels hierarchically (from Korn and Shneiderman 1995)

It is difficult to design a good interface to integrate category selection into query specification, in part because display of category hierarchies takes up large amounts of screen space. For example, Internet Grateful Med[3] is a web-based service that allows an integration of search with display and selection of MeSH category labels. After the user types in the name of a potential category label, a long list of choices is shown in a page. To see more information about a given label, the user selects a link (e.g., Radiation Injuries). The causes the context of the query to disappear because a new web page appears showing the ancestors of the term and its immediate descendants. If the user attempts to see the siblings of the parent term (Wounds and Injuries) then a new page appears

[3] http://igm.nlm.nih.gov:80/

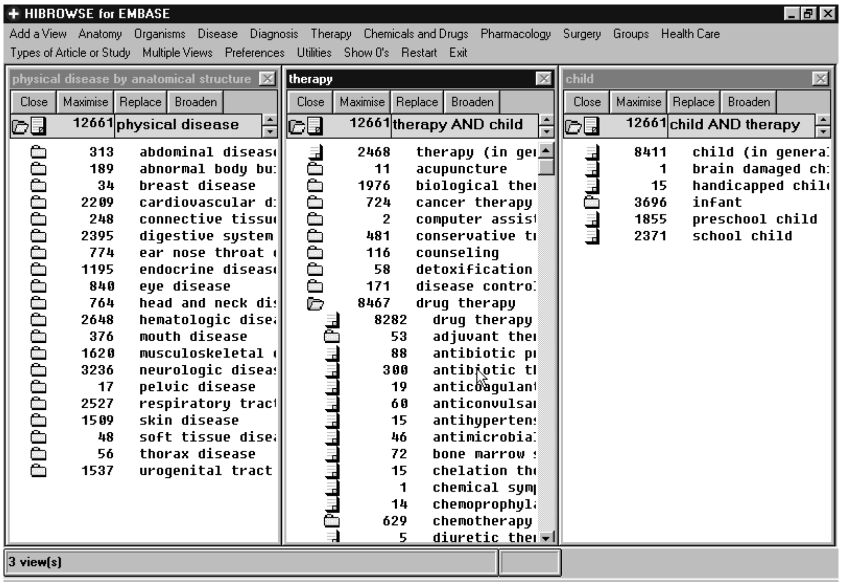

Figure 10.5: The HiBrowse interface for viewing category labels hierarchically and according to facets (from Pollitt 1997).

that changes the context again. Radiation Injuries appears as one of many siblings and its children can no long be seen. To go back to the query, the illustration of the category hierarchy disappears.

The MeSHBROWSE system [99] allows users to interactively browse a subset of semantically associated links in the MeSH hierarchy. From a given starting point, clicking on a category causes the associated categories to be displayed in a 2D tree representation. Thus only the relevant subset of the hierarchy is shown at one time, making browsing of this very large hierarchy a more tractable endeavor. The interface has the space limitations inherent in a 2D hierarchy display and does not provide mechanisms for search over an underlying document collection. See Figure 10.4.

The HIBROWSE system [150] represents category metadata more efficiently by allowing users to display several different subsets of category metadata simultaneously. The user first selects which attribute type (or facet, as attributes are called in this system) to display, for example, the user may first choose the "physical disease" value for the Disease facet. The categories that appear one level below this are shown along with the number of documents that contain each category. The user can then select other attribute types, such as Therapy and Groups (by age). The number of documents that contain attributes from all three types are shown. If the user now selects a refinement of one of the categories, such as the "child" value for the Groups attribute, then the number of documents that contain all three selected facet types a shown. At the same time, the numbers of documents containing the subcategories found below "physical disease" and "therapy (general)" are updated to reflect this more restricted specification. See Figure 10.5. A problem with the HIBROWSE system is that it requires users to navigate through the category hierarchy, rather than specify queries directly. In other words, query specification is not tightly coupled with display of category metadata. As a solution to some of these problems, the Cat-a-Cone interface

[78] will be described in Section 10.8.

Automatically Derived Collection Overviews

Many attempts to display overview information have focused on automatically extracting the most common general themes that occur within the collection. These themes are derived via the use of unsupervised analysis methods, usually variants of document clustering. Clustering organizes documents into groups based on similarity to one another; the centroids of the clusters determine the themes in the collections.

The Scatter/Gather browsing paradigm [43, 42] clusters documents into topically-coherent groups, and presents descriptive textual summaries to the user. The summaries consist of topical terms that characterize each cluster generally, and a set of typical titles that hint at the contents of the cluster. Informed by the summaries, the user may select a subset of clusters that seem to be of most interest, and re-cluster their contents. Thus the user can examine the contents of each subcollection at progressively finer granularity of detail. The reclustering is computed on-the-fly; different themes are produced depending on the documents contained in the subcollection to which clustering is applied. The choice of clustering algorithm influences what clusters are produced, but no one algorithm has been shown to be particularly better than the rest when producing the same number of clusters [186].

A user study [147] showed that the use of Scatter/Gather on a large text collection successfully conveys some of the content and structure of the corpus. However, that study also showed that Scatter/Gather without a search facility was less effective than a standard similarity search for finding relevant documents for a query. That is, subjects allowed only to navigate, not to search over, a hierarchical structure of clusters covering the entire collection were less able to find documents relevant to the supplied query than subjects allowed to write queries and scan through retrieval results.

It is possible to integrate Scatter/Gather with conventional search technology by applying clustering on the results of a query to organize the retrieved documents (see Figure 10.6). An off-line experiment [79] suggests that clustering may be more effective if used in this manner. The study found that documents relevant to the query tend to fall mainly into one or two out of five clusters, if the clusters are generated from the top-ranked documents retrieved in response to the query. The study also showed that precision and recall were higher within the best cluster than within the retrieval results as a whole. The implication is that a user might save time by looking at the contents of the cluster with the highest proportion of relevant documents and at the same time avoiding those clusters with mainly non-relevant documents. Thus clustering of retrieval results may be useful for helping direct users to a subset of the retrieval results that contain a large proportion of the relevant documents.

General themes do seem to arise from document clustering, but the themes are highly dependent on the makeup of the documents within the clusters [79, 77]. The unsupervised nature of clustering can result in a display of topics at varying levels of description. For example, clustering a collection of documents about computer science might result in cluster containing documents about artificial intelligence, computer theory, computer graphics, computer architecture, programming languages, government, and legal issues. The latter two themes are more general than the others, because

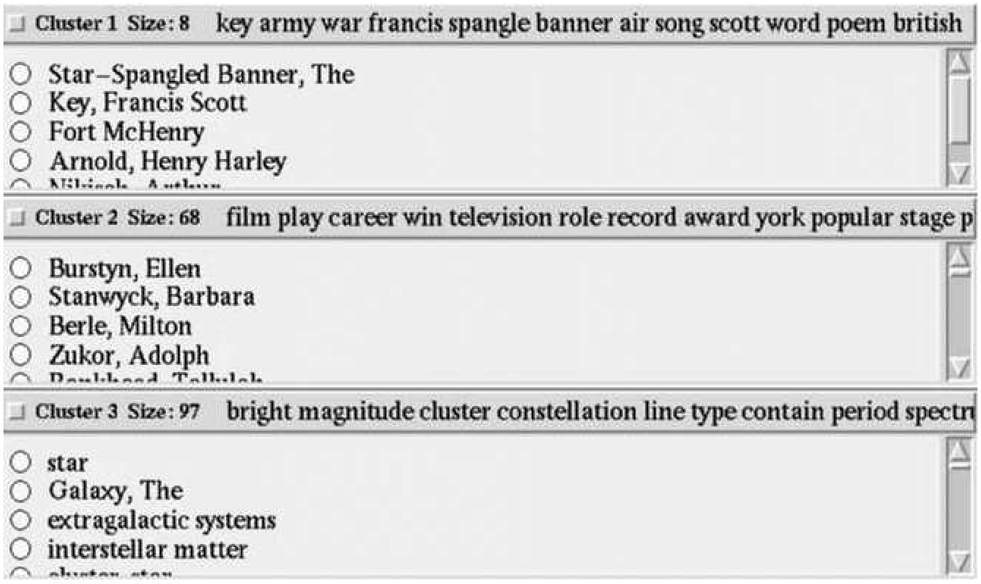

Figure 10.6: Display of Scatter/Gather clustering retrieval results (Cutting et al. 1992).

they are about topics outside the general scope of computer science. Thus clustering can results in the juxtaposition of very different levels of description within a single display.

Scatter/Gather shows a textual representation of document clusters. Researchers have developed several approaches to map documents from their high dimensional representation in document space into a 2D representation in which each document is represented as a small glyph or icon on a map or within an abstract 2D space. The functions for transforming the data into the lower dimensional space differ, but the net effect is that each document is placed at one point in a scatter plot-like representation of the space. Users are meant to detect themes or clusters in the arrangement of the glyphs. Systems employing such graphical displays include BEAD [33], the Galaxy of News [155], and ThemeScapes [188]. The ThemeScapes view imposes a three-dimensional representation on the results of clustering (see Figure 10.7). The layout makes use of "negative space" to help emphasize the areas of concentration where the clusters occur. Other systems , display inter-document similarity hierarchically[117, 3], while still others display retrieved documents in networks based on interdocument similarity [58, 175].

Kohonen's feature map algorithm has been used to create maps that graphically characterize the overall content of a document collection or subcollection [114, 35] (see Figure 10.8). The regions of the 2D map vary in size and shape corresponding to how frequently documents assigned to the corresponding themes occur within the collection. Regions are characterized by single words or phrases, and adjacency of regions is meant to reflect semantic relatedness of the themes within the collection. A cursor moved over a document region causes the titles of the documents most strongly associated with that region to be displayed in a pop-up window. Documents can be associated with more than one region.

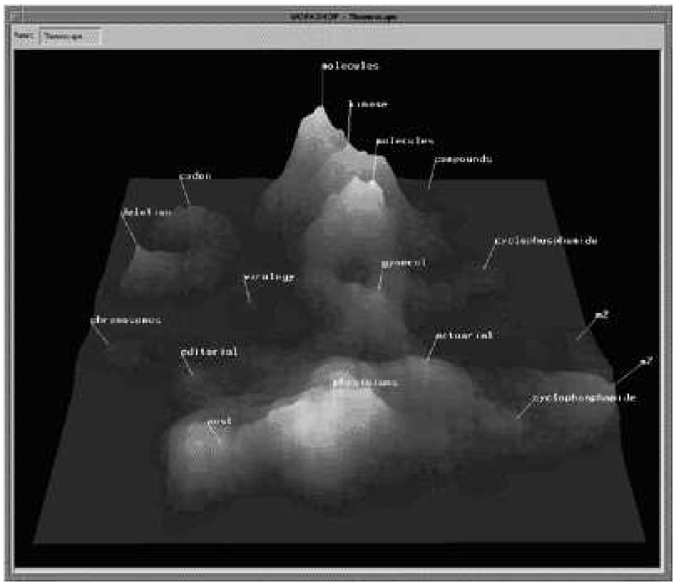

Figure 10.7: A three-dimensional overview based on document clustering (from Wise et al. 1995)

Evaluations of Graphical Overviews

Although intuitively appealing, graphical overviews of large document spaces have yet to be shown to be useful and understandable for users. In fact, evaluations that have been conducted so far provide negative evidence as to their usefulness. One study found that for non-expert users the results of clustering were difficult to use, and that graphical depictions (for example, representing clusters with circles and lines connecting documents) were much harder to use than textual representations (for example, showing titles and topical words, as in Scatter/Gather), because documents' contents are difficult to discern without actually reading some text [94].

Another recent study compared the Kohonen feature map overview representation on a browsing task to that of Yahoo [35]. For one of the tasks, subjects were asked to find an "interesting" web page within the entertainment category of Yahoo and of an organization of the same web pages into a Kohonen map layout. The experiment varied whether subjects started in Yahoo or in the graphical map. After completion of the browing task, subjects were asked to attempt to repeat the browse using the other tool. For the subjects that began with the Kohonen map visualization, 11 out of 15 found an interesting page within 10 minutes. Eight of these were able to find the same page using Yahoo. Of the subjects who started with Yahoo, 14 out of 16 were able to find interesting home pages. However, only 2 of the 14 were able to find the page in the graphical map display! This is strong evidence against the navigability of the display and certainly suggests that the simply label view provided by Yahoo is more useful. However, the map display may be more useful if the system is modified to tightly integrate querying with browsing.

The subjects did prefer some aspects of the map representation. In particular, some liked the ease of being able to jump from one area to another without having to back up as is required in

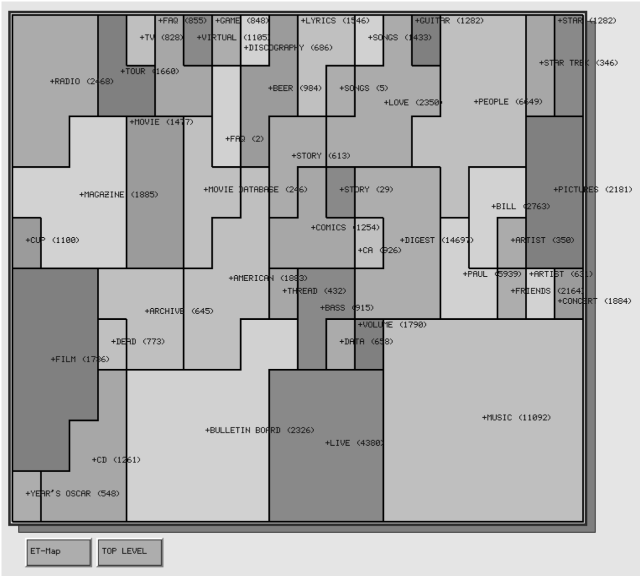

Figure 10.8: A two-dimensional overview created using a Kohonen feature map learning algorithm on web pages having to do with the topic Entertainment (from Chen et al. 1998)

Yahoo, and some liked the fact that the maps have varying levels of granularity. The subjects disliked several aspects of the display. The experimenters found that some subjects expressed a desire for a visible hierarchical organization, others wanted an ability to zoom in on a subarea to get more detail, and some users disliked having to look through the entire map to find a theme, desiring an alphabetical ordering instead. Many found the single-term labels to be misleading, in part because they were ambiguous (one region called "BILL" was thought to correspond to a person's name rather than counting money).

The authors concluded that this interface is more appropriate for casual browsing than for search. In general, unsupervised thematic overviews are perhaps most useful for giving users a "gist" of the kinds of information that can be found within the document collection, but generally have not been shown to be useful for use in the information access process.

Cocitation Clustering for Overviews

Citation analysis has long been recognized as a way to show an overview of the contents of a collection [184]. The main idea is to determine "centrally-located" documents based on cocitation patterns. There are different ways to determine citation patterns: one method is to measure how often two articles are cited together by a third. Another alternative is to pair articles that cite the same third article. In both cases the assumption is that the paired articles share some commonalities. After a matrix of cocitations is built, documents are clustered based on the similarity of their cocitation patterns. The resulting clusters are interpreted to indicate dominant themes within the collection. Clustering can focus on the authors of the documents rather than the contents, to attempt to identify central authors within a field. This idea has recently been implemented using web based documents in the Referral Web project [91]. This idea has also been applied to web pages, using web link structure to identify major topical themes among web pages [109, 146]. A similar idea, but computed a different way, is used to explicitly identify pages that act as good starting points for particular topics (called "authority pages" by Kleinberg [95]).

10.4.3 Examples, Dialogues, and Wizards

Another way to help users get started is to start them off with an example of interaction with the system. This technique is also known as **retrieval by reformulation**. An early version of this idea is embodied in the Rabbit system [187] which provides graphical representations of example database queries. A general framework for a query is shown to the user who then modifies it to construct a partially complete description of what they want. The system then shows an example of the kind of information available that matches this partial description. For instance, if a user searching a computer products database indicates an interest in disks, an example item is retrieved with its disk descriptors filled in. The user can use or modify the displayed descriptors, and iterate the procedure.

The idea of retrieval by reformulation has been developed further and extended to the domains of user interface development [134] and software engineering [154]. The Helgon system [56] is a modern variant of this idea applied to bibliographic database information. In Helgon, users begin by navigating a hierarchy of topics from which they select structured examples, according to their interests. If a feature of an example is inappropriately set, the user can modify the feature to indicate how it would appear in the desired information. Unfortunately, in tests with users, the system was found to be problematic. Users had problems with the organization of the hierarchy, and found that finding a useful example by critiquing an existing one to be tedious. This result underscores an unfortunate difficulty with examples and dialogues: that of getting the user to the right starting dialogue or the right example strategy becomes a search problem in itself. (How to index prior examples is studied extensively in the case-based reasoning (CBR) literature [110, 97].)

A more dynamic variation on this theme is the interactive dialogue. Dialogue-based interfaces have been explored since the early days of information retrieval research, in an attempt to mimic the interaction provided by a human search intermediary (e.g., a reference librarian). Oddy did early work in the THOMAS system, which provided a question and answer session within a command-line based interface [141]. More recently, Belkin, et al. have defined quite elaborate dialogue interaction models [18] although these have not been assessed empirically to date.

The DLITE system interface [40] uses an animated focus-plus-context dialogue as a way to acquaint users with standard sequences of operations within the system. Initially an outline of all of the steps of the dialogue is shown as a list. The user can expand the explanation of any individual step by clicking on its description. The user can expand out the entire dialogue to see what questions are coming next, and the collapse it again in order to focus on the current tactic.

A more restricted form of dialogue that has become widely used in commercial products is that of the **wizard**. This tool helps users in time-limited tasks, but does not attempt to overtly teach them about the processes required to complete the tasks. The wizard presents a step-by-step shortcut through the sequence of menu choices (or tactics) that a user would normally perform in order to get a job done, reducing user input to just a few choices with default settings [145]. A recent study [31] found wizards to be useful for goals that require many steps, for users who lack necessary domain knowledge (for example, a restaurant owner installing accounting software), and when steps must be completed in a fixed sequence (for example, a procedure for hiring personnel). Properties of successful wizards included allowing users to re-run a wizard and modify their previous work, show an overview of the functions they contain, and provide lucid descriptions and understandable outcomes for choices. Wizards were found not to be helpful when the interface did not solve a problem effectively (for example, a commercial wizard for setting up a desktop search index requests users to specify how large to make the index, but supplies no information about how to make this decision). Wizards were also found not to be helpful when the goal was to teach the user how to use the interface, and when the wizard was not user-tested. Most likely information access is too variable a process for the use of wizards.

A **guided tour** leads a user through a sequence of navigational choices through hypertext links, presenting the nodes in a logical order for some goal. In a dynamic tour, only relevant nodes are displayed, as opposed to the static case where the author decides what is relevant before the users have even formulated their queries [66]. A recent application is the Walden Paths project which enables teachers to define instructionally useful paths through pages found on the web [63]. This approach has not been commonly used to date for information access but could be a promising direction for acquainting users with search strategies in large hyperlinked systems.

10.4.4 Automated Source Selection

Human-computer interfaces for helping guide users to appropriate sources is a wide open area for research. It requires both eliciting the information need from the user and understanding which needs can be satisfied by which sources. An ambitious approach is to build a model of the source and of the information need of the user and try to determine which fit together best. User modeling systems and intelligent tutoring systems attempt to do this both for general domains [44, 185] and for online help systems [84].

A simpler alternative is to create a representation of the contents of information sources and match this representation against the query specification. This approach is taken by GLOSS, a system which tries to determine in advance the best bibliographic database to send a search request to, based on the terms in the query [176]. The system uses a simple analysis of the combined frequencies of the query words within the individual collections. The SavvySearch system [85] takes this idea a step further, using actions taken by users after a query to decide how to downweight or increase the ranking of a search engine for a particular query.

The flip side to automatically selecting the best source for a query is to automatically send a query to multiple sources and then combine the results from the various systems in some way. Many metasearch engines exist on the web. How to combine the results effectively is an active area of research, sometimes known as collection fusion [12, 177, 86].

10.5 Query Specification

To formulate a query, a user must select collections, metadata descriptions, or information sets against which the query is to be matched, and must specify words, phrases, descriptors, or other kinds of information that can be compared to or matched against the information in the collections. As a result, the system creates a set of documents, metadata, or other information type that match the query specification in some sense and displays the results to the user in some form.

Shneiderman [168] identifies five primary human-computer interaction styles. These are: **command language**, **form fillin**, **menu selection**, **direct manipulation**, and **natural language**.[4] Each technique has been used in query specification interfaces and each has advantages and disadvantages. These are described below in the context of Boolean query specification.

10.5.1 Boolean Queries

In modern information access systems the matching process usually employs a statistical ranking algorithm. However, until recently most commercial full-text systems and most bibliographic systems supported only Boolean queries. Thus the focus of many information access studies has been on the problems users have in specifying Boolean queries. Unfortunately, studies have shown time and again that most users have great difficulty specifying queries in Boolean format and often misjudge what the results will be [23, 65, 130, 191].

Boolean queries are problematic for several reasons. Foremost among these is that most people find the basic syntax counter-intuitive. Many English-speaking users assume everyday semantics are associated with Boolean operators when expressed using the English words AND and OR, rather than their logical equivalents. To inexperienced users, using AND implies the widening of the scope of the query, because more kinds of information are being requesting. For instance, "dogs and cats" may imply a request for documents about dogs and documents about cats, rather than documents about both topics at once. "Tea or coffee" can imply a mutually exclusive choice in everyday language. This kind of conceptual problem is well-documented [23, 65, 130, 191]. In addition, most query languages that incorporate Boolean operators also require the user to specify complex syntax for other kinds of connectors and for descriptive metadata. Most users are not familiar with the use of parentheses for nested evaluation, nor with the notions associated with operator precedence.

By serving a massive audience possessing little query-specification experience, the designers of World Wide Web search engines have had to come up with more intuitive approaches to query specification. Rather than forcing users to specify complex combinations of ANDs and ORs, they allow users to choose from a selection of common simple ways of combining query terms, including "all the words" (place all terms in a conjunction) and "any of the words" (place all terms in a disjunction).

Another web-based solution is to allow syntactically based query specification, but to provide a simpler or more intuitive syntax. The "+" prefix operator gained widespread use with the advent of its use as a mandatory specifier in the Altavista web search engine. Unfortunately, users can be

[4] This list omits non-visual modalities, such as audio.

misled to think it is an infix AND rather than a prefix mandatory operator, and thus assume that "cat + dog" will only retrieve articles containing both terms (where in fact this query requires dog but allows cat to be optional).

Another problem with pure Boolean systems is they do not rank the retrieved documents according to their degree of match to the query. In the pure Boolean framework a document either satisfies the query or it does not. Commercial systems usually resort to ordering documents according to some kind of descriptive metadata, usually in reverse chronological order. (Since these systems usually index timely data corresponding to newspaper and newswires, date of publication is often one of the most salient features of the document.) Web-based systems usually rank order the results of Boolean queries using statistical algorithms and web-specific heuristics.

10.5.2 From Command Lines to Forms and Menus

Aside from conceptual misunderstandings of the logical meaning of AND and OR, another part of the problem with pure Boolean query specification in online bibliographic systems is the arbitrariness of the syntax and the contextlessness nature of the TTY-based interface in which they are predominantly available. Typically input is typed at a prompt and is of a form something like the following:

> COMMAND ATTRIBUTE value {BOOLEAN-OPERATOR ATTRIBUTE value}*
> e.g.,
> FIND PA darwin AND TW species OR TW descent
> or
> FIND TW Mt St. Helens AND DATE 1981

(These examples are derived from the syntax of the telnet interface to the University of California Melvyl system [116].) The user must remember the commands and attribute names, which are easily forgotten between usages of the system [127]. Compounding this problem, despite the fact that the command languages for the two main online bibliographic systems at UC Berkeley have different but very similar syntaxes, after more than 10 years one of the systems still reports an error if the author field is specified as PA instead of PN, as is done in the other system. This lack of flexibility in the syntax is characteristic of interfaces designed to suit the system rather than its users.

The new Web-based version of Melvyl provides form fillin and menu selection so the user no longer has to remember the names and types of attributes available. Users select metadata types from listboxes and attributes are shown explicitly, allowing selection as an alternative to specification. For example, the "search type" field is adjacent to an entry form in which users can enter keywords, and a choice between AND and NOT is provided adjacent to a list of the available document types (editorial, feature, etc). Only the metadata associated with a given collection is shown in the context of search over that collection. (Unfortunately the system is restricted to searching over only one database at a time. It does however provide a mechanism for applying a previously executed search to a new database.) See Figure 10.9.

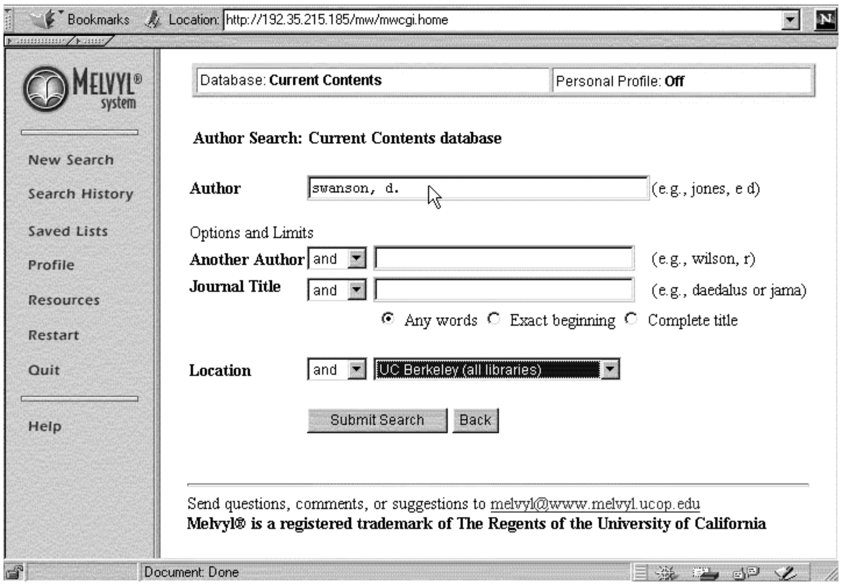

Figure 10.9: A view of query specification in the Web-based version of the Melvyl bibliographic catalog.

The Web-based version of Melvyl also allows retention of context between searches, storing prior results in tables and hyperlinking these results to lists containing the retrieved bibliographic information. Users can also modify any of the previously submitted queries by selecting a checkbox beside the record of the query. The graphical display makes explicit and immediate many of the powerful options of the system that most users would not learn the command-line version of the interface.

Bitmapped displays are an improvement over command-line interface, but do not solve all the problems. For example, a blank entry form is in some ways not much better than a TTY prompt, because it does not provide the user with clues about what kinds of terms should be entered.

10.5.3 Faceted Queries

Yet another problem with Boolean queries is that their strict interpretation tends to yield result sets that are either too large, because the user includes many terms in a disjunct, or are empty, because the user conjoins terms in an effort to reduce the result set. This problem occurs in large part because the user does not know the contents of the collection or the role of terms within the collection.

A common strategy for dealing with this problem, employed in systems with command-line based interfaces like DIALOG's, is to create a series of short queries, view the number of documents returned for each, and combine those queries that produce a reasonable number of results. For example, in DIALOG, each query produces a resulting set of documents that is assigned an identifying name. Rather than returning a list of titles themselves, DIALOG shows the set number with a listing of the number of matched documents. Titles can be shown by specifying the set number and issuing a command to show the titles. Document sets that are not empty can be referred to by a set name and combined with AND operations to produce new sets. If this set in turn is too small, the user can back up and try a different combination of sets, and this process is repeated in pursuit of producing a reasonably sized document set.

This kind of query formulation is often called a **faceted** query, to indicate that the users' query is divided into topics or facets, each of which should be present in the retrieved documents [127, 73]. For example, a query on drugs for the prevention of osteoporosis might consist of three facets, indicated by the disjuncts

(osteoporosis OR "bone loss")
(drugs OR pharmaceuticals)
(prevention OR cure).

This query implies that the user would like to view documents that contain all three topics.

A technique to impose an ordering on the results of Boolean queries is what is known as **post-coordinate** or **quorum-level** ranking [162, Ch. 8]. In this approach, documents are ranked according to the size of the subset of the query terms they contain. So given a query consisting of "cats", "dogs", "fish", and "mice", the system would rank a document with at least one instance of "cats", "dogs", and "fish" higher than a document containing 30 occurrences of "cats" but no occurrences of the other terms.

Combining faceted queries with quorum ranking yields a situation intermediate between full Boolean syntax and free-form natural language queries. An interface for specifying this kind of interaction can consist of a list of entry lines. The user enters one topic per entry line, where each topic consists of a list of semantically related terms that are combined in a disjunct. Documents that contain at least one term from each facet are ranked higher than documents containing terms only from one or a few facets. This helps ensure that documents which contain discussions of several of the users' topics are ranked higher than those that contain only one topic. By only requiring that one term from each facet be matched, the user can specify the same concept in several different ways in the hopes of increasing the likelihood of a match. If combined with graphical feedback about which subsets of terms matched the document, the user can see the results of a quorum ranking by topic rather than by word. Section 10.6 describes the TileBars interface which provides this type of feedback.

This idea can be extended yet another step by allowing users to weight each facet. More likely to be readily usable, however, is a default weighting in which the facet listed highest is assigned the most weight, the second facet is assigned less weight, and so on, according to some distribution over weights.

10.5.4 Graphical Approaches to Query Specification

Direct manipulation interfaces provide an alternative to command line syntax. The properties of direct manipulation are [168][p.205]: (1) continuous representation of the object of interest (2) physical actions or button presses instead of complex syntax, and (3) rapid incremental reversible operations whose impact on the object of interest is immediately visible. Direct manipulation interfaces often evoke enthusiasm from users, and for this reason alone it is worth exploring their use. Although they are not without drawbacks, they are easier to use than other methods for many users in many contexts.

Several variations of graphical interfaces, both directly manipulable and static, have been developed for simplifying the specification of Boolean syntax. User studies tend to reveal that these graphical interfaces are more effective in terms of accuracy and speed than command-language counterparts. Three such approaches are described below.

Graphical depictions of **Venn diagrams** have been proposed several times as a way to improve Boolean query specification. A query term is associated with a ring or circle and intersection of rings indicates conjunction of terms. Typically the number of documents that satisfy the various conjuncts are displayed within the appropriate segments of the diagram. Several studies have found such interfaces more effective than their command-language based syntax [90, 5, 130]. Hertzum and Frokjaer found that a simple Venn diagram representation produced faster and more accurate results than a Boolean query syntax. However, a problem with this format is the limitations on the complexity of the expression. For example, a maxium of three query terms can be ANDed together in a standard Venn diagram. Innovations have been designed to get around this problem, as seen in the VQuery system [90] (see Figure 10.10). In VQuery, a direct manipulation interface allows users to assign any number of query terms to ovals. If two or more ovals are placed such that they overlap with one another, and if the user selects the area of their intersection, an AND is implied among those terms. (In Figure 10.10, the term "Query" is conjoined with "Boolean"). If the user selects outside the area of intersection but within the ovals, and OR is implied among the corresponding terms. A NOT operation is associated with any term whose oval appears in the active area of the display but which remains unselected (in the figure, NOT "Ranking" has been specified). An active area indicates the current query; ovals containing terms can be moved out of the active area for later use.

Young and Shneiderman [191] found improvements over standard Boolean syntax by providing users with a direct manipulation **filter-flow** model. The user is shown a scrollable list of attribute types on the lefthand side and selects attributes from another list of attribute types shown across the top of the screen. Clicking on an attribute name causes a listbox containing values for those attributes to be displayed in the main portion of the screen. The user then selects which values of the attributes to let the flow go through. Placing two or more of these attributes in sequence creates the semantics of a conjunct over the selected values. Placing two or more of these in parallel creates the semantics of a disjunct. The number of documents that match the query at each point is indicated by the width of the "water" flowing from one attribute to the next. (See Figure 10.11.) A conjunct can reduce the amount of flow. The items that match the full query are shown on the far righthand side. A user study found that fewer errors were made using the filter flow model than a standard SQL database query. However, the examples and study pertain only to database querying rather than information access, since the possible query terms for information access cannot be represented realistically in a scrollable list. It should be feasible to display a selectable list of terms conceptually related to those already chosen.

Anick et al. [6] describe another innovative direct manipulation interface for Boolean queries.

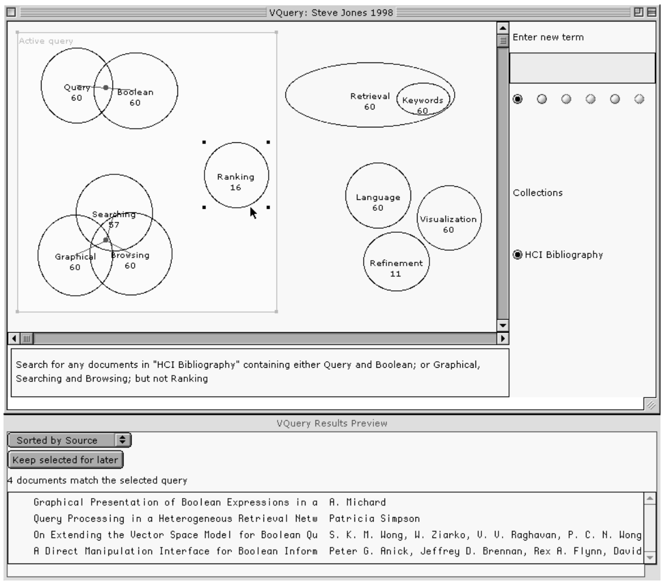

Figure 10.10: The VQuery Venn-diagram visualization for Boolean query specification (Jones 1998).

Initially the user types a natural language query which is automatically converted to a representation in which each query term is represented by a block. The blocks are arranged into rows and columns. (See Figure 10.12.) If two or more blocks appear along the same row they are considered to be ANDed together. Two or more blocks within the same column are ORed. Thus the user can represent a technical term in multiple ways within the same query, providing a kind of faceted query interface. For example, the terms "version 5", "version 5.0", and "v5" might be shown in the same row. Blocks can also be activated and deactivated so users can quickly experiment with different combinations of terms and Boolean queries simply by activating and deactivating blocks. This facility also allows users to have multiple representations of the same term in different places throughout the display, thus allowing rapid feedback on the consequences of specifying various combinations of query terms. Informal evaluation of the system found that users were able to learn to manipulate the interface quickly and enjoyed using it. It was not formally compared to other interaction techniques [6].

This interface provides a kind of **query preview**: a low-cost, rapid-turnaround visualization of the results of many variations on a query [148]. Another example of query previewing can be found in some help systems, which show all the words in the index whose first letters match the characters that the user has typed so far. The more characters typed, the fewer possible matches become available. The HiBrowse system described above [150] also provides a kind of preview for viewing category hierarchies and facets, showing how many documents would be matched if a category one level below the current one were selected. It perhaps could be improved by showing the consequences of more combinations of categories in an animated manner. If based on prior action and interests of the user, query previewing may become more generally applicable for information access interfaces.

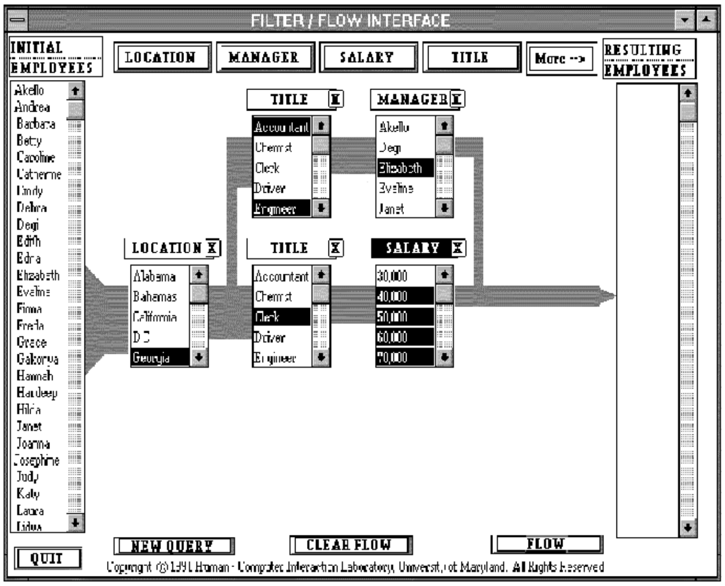

Figure 10.11: The filter-flow visualization for Boolean query specification (from Young and Shneiderman 1993).

Figure 10.12: A block-oriented diagram visualization for Boolean query specification (from Anick et al. 1990).

A final example of a graphical approach to query specification is the use of magic lenses. Fishkin and Stone have suggested an extension to the usage of this visualization tool for the specification of Boolean queries [57]. Information is represented as lists or icons within a 2D space. Lenses act as filters on the document set. (See Figure 10.13.) For example, a word can be associated with a transparent lens. When this lens is placed over an iconic representation of a set of documents, it can cause all documents that do not contain a given word to disappear. If a second lens representing another word is then laid over the first, the lenses combine to act as a conjunction of the two words with the document set, hiding any documents that do not contain both words. Additional information can be adjusted dynamically, such as a minimum threshold for how often the term occurs in the documents, or an on-off switch for word stemming. This approach, while promising, has not been evaluated in an information access setting.

Figure 10.13: A magic lens interface for query specification (from Fishkin and Stone 1995).

10.5.5 Phrases and Proximity

In general, proximity information can be quite effective at improving precision of searches. On the Web, the difference between a single-word query and a two-word exact phrase match can mean the difference between an unmanageable mess of retrieved documents and a short list with mainly relevant documents.

A large number of methods for specifying phrases have been developed. The syntax in Lexis-Nexis requires the proximity range to be specified with an infix operator. For example, "white w/3 house" means "white within 3 words of house, independent of order". Exact proximity of phrases is specified by simply listing one word beside the other, separated by a space. A popular method used by web search engines is the enclosure of the terms between quotation marks. Shneiderman et al [169] suggest providing a list of entry labels, as suggested above for specifying facets. The difference is, instead of a disjunction, the terms on each line are treated as a phrase. This is suggested as a way to guide users to more precise query specification.

The disadvantage of these methods is they require exact match of phrases, when it is often the case (in English) that one or a few words comes between the terms of interest. For example, in most cases the user probably wants "president" and "lincoln" to be adjacent, but still wants to catch cases of the sort "President Abraham Lincoln". Another consideration is whether or not stemming is performed on the terms included in the phrase. The best solution may be to allow users to specify exact phrases but treat them as small proximity ranges, with perhaps an exponential fall-off in weight in terms of distance of the terms. This has been shown to be a successful strategy in non-interactive ranking algorithms [37]. It has also been shown that a combination of quorum ranking of faceted queries with the restriction that the facets occur within a small proximity range can dramatically improve precision of results [76, 132].

10.5.6 Natural Language and Free Text Queries

Statistical ranking algorithms have the advantage of allowing users to specify queries naturally, without having to think about Boolean or other operators. But they have the drawback of giving the user less feedback about and control over the results. Usually the result of a statistical ranking is the listing of documents and the association of a score, probability, or percentage beside the title. Users are given little feedback about why the document received the ranking it did and what the roles of the query terms are. This can be especially problematic if the user is particularly interested in one of the query terms' being present.

One search strategy that can help with this particular problem with statistical ranking algorithms is the specification of "mandatory" terms within the natural language query. This in effect helps the user control which terms are considered important, rather than relying on the ranking algorithm to correctly weight the query terms. But knowing to include a mandatory specification requires the user to know about a particular command and how it works.

The preceding discussion assumes that a natural language query entered by the user is treated as a bag of words, with stop words removed, for the purposes of document match. However, some systems attempt to parse natural language queries in order to extract concepts to match against concepts in the text collection [87, 126, 172].

Alternatively the natural language syntax of a question can be used to attempt to answer the question. (Question answering in information access is different than that of database management systems, since the information desired is encoded within the text of documents rather than specified by the database schema.) The Murax system [102] determines from the syntax of a question if the user is asking for a person, place, or date. It then attempts to find sentences within encyclopedia articles that contain noun phrases that appear in the question, since these sentences are likely to contain the answer to the question. For example, given the question "Who was the Pulitzer Prize-winning novelist that ran for mayor of New York City?", the system extracts the noun phrases "Pulitzer Prize", "winning novelist", "mayor", and "New York City". It then looks for proper nouns representing people's names (since this is a "who" question) and finds, among others, the following sentences:

> The Armies of the Night (1968), a personal narrative of the 1967 peace march on the Pentagon, won **Mailer** the **Pulitzer Prize** and the National Book Award.
>
> In 1969 **Mailer** ran unsuccessfully as an independent candidate for **mayor** of **New York City**.

Thus the two sentences link together the relevant noun phrases and the system hypothesizes (correctly) from the title of the article in which the sentences appear that Norman Mailer is the answer.

Another approach to automated question-answering is the FAQ finder system which matches question-style queries against question-answer pairs on various topics [25]. The system uses a standard IR search to find the most likely FAQ (frequently asked questions) files for the question and then matches the terms in the question against the question portion of the question-answer pairs.

A less automated approach to question answering can be found in the Ask Jeeves system[5]. This system makes use of hand-picked web sites, matches these to a pre-defined set of question types. A user's query is first matched against the question types. The user selects the most accurate rephrase of their question and this in turn is linked to suggested web sites. For example, the question "Who is the leader of Sudan?" is mapped into the question type "Who is the head of state of X (Sudan)?" where the variable is replaced by a listbox of choices, with Sudan the selected choice in this case. This is linked to a web page that lists current heads of state. The system also automatically substitutes in the name "Sudan" in a query against that web page, thus bringing the answer directly to the users' attention. The system is only as good as its question templates. For example a question "Where can I find reviews of spas in Calistoga?" matches the question "Where can I find X (reviews) of activities for children aged Y (1)?" and "Where can I find a concise encyclopedia article on X (hot springs)?" The question is also sent to standard web search engines.

10.6 Context

This section discusses interface techniques for placing the current document set in **context** of other information types, in order to make the document set more understandable. This includes showing the relationship of the document set to query terms, collection overviews, descriptive metadata, hyperlink structure, document structure, and to other documents within the set.

[5]http://www.askjeeves.com

10.6.1 Document Surrogates

The most common way to show results for a query is to list information about documents in order of their computed relevance to the query. Alternatively, for pure Boolean ranking, documents are listed according to a metadata attribute, such as date. Typically the document list consists of the document's title and a subset of important metadata, such as date, source, and length of the article. In systems with statistical ranking, a numerical score or percentage is also often shown alongside the title, where the score indicates a computed degree of match or probability of relevance. This kind of information is sometimes referred to as a **document surrogate**. See Figure 10.14 from [189].

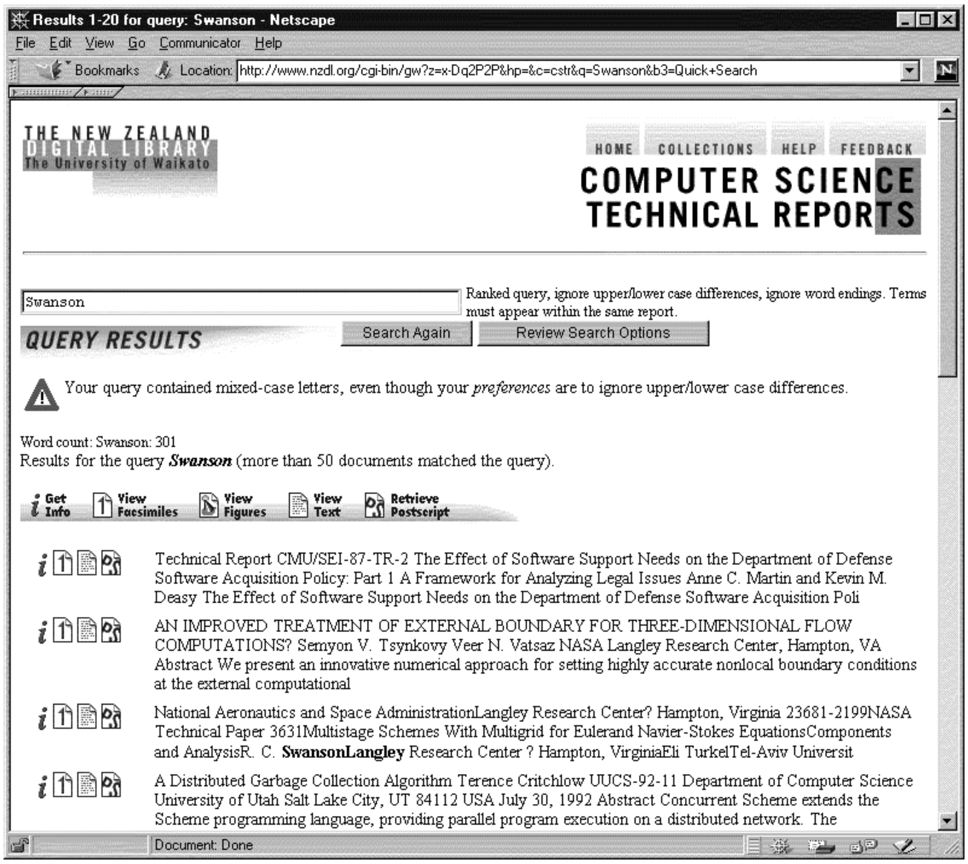

Figure 10.14: An example of a ranked list of titles and other document surrogate information (from Witten et al. 1998).

Some systems provide users with a choice between a short and a detailed view. The detailed view typically contains a summary or abstract. In bibliographic systems, the author-written or service-written abstract is shown. Web search engines automatically generate excerpts, usually extracting first few lines of non-markup text in the web page.

In most interfaces, clicking on the document's title or an iconic representation of the document shown beside the title will bring up a view of the document itself, either in another window on the screen, or replacing the listing of search results. (In traditional bibliographic systems, the full text was unavailable online, and only bibliographic records could be readily viewed.)

10.6.2 Query Term Hits Within Document Content

In systems in which the user can view the full text of a retrieved document, it is often useful to **highlight** the occurrences of the terms or descriptors that match those of the user's query. It can also be useful for the system to scroll the view of the document to the first passage that contains one or more of the query terms, and highlight the matched terms in a contrasting color or reverse video. This display is thought to help draw the user's attention to the parts of the document most likely to be relevant to the query. Highlighting of query terms has been found time and again to be a useful feature for information access interfaces [107],[123][p.31]. Color highlighting has also recently been found to be useful for scanning lists of bibliographic records [10].

KWIC

A facility related to highlighting is the **keyword-in-context** (KWIC) document surrogate. Sentence fragments, full sentences, or groups of sentences that contain query terms are extracted from the full text and presented for viewing along with other kinds of surrogate information (such as document title and abstract). Note that a KWIC listing is different than an abstract. An abstract summarizes the main topics of the document but might not contain references to the terms within the query. A KWIC extract shows sentences that summarize the ways the query terms are used within the document. This display can show not only which subsets of query terms occur in the retrieved documents, but also the context they appear in with respect to one another.

Tradeoff decisions must be made between how many lines of text to show and which lines to display. It is not known which contexts are best selected for viewing but results from text summarization research suggest that the best fragments to show are those that appear near the beginning of the document and that contain the largest subset of query terms [103]. If users have specified which terms are more important than others, then those fragments containing important terms should be shown before of those that contain only less-important terms. However, to help retain coherence of the excerpts, selected sentences should be shown in order of their occurrence in the original document, independent of how many search terms they contain.

The KWIC facility is usually not shown in web search result display, most likely because the system must have a copy of the original document available from which to extract the sentences containing the search terms. Web search engines typically only retain the index without term position information. Systems that index individual web sites can show KWIC of information in the document list display.

TileBars

A more compact form of query term hit display is made available through the TileBars interface. The user enters a query in a faceted format, with one topic per line. After the system retrieves documents (using a quorum or statistical ranking algorithm), a graphical bar is displayed next to the title of each document showing the degree of match for each facet. TileBars thus illustrate at a glance which passages in each article contain which topics – and moreover, how frequently each topic is mentioned (darker squares represent more frequent matches).

Each document is represented by a rectangular bar. Figure 10.15 shows an example. The bar is subdivided into rows that correspond to the query facets. The top row of each TileBar corresponds to "osteoporosis", the second row to "prevention", and the third row to "research". The bar is also subdivided into columns, where each column refers to a passage within the document. Hits that overlap within the same passage are more likely to indicate a relevant document than hits that are widely dispersed throughout the document [76]. The patterns are meant to indicate whether terms from a facet occur as a main topic throughout the document, as a subtopic, or are just mentioned in passing.

The darkness of each square corresponds to the number of times the query occurs in that segment of text; the darker the square the greater the number of hits. White indicates no hits on the query term. Thus the user can quickly see if some subset of the terms overlap in the same segment of the document. (The segments for this version of the interface are fixed blocks of 100 tokens each.)

The first document can be seen to have considerable overlap among the topics of interest towards the middle, but not at the beginning or the end (the actual end is cut off). Thus it most likely discusses topics in addition to research into osteoporosis. The second through fourth documents, which are considerably shorter, also have overlap among all terms of interest, and so are also probably of interest to the user. (The titles help to verify this.) The next three documents are all long, and from the TileBars we can tell they discuss research and prevention, but do not even touch on osteoporosis, and so probably are not of interest.

Because the TileBars interface allows the user to specify the query in terms of facets, where the terms for each facet are listed on an entry line, a color can be assigned to each facet. When the user displays a document with query term hits, the user can quickly ascertain what proportion of search topics appear in a passage based only on how many different highlight colors are visible. Most systems that use highlighting use only a single color to bring attention to all of the search terms.

It would be difficult for users to specify in advance which patterns of term hits they are interested. Instead, TileBars allows users to scan graphic representations and recognize which documents are and are not of interest. It may be the case the TileBars may be most useful for helping users discard misleadingly interesting documents, but only preliminary studies have been conducted to date. Passages can correspond to paragraphs or sections, fixed sized units of arbitrary length, or to automatically determined multi-paragraph segments [75].

SeeSoft

The SeeSoft visualization [53] represents text in a manner resembling columns of newspaper text, with one "line" of text on each horizontal line of the strip. (See Figure 10.16.) The representation is compact and aesthetically pleasing. Graphics are used to abstract away the details, providing an overview showing the amount and shape of the text. Color highlighting is used to pick out various

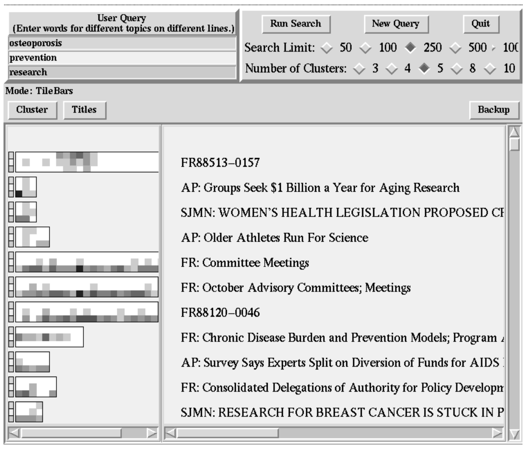

Figure 10.15: An example of the TileBars retrieval results visualization (Hearst 1995).

attributes, such as where a particular word appears in the text. Details of a smaller portion of the display can be viewed via a pop-up window; the overview shows more of the text but in less detail.

SeeSoft was originally designed for software development, in which a line of text is a meaningful unit of information. (Programmers tend to place each individual programming statement on one line of text.) Thus SeeSoft shows attributes relevant to the programming domain, such as which lines of code were modified by which programmer, and how often particular lines have been modified, and how many days have elapsed since the lines were last modified. The SeeSoft developers then experimented with applying this idea to the display of text, although this has not been integrated into an information access system. Color highlighting is used to show which characters appear where in a book of fiction, and which passages of the Bible contain references to particular people and items. Note the use of the abstraction an entire line to stand for a single word such as a character's name (even though though this might obscure a tightly interwoven conversation between two characters).

10.6.3 Query Term Hits Between Documents

Other visualization ideas have been developed to show a different kind of information about the relationship between query terms and retrieved documents. Rather than showing how query terms appear within individual documents, as is done in KWIC interfaces and TileBars, these systems display an overview or summary of the retrieved documents according to which subset of query terms they contain. The next subsections describe variations on this idea.

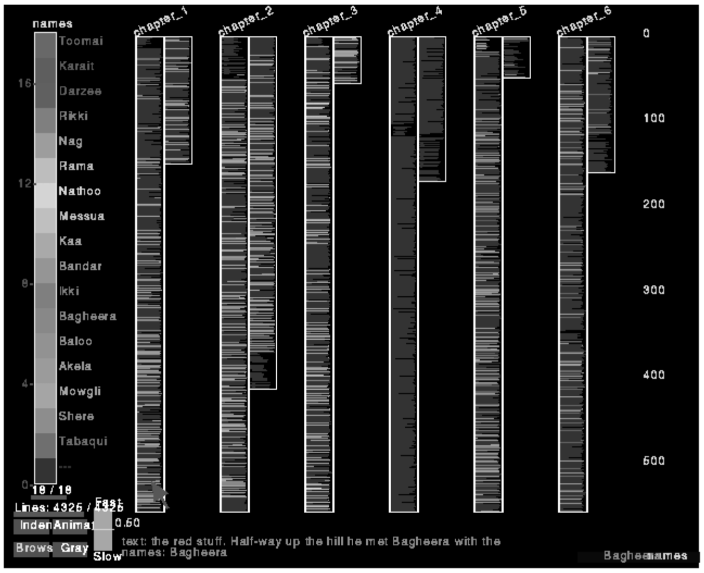

Figure 10.16: An example of the SeeSoft visualization for showing locations of characters within a text (from Eick 1994).

InfoCrystal

The InfoCrystal shows how many documents contain each subset of query terms [170]. This relieves the user from the need to specify Boolean ANDs and ORs in their query, while still showing which combinations of terms actually appear in documents that were ordered by a statistical ranking (although beyond four terms the interface becomes difficult to understand). The InfoCrystal allows visualization of all possible relations among N user-specified "concepts" (or Boolean keywords). The InfoCrystal displays, in a clever extension of the Venn-diagram paradigm, the number of documents retrieved that have each possible subset of the N concepts. Figure 10.17 shows a sketch of what the InfoCrystal might display as the result of a query against four keywords or Boolean phrases, labeled A, B, C, and D. The diamond in the center indicates that one document was discovered that contains all four keywords. The triangle marked with "12" indicates that twelve documents were found containing attributes A, B, and D, and so on.

The InfoCrystal does not show proximity among the terms within the documents, nor their relative frequency. So a document that contains dozens of hits on "volcano" and "lava" and one hit on "Mars" will be grouped with documents that contain mainly hits on "Mars" but just one mention each of "volcano" and "lava".

Figure 10.17: An example of the InfoCrystal retrieval results display (from Spoerri 1993).

VIBE and Lyberworld

Graphical presentations that operate on similar principles are VIBE [98] and Lyberworld [80]. In these displays, query terms are placed in an abstract graphical space. After the search, icons are created that indicate how many documents contain each subset of query terms. The subset status of each group of documents is indicated by the placement of the icon. For example, in VIBE a set of documents that contain three out of five query terms are shown on axis connecting these three terms, at a point midway between the representations of the three query terms in question. (See Figure 10.18.) Lyberworld presents a 3D version of this idea.

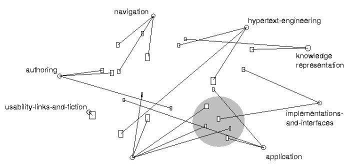

Figure 10.18: An example of the VIBE retrieval results display (from Korfhage 91).

Lattices

Several researchers have employed a graphical depiction of a mathematical lattice for the purposes of query formulation, where the query consists of a set of constraints on a hierarchy of categories (actually, semantic attributes in these systems) [144, 32]. This is one solution to the problem of displaying documents in terms of multiple attributes; a document containing terms A, B, C, and D could be placed at a point in the lattice with these four categories as parents. However, if such a representation were to be applied to retrieval results instead of query formulation, the lattice layout would in most cases be too complex to allow for readability.

None of the displays discussed in this subsection have been evaluated for effectiveness at improving query specification or understanding of retrieval results, but they are intriguing ideas and perhaps are useful in conjunction with other displays.

10.6.4 SuperBook: Context via Table of Contents

The SuperBook system [107, 51, 52] makes use of the structure of a large document to display query term hits in context. The table of contents (TOC) for a book or manual are shown in a hierarchy on the lefthand side of the display, and full text of a page or section is shown on the righthand side. The user can manipulate the table of contents to expand or contract the view of sections and subsections. A focus-plus-context mechanism is used to expand the viewing area of the sections currently being looked at and compress the remaining sections. When the user moves the cursor to another part of the TOC, the display changes dynamically, making the new focus larger and shrinking down the previously observed sections.

After the user specifies a query on the book, the search results are shown in the context of the table of contents hierarchy. (See Figure 10.19.) Those sections that contain search hits are made larger and the others are compressed. The query terms that appear in chapter or section names are highlighted in reverse video. When the user selects a page from the table of contents view, the page itself is displayed in the righthand side and the query terms within the page are highlighted in reverse video.

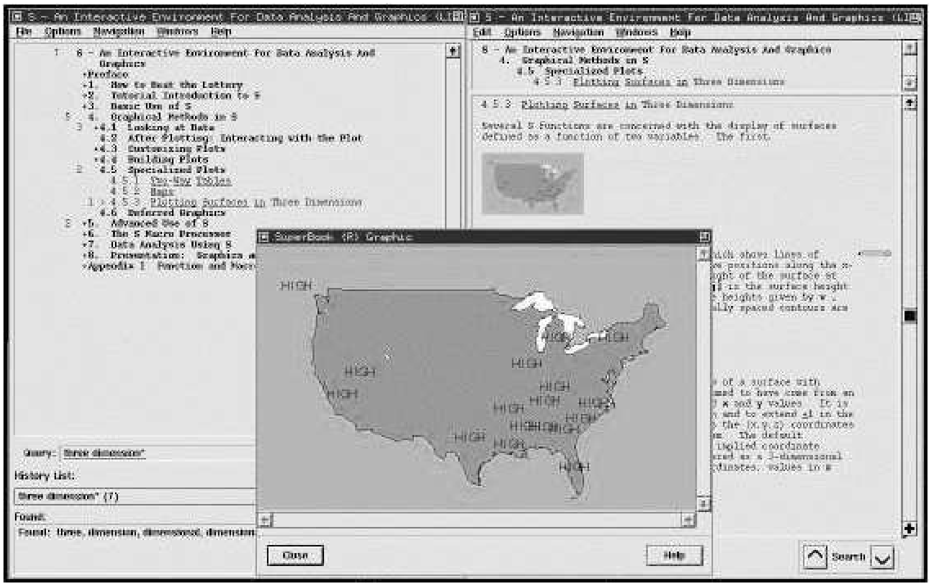

Figure 10.19: The SuperBook interface for showing retrieval results on a large manual in context (Landauer et al. 1993)

The SuperBook designers created innovative techniques for evaluating its special features. Subjects were compared using this system against using paper documentation and against a more standard online information access system. Subjects were also compared on different kinds of carefully selected tasks: browsing topics of interest, citation searching, searching to answer questions, and searching and browsing to write summary essays. For most of the tasks SuperBook subjects were faster and more accurate or equivalent in speed and accuracy to a standard system.

When differences arose between SuperBook and the standard system, the investigators examined the logs carefully and hypothesized plausible explanations. After the initial studies, they modified SuperBook according to these hypotheses and usually saw improvements as a result [107].

The user studies on the improved system showed that users were faster and more accurate at answering questions in which some of the relevant terms were within the section titles themselves, but they were also faster and more accurate at answering questions in which the query terms fell within the full text of the document only, as compared both to a paper manual and to an interface that did not provide such contextualizing information. SuperBook was not faster than paper when the query terms did not appear in the document text or the table of contents. This and other evidence from the SuperBook studies suggest that query term highlighting is at least partially responsible for improvements seen in the system.

10.6.5 Categories for Results Set Context

In Section 10.4 we saw the use of category or directory information for providing overviews of text collection content. Category metadata can also be used to place the results of a query in context.

For example, the original formulation of SuperBook allowed navigation within a highly structured document, a computer manual. The CORE project extended the main idea to a collection of over 1,000 full-text chemistry articles. A study of this representation demonstrated its superiority to a standard search system on a variety of task types [50]. Since a table of contents is not available for this collection, context is provided by placing documents within a category hierarchy containing terms relevant to chemistry. Documents assigned a category are listed when that category is selected for more detailed viewing, and the categories themselves are organized into a hierarchy, thus providing a hierarchical view on the collection.

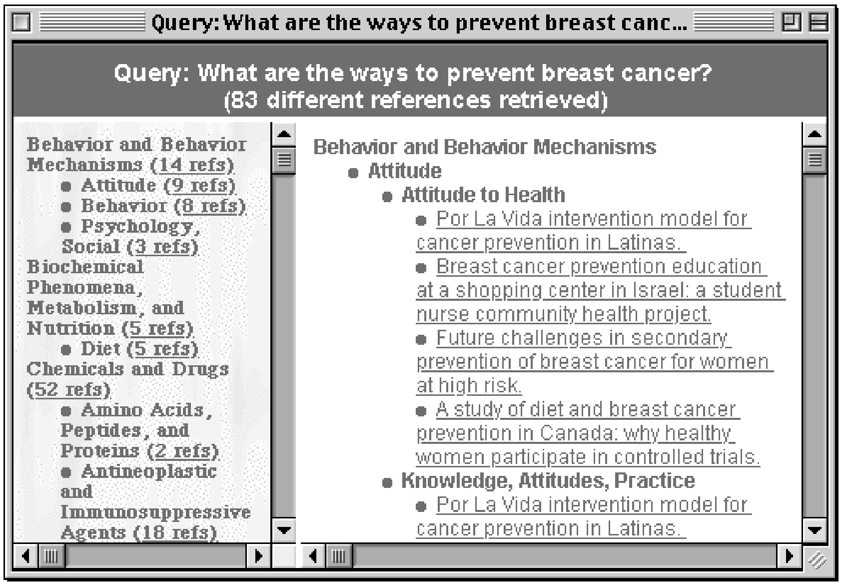

Figure 10.20: The DynaCat interface for viewing category labels that correspond to (Pratt 1997).

Another approach to using pre-defined categories to provide context for retrieval results is demonstrated by the DynaCat system [151]. The DynaCat system organizes retrieved documents according to which types of categories, selected from the large MeSH taxonomy, are known in advance to be important for a given query type. DynaCat begins with a set of query types known to be useful for a given user population and collection. One query type can encompass many different queries. For example, the query type "Treatment-Adverse Effects" covers queries such as "What are the complications of a mastectomy?" as well as "What are the side-effects of aspirin?" Documents are then organized according to a set of criteria associated with each query type which specify the types of categories that are acceptable to use for organizing the documents. Once categories have been assigned to the retrieved documents, a hierarchy is formed based on where the categories exist within MeSH. The algorithm selects only a subset of the category labels that might be assigned to the document to be used in the organization.

Figure 10.20 shows the results for a query on breast cancer prevention. The interface is tiled into three windows. The top window displays the user's query and the number of documents found. The left window shows the categories in the first two levels of the hierarchy, providing a table-of-contents view of the organization of search results. The right pane displays all the categories in the hierarchy and the titles of the documents that belong in those categories.

10.6.6 Using Hyperlinks to Organize Retrieval Results

Although the SuperBook authors describe it as a hypertext system, it is actually better thought of as a means of showing search results in the context of a structure that users can understand and view all at once. The hypertext component was not analyzed separately to assess its importance, but it usually is not mentioned by the authors when describing what is successful about their design. In fact, it seems to be responsible for one of the main problems seen with the revised version of the system – that users tend to wander off (often unintentionally) from the pages they are reading, thus causing the time spent on a given topic to be longer for SuperBook in some cases. (Using completion time to evaluate users on browsing tasks can be problematic, however, since by definition browsing is a casual, unhurried process [183]).

This wandering may occur in part because SuperBook uses a non-standard kind of hypertext, in which any word is automatically linked to occurrences of the same word in other parts of the document. This has not turned out to be how hypertext links are created in practice. Today, hyperlinked help systems and hyperlinks on the Web make much more discriminating use of hyperlink connections (in part since they are usually generated by an author rather than automatically). These links tend to be labeled in a somewhat meaningful manner by their surrounding context. Back-of-the-book indexes often do not contain listings of every occurrence of a word, but rather to the more important uses or the beginnings of series of uses. Automated hypertext linking should perhaps be based on similar principles. Additionally, at least one study showed that users formed better mental models of a small hypertext system that was organized hierarchically than one that allowed more flexible access [48]. Problems relating to navigation of hypertext structure have long been suspected and investigated in the hypertext literature [38, 125, 93, 67].

More recent work has made better use of hyperlink information for providing context for retrieval results. Some of this work is described below.

Cha-Cha: SuperBook on the Web

The Cha-Cha intranet search system [36] extends the SuperBook idea to a large heterogeneous web site such as might be found in an organization's Intranet. Figure 10.21 shows an example. This system differs from SuperBook in several ways. On most web sites there is no existing real table of contents or category structure, and an Intranet like those found at large universities or large corporations are usually not organized by one central unit. Cha-Cha uses link structure present within the site to create what is intended to be a meaningful organization on top of the underlying chaos. After the user issues a query, the shortest paths from the root page to each of the search hits are recorded and a subset of these are selected to be shown as a hierarchy, so that each hit is shown only once. (Users can begin with a query, rather than with a table of contents view.) If a user does not know to use the term "health center" but instead queries on "medical center," if "medical" appears as a term in a document within the health center part of the web, the home page (or starting point) of this center will be presented as well as the more specific hits. Users can then either query or navigate within a subset of sites if they wish. The organization produced by this simple method is surprisingly comprehensible on the UC Berkeley site. It seems especially useful for providing the information about the sources (the web server) associated with the search hits, whose titles are often cryptic.

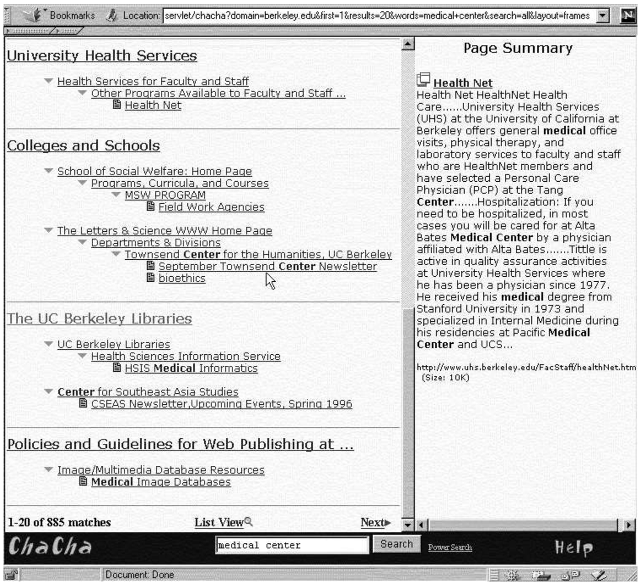

Figure 10.21: The Cha-Cha interface for showing web intranet search results in context (Chen and Hearst 1998).

The AMIT system [190] also applies the basic ideas behind SuperBook to the web, but focuses on a single-topic web site, which is likely to have a reasonable topic structure than a complex intranet. The link structure of the web site is used as contextualizing information but all of the paths to a given document are shown and focus-plus-context is used to emphasize subsets of the document space. The WebTOC system [137] is similar to AMIT but focuses on showing the structure and number of documents within each web subhierarchy, and is not tightly coupled with search.

WebCutter: Graphical Depiction of Link Structure

The WebCutter system [118] allows the user to issue a query on a particular web site. The system crawls the site in real time, checking each encountered page for relevance to the query. When a relevant page is found, the weights on that page's outlinks are increased. Thus, the search is based partly on an assumption that relevant pages will occur near one another in the web site. The subset of the web site that has been crawled is depicted graphically in a nodes-and-links view (see Figure 10.22). This kind of display does not provide the user with information about what the *contents* of the pages are, but rather only shows their link structure. Other researchers have also investigated spreading activation among hypertext links as a way to guide an information retrieval system, e.g., [61, 128].

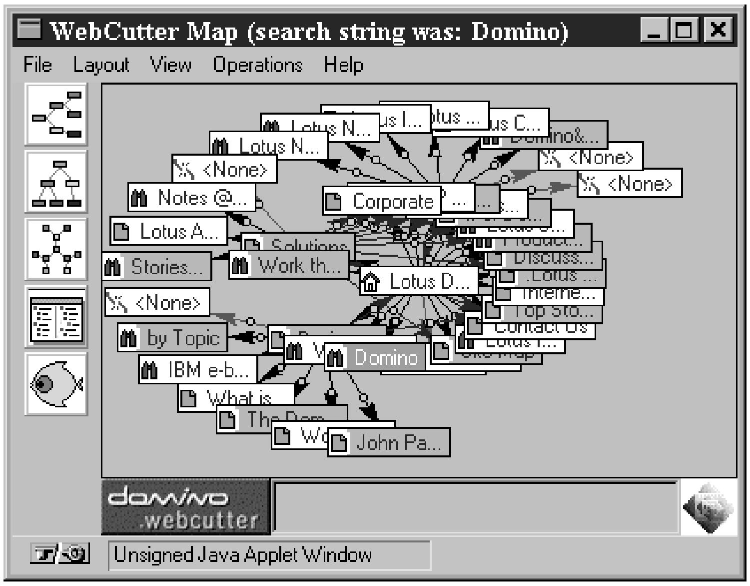

Figure 10.22: The WebCutter view of retrieval results (from Maarek and Ben Shaul 1997).

10.6.7 Tables

Tabular display is another approach for showing relationships among retrieval documents. The Envision system [59] allows the user to organize results according to metadata such as author or date along the X and Y-axes, and uses graphics to show values for attributes associated with retrieved documents within each cell (see Figure 10.23. Color, shape, and size of an iconic representation of a document are used to show the computed relevance, the type of document, or other attributes. Clicking on an icon brings up more information about the document in another window. Like the

WebCutter system, this view provides few cues about how the documents are related to one another in terms of their content or meaning. The SenseMaker system also allows users to group documents into different views via a table-like display [9], including a Scatter/Gather [43] style view. Although tables are appealing, they cannot show the intersections of many different attributes; rather they are better for pairwise comparisons. Another problem with tables for display of textual information is that very little information can be fit on a screen at a time, making comparisons difficult.

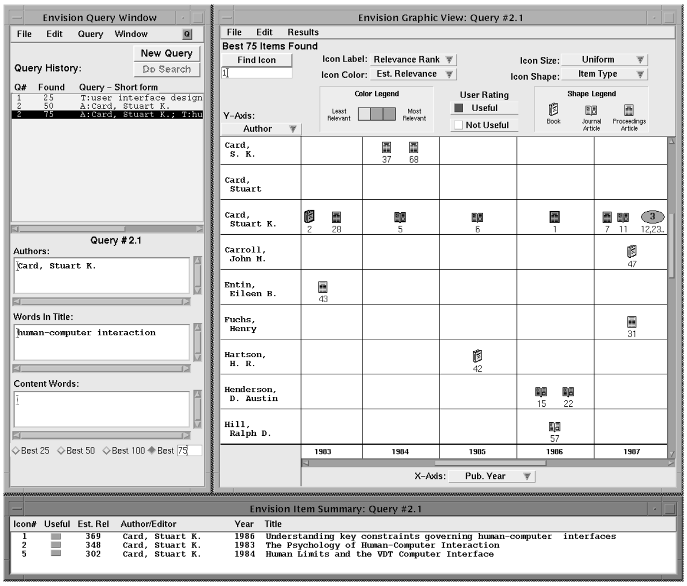

Figure 10.23: The Envision tabular display for graphically organizing retrieved documents (from Fox et al. 1993).

The Table Lens [152] is an innovative interface for viewing and interactively re-organizing very large tables of information. It uses focus plus context to fit hundreds of rows of information in a space occupied by at most two dozen rows in standard spreadsheets. And because it allows for rapid re-organization via sorting of columns, users can quickly switch from a view focused around one kind of metadata to another. For example, first sorting documents by rank and then by author name can show the relative ranks of different articles by the same author. A re-sort by date can show patterns in relevance scores with respect to date of publication. This rapid re-sorting capability helps circumvent the problems associated with the fact that tables cannot show many simultaneous intersections.

Another variation on the table theme is that seen in the Perspective Wall [119] in which a focus-plus-context display is used to center information currently of interest in the middle of the display, compressing less important information into the periphery on the sides of the wall. The idea is to show in detail the currently most important information while at the same time retaining the context of the rest of the information. For example, if viewing documents in chronological order, the user can easily tell if they are currently looking at documents in the beginning, middle, or end of the time range.

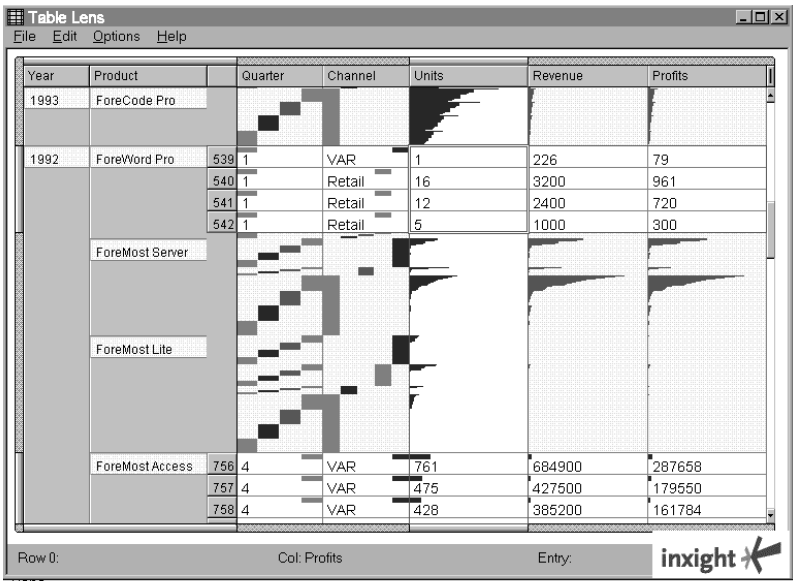

Figure 10.24: The TableLens visualization (from Rao and Card 1994).

These interfaces have not been applied to information access tasks. The problem with such displays when applied to text is that they require an attribute that can be shown according to an underlying order, such as date. Unfortunately, information useful for organizing text content, such as topic labels, does not have an inherent meaningful order. Alphabetical order is useful for looking up individual items, but not for seeing patterns across items according to adjacency, as in the case for ordered data types like dates and size.

10.7 Using Relevance Judgements

An important part of the information access process is query reformulation, and a proven effective technique for query reformulation is **relevance feedback**. In its original form, relevance feedback refers to an interaction cycle in which the user selects a small set of documents that appear to be relevant to the query, and the system then uses features derived from these selected relevant documents to revise the original query. This revised query is then executed and a new set of documents is returned. Documents from the original set can appear in the new results list, although they are likely to appear in a different rank order. Relevance feedback in its original form has been shown to be an effective mechanism for improving retrieval results in a variety of studies and settings [163, 72, 24]. In recent years the scope of ideas that can be classified under this term has widened greatly.

Relevance feedback introduces important design choices, including which operations should be performed automatically by the system and which should be user-initiated and controlled. Bates discusses this issue in detail [15], asserting that despite the emphasis in modern systems to try to automate the entire process, an intermediate approach in which the system helps automate search at a *strategic* level is preferable. Bates suggests an analogy of an automatic camera versus one with adjustable lenses and shutter speeds. On many occasions, a quick, easy method that requires little training or thought is appropriate. At other times the user needs more control over the operation of the machinery, while still not wanting to know about the low-level details of its operation.

A related idea is that, for any interface, control should be described in terms of the task being done, not in terms of how the machine can be made to accomplish the task [139]. Continuing the camera analogy, the user should be able to control the mood created by the photograph, rather than the adjustment of the lens. In information access systems, control should be over the kind of information returned, not over which terms are used to modify the query. Unfortunately it is often quite difficult to build interfaces to complex systems that behave in this manner.

10.7.1 Interfaces for Standard Relevance Feedback

A standard interface for relevance feedback consists of a list of titles with checkboxes beside the titles that allow the user to mark relevant documents. This can imply either that unmarked documents are not relevant or that no opinion has been made about unmarked documents, depending on the system. Another option is to provide a choice among several checkboxes indicating relevant or not relevant (with no selection implying no opinion). In some cases users are allowed to indicate a value on a relevance scale [17]. Standard relevance feedback algorithms usually do not perform better given negative relevance judgement evidence [47], but machine learning algorithms can take advantage of negative feedback [143, 101].

After the user has made a set of relevance judgements and issued a search command, the system can either automatically reweight the query and re-execute the search, or generate a list of terms for the user to select from in order to augment the original query. (See Figure 10.25, taken from [96].) Systems usually do not suggest terms to remove from the query.

After the query is re-executed, a new list of titles is shown. It can be helpful to retain an indicator such as a marked checkbox beside the documents that the user has already judged. A difficult design decision concerns whether or not to show documents that the user has already viewed towards the top of the ranked list [1]. Repeatedly showing the same set of documents at the top may inconvenience a user who is trying to create a large set of relevant documents, but at the same time, this can serve as feedback indicating that the revised query does not downgrade the ranking of those documents that have been found especially important. One solution is to retain a separate window that shows the rankings of only the documents that have not been retrieved or ranked highly previously. Another solution is to use smaller fonts or grey-out color for the titles of documents already seen.

Creating multiple relevance judgements is an effortful task, and the notion of relevance feedback is unfamiliar to most users. To circumvent these problems, web-based search engines have adopted the terminology of "more like this" as a simpler way to indicate that the user is requesting documents similar to the selected one. This "one-click" interaction method is simpler than standard relevance feedback dialogue which requires users to rate a small number of documents and then request a re-ranking. Unfortunately, in most cases relevance feedback requires many relevance judgements in order to work well. To partly alleviate this problem, Aalbersberg [1] proposes incremental relevance feedback which works well given only one relevant document at a time and thus can be used to

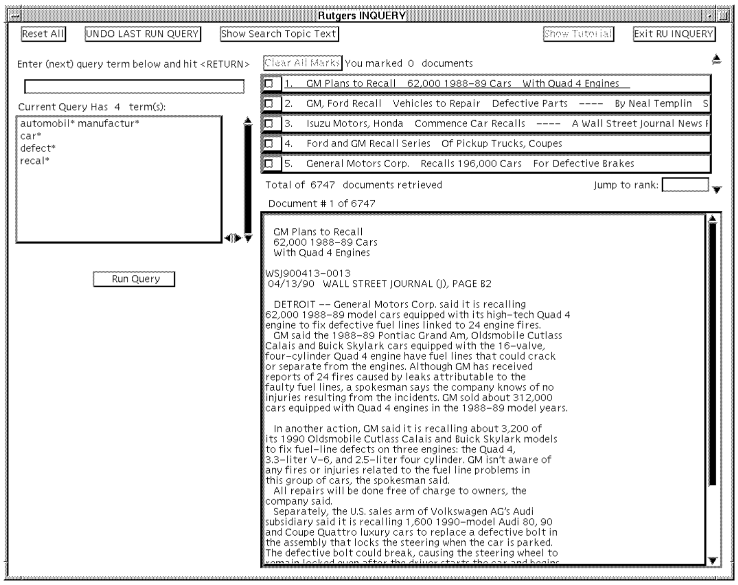

Figure 10.25: An example of an interface for relevance feedback (from Koenemann and Belkin 1996).

hide the two-step procedure from the user.

10.7.2 Studies of User Interaction with Relevance Feedback Systems

Standard relevance feedback assumes the user is involved in the interaction by specifying the relevant documents. In some interfaces users are also able to select which terms to add to the query. However, most ranking and reweighting algorithms are difficult to understand or predict (even for the creators of the algorithms!) and so it might be the case that users have difficulties controlling a relevance feedback system explicitly.

A recent study was conducted to investigate directly to what degree user control of the feedback process is beneficial. Koenemann and Belkin [96] measured the benefits of letting users "under the hood" during relevance feedback. They tested four cases using the Inquery system [180]:

- Control: No relevance feedback; the subjects could only reformulate the query by hand.
- Opaque: The subjects simply selected relevant documents and saw the revised rankings.
- Transparent: The subjects could see how the system reformulated the queries (that is, see which terms were added – the system did not reweight the subjects' query terms) and the revised rankings.
- Penetrable: The system is stopped midway through the reranking process. The subjects are shown the terms that the system would have used for opaque and transparent query reformulation. The subjects then select which, if any, of the new terms to add to the query. The system then presents the revised rankings.

The 64 subjects were much more effective (measuring precision at a cutoff of top 5, top 10, top 30, and top 100 documents) with relevance feedback than without it. The penetrable group performed significantly better than the control, with the opaque and transparent performances falling between the two in effectiveness. Search times did not differ significantly among the conditions, but there were significant differences in the number of feedback iterations. The subjects in the penetrable group required significantly fewer iterations to achieve better queries (an average of 5.8 cycles in the penetrable group, 8.2 cycles in the control group, 7.7 cycles in the opaque group, and surprisingly, the transparent group required more cycles, 8.8 on average). The average number of documents marked relevant ranged between 11 and 14 for the three conditions. All subjects preferred relevance feedback over the baseline system, and several remarked that they preferred the "lazy" approach of selecting suggested terms over having to think up their own.

An observational study on a TTY-based version of an online catalog system [71] also found that users performed better using a relevance feedback mechanism that allowed manual selection of terms. However, a later observational study did not find overall success with this form of relevance feedback [70]. The authors attribute these results to a poor design of a new graphical interface. These results may also be due to the fact that users often selected only one relevant document before performing the feedback operation, although they were using a system optimized from multiple document selection.

10.7.3 Fetching Relevant Information in the Background

Standard relevance feedback is predicated on the goal of improving an ad hoc query or building a profile for a routing query. More recently researchers have begun developing systems that monitor users' progress and behavior over long interaction periods in an attempt to predict which documents or actions the user is likely to want in future. These systems are called semi-automated **assistants** or recommender "agents," and often make use of machine learning techniques [131]. Some of these systems require explicit user input in the form of a goal statement [88] or relevance judgements [143], while others quietly record users' actions and try to make inferences based on these actions.

A system developed by Kozierok and Maes [101, 122] makes predictions about how users will handle email messages (what order to read them in, where to file them) and how users will schedule meetings in a calendar manager application. The system "looks over the shoulder" of the users, recording every relevant action into a database. After enough data has been accumulated, the system uses a nearest-neighbors method [171] to predict a user's action based on the similarity of the current situation to situations already encountered. For example, if the user almost always saves email messages from a particular person into a particular file, the system can offer to automate this action the next time a message from that person arrives [122]. This system integrates learning from both implicit and explicit user feedback. If a user ignores the system's suggestion, the system treats this as negative feedback, and accordingly adds the overriding action to the action database. After certain types of incorrect predictions, the system asks the user questions that allow it to adjust the weight of the feature that caused the error. Finally, the user can explictly train the system by presenting it with hypothetical examples of input-action pairs.

Another system, Syskill and Webert [143], attempts to learn a user profile based on explicit relevance judgements of pages explored while browsing the web. In a sense this is akin to standard relevance feedback, except the user judgements are retained across sessions and the interaction model differs: as the user browses a new web page, the links on the page are automatically annotated as to whether or not they should be relevant to the user's interest.

A related system is Letizia [113], whose goal is to bring to the user's attention a percentage of the available next moves that are most likely to be of interest, given the user's earlier actions. Upon request, Letizia provides recommendations for further action on the user's part, usually in the form of suggestions of links to follow when the user is unsure what to do next. The system monitors the user's behavior while navigating and reading web pages, and concurrently evaluates the links reachable from the current page. The system uses only implicit feedback. Thus, saving a page as a bookmark is taken as strong positive evidence for the terms in the corresponding web page. Links skipped are taken as negative support for the information reachable from the link. Selected links can indicate positive or negative evidence, depending on how much time the user spends on the resulting page and whether or not the decision to leave a page quickly is later reversed. Additionally, the evidence for user interest remain persistent across browsing sessions. Thus a user who often reads kayaking pages is at another time reading the home page of a professional contact may be alerted to the fact that the colleague's personal interests page contains a link to a shared hobby. The system uses a best-first search strategy and heuristics to determine which pages to recommend most strongly.

A more user-directed approach to pre-fetching potentially relevant information is seen in the Butterfly system [120]. This interface helps the user follow a series of citation links from a given reference, an important information seeking strategy [15]. The system automatically examines the document the user is currently reading and prefetches the bibliographic citations it refers to. It also retrieves lists of articles that cite the focus document. The underlying assumption is that the services from which the citations are requested do not respond immediately. Rather than making the user wait through the delay after each request, the system handles many requests in parallel and the interface uses graphics and animations to show the incrementally growing list of available citations. The system does not try to be clever about which cites to bring first; rather the user can watch the "organically" growing visualization of the document and its citations, and based on what looks relevant, direct the system as to which parts of the citation space to spend more time on.

10.7.4 Group Relevance Judgements

Recently there has been much interest in using relevance judgements from a large number of different users to rate or rank information of general interest [156]. Some variations of this **social recommendation** approach use only similarity among relevance judgements by people with similar tastes, ignoring the representation of the information being judged altogether. This has been found highly effective for rating information in which taste plays a major role, such as movie and music recommendations [165]. More recent work has combined group relevance judgements with content information [13].

10.7.5 Pseudo-Relevance Feedback

At the far end of the system versus user feedback spectrum is what is informally known as pseudo-relevance feedback. In this method, rather than relying on the user to choose the top k relevant documents, the system simply assumes that its top-ranked documents are relevant, and uses these documents to augment the query with a relevance feedback ranking algorithm. This procedure has been found to be highly effective in some settings [174, 104, 2], most likely those in which the original query statement is long and precise. An intriguing extension to this idea is to use the output of clustering of retrieval results as the input to a relevance feedback mechanism, either by having the user or the system select the cluster to be used [79], but this idea has not yet been evaluated.

10.8 Interface Support for the Search Process

The user interface designer must make decisions about how to arrange various kinds of information on the computer screen and how to structure the possible sequences of interactons. This design problem is especially daunting for a complex activity like information access. In this section we discuss design choices surrounding the layout of information within complex information systems, and illustrate the ideas with examples of existing interfaces. We begin with a discussion of very simple search interfaces, those used for string search in "find" operations, and then progress to multi-window interfaces and sophisticated workspaces. This is followed by a discussion of the integration of scanning, selecting and querying within information access interfaces and concludes with interface support for retaining the history of the search process.

10.8.1 Interfaces for String Matching

A common simple search need is that of the "find" operation, typically run over the contents of a document that is currently being viewed. Usually this function does not produce ranked output, nor allow Boolean combinations of terms; the main operation is a simple string match (without regular expression capabilities). Typically a special-purpose search window is created, containing a few simple controls (e.g., case-sensitivity, search forward or backward). The user types the query string into an entry form and string matches are highlighted in the target document (see Figure 10.26).

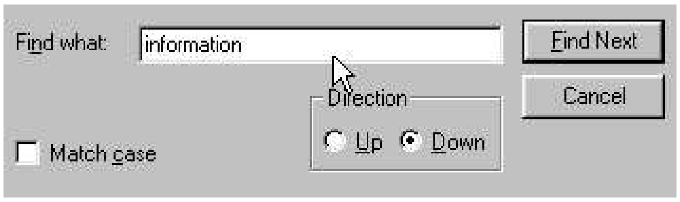

Figure 10.26: An example of a simple interface for string matching, from Netscape Communicator 4.05.

The next degree of complexity is the "find" function for searching across small collections, such as the files on a personal computer's hard disk, or the history list of a web browser. This type of function is also usually implemented as a simple string match. Again, the controls and parameter settings are shown at the top of a special-purpose search window and the various options are set via checkboxes and entry forms. The difference from the previous example is that a results list is shown within the search interface itself (see Figure 10.27).

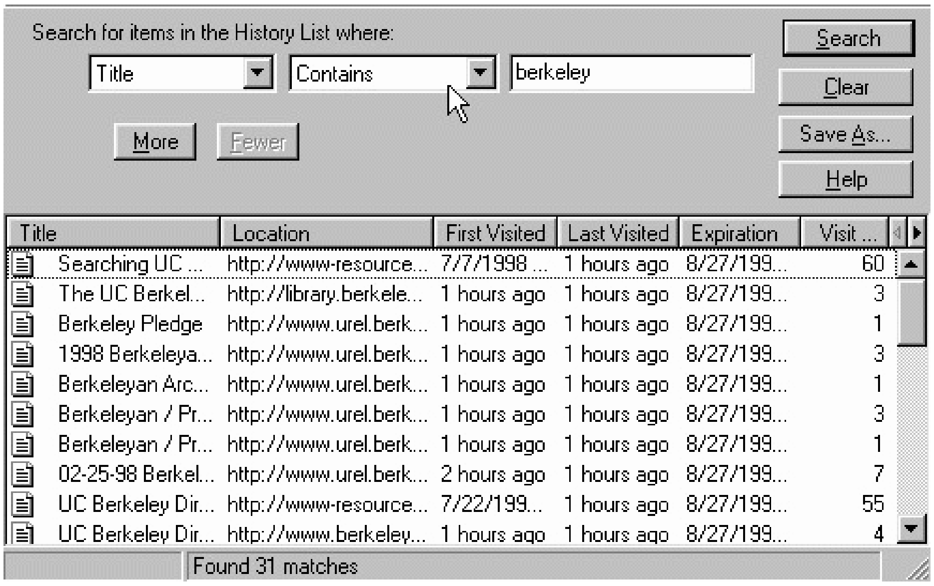

Figure 10.27: An example of an string matching over a list, in this case, a history of recently viewed web pages, from Netscape Communicator 4.05.

A common problem arises even in these very simple interfaces. An ambiguous state occurs in which the results for an earlier search are shown while the user is entering a new query or modifying the previous one. If the user types in new terms and but then does not activate the search, the interface takes on a potentially misleading state, since a user could erroneously assume that the old search hits shown correspond to the newly typed-in query. One workaround for this problem is to clear the results list as soon as the user begins to type in a new query.

However, the user may want to refer to terms shown in the search results to help reformulate the query, or may decide not to issue the new query and instead continue with the previous results. These goals would be hampered by erasing the current result set as soon as the new query is typed. Another solution is to bring up a new window for every new query. However, this requires the user to execute an additional command and can lead to a proliferation of windows. A third, probably more workable solution, is to automatically "stack" the queries and results lists in a compact format and allow the user to move back and forth among the stacked up prior searches.

Simple interfaces like these can be augmented with functionality that can greatly aid initial query formulation. Spelling errors are a major cause of void result sets. A spell-checking function that suggests alternatives for query terms that have low frequency in the collection might be useful at this stage. Another option is to suggest thesaurus terms associated with the query terms at the time the query terms are entered. Usually these kinds of information are shown after the query is entered and documents have been retrieved, but an alternative is to provide this information as the user enters the query, in a form of query preview.

10.8.2 Window Management

For search tasks more complex than the simple string matching find operations described above, the interface designer must decide how to lay out the various choices and information displays within the interface.

As discussed above, traditional bibliographic search systems use TTY-based command-line interfaces or menus. When the system responds to a command, the new results screen obliterates the contents of the one before it, requiring the user to remember the context. For example, the user can usually see only one level of a subject hierarchy at a time, and must leave the subject view in order to see query view or the document view. The main design choices in such a system are in the command or menu structure, and the order of presentation of the available options.

In modern graphical interfaces, the windowing system can be used to divide functionality into different, simultaneously displayed views [135]. In information access systems, it is often useful to link the information from one window to the information in another, for example, linking documents to their position in a table of contents, as seen in SuperBook. Users can also use the selection to cut and paste information from one window into another, for example, copy a word from a display of thesaurus terms and paste the word into the query specification form.

When arranging information within windows, the designer much choose between a **monolithic** display, in which all the windows are laid out in predefined positions and are all simultaneously viewable, **tiled windows**, and **overlapping windows**. User studies have been conducted comparing these options when applied to various tasks [168, 20]. Usually the results of these studies depend on the domain in which the interface is used, and no clear guidelines have yet emerged for information access interfaces.

The monolithic interface has several advantages. It allows the designer to control the organization of the various options, makes all the information simultaneously viewable, and places the features in familiar positions, making them easier to find. But monolithic interfaces have disadvantages as well. They often work best if occupying the full viewing screen, and the number of views is inherently limited by the amount of room available on the screen (as opposed to overlapping windows which allow display of more information than can fit on the screen at once). Many modern work-intensive applications adopt a monolithic design, but this can hamper the integration of information access with other work processes such as text editing and data analysis. Plaisant et al. [149] discuss issues relating to coordinating information across different windows to providing overview plus details.

A problem for any information access interface is an inherent limit in how many kinds of information can be shown at once. Information access systems must always reserve room for a text display area, and this must take up a significant proportion of screen space in order for the text to be legible. A tool within a paint program, for example, can be made quite small while nevertheless remaining recognizable and useable. For legibility reasons, it is difficult to compress many of the information displays needed for an information access system (such as lists of thesaurus terms, query specifications, and lists of saved titles) in this manner. Good layout, graphic, and font design can improve the situation; for example, web search results can look radically different depending on spacing, font, and other small touches [133].

Overlapping windows provide flexibility in arrangement, but can quickly lead to a crowded, disorganized display. Researchers have observed that much user activity is characterized by movement from one set of functionally related windows to another. Bannon et al. [11] define the notion of a **workspace** – the grouping together of sets of windows known to be functionally related to some activity or goal – arguing that this kind of organization more closely matches users' goal structure than individual windows [20]. Card et al. [26] also found that window usage could be categorized according to a "working set" model. They looked at the relationship between the demands of the task and the number of windows in use, and found the largest number of individual windows were in use when users transitioned from one task to another.

Based on these and other observations, Hendersen and Card [45] built a system intended to make it easier for users to move between "multiple virtual workspaces" [20]. The system uses a 3D spatial metaphor, where each workspace is a "room", and users transition between workspaces by "moving" through virtual doors (see Figure 10.28). By "traveling" from one room to the next, users can change from one work context to another. In each work context, the application programs and data files that are associated with that work context are visible and readily available for reopening and perusal. The workspace notion as developed by Card et al. also emphasizes the importance of having sessions persist across time. The user should be able to leave a room dedicated to some task, work on another task, and three days later return to the first room and see all of the applications still in the same state as before. This notion of bundling applications and data together for each task has since been widely adopted by window manager software in workstation operating system interfaces.

Figure 10.28: The Rooms 3D Workspace visualization (from Robertson et al. 1993).

Elastic windows [4] is an extention to the workspace or rooms notion to the organization of 2D tiled windows. The main idea is to make it easier to transition from one role or task to another, by adjusting how much of the screen real estate is consumed by the current role. The user can enlarge an entire group of windows with a simple gesture, and this resizing automatically causes the rest of the workspaces to reduce in size so they all still fit on the screen without overlap.

10.8.3 Example Systems

The following sections describe the information layout and management approaches taken by several modern information access interfaces.

The InfoGrid Layout

The InfoGrid system [153] is a typical example of a monolithic layout for an information access interface. The layout assumes a large display is available and is divided into a lefthand and righthand side (see Figure 10.29). The lefthand side is further subdivided into an area at the top that contains structured entry forms for specifying the properties of a query, a column of iconic controls lining the left side, and an area for retaining documents of interest along the bottom. The main central area is used for the viewing of retrieval results, either as thumbnail representations of the original documents, or derived organizations of the documents, such as Scatter/Gather-style cluster results. Users can select documents from this area and storage them in the holding area below or view them in the righthand side. Most of the righthand side of the display is used for viewing selected documents, with the upper portion showing metadata associated with the selected document. The area below the document display is intended to show a graphical history of earlier interactions.

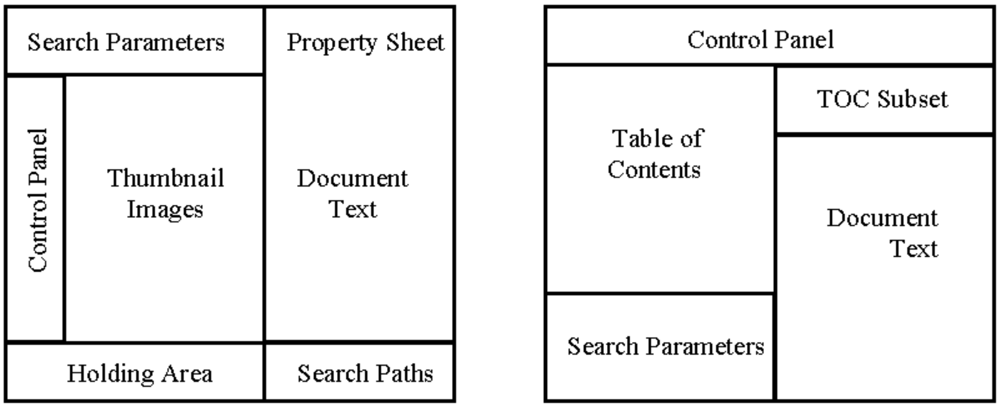

Figure 10.29: Diagrams of monolithic layouts for information access interfaces.

Designers must make decisions about which kinds of information to show in the primary view(s). If InfoGrid were used on a smaller display, either the document viewing area or the retrieval results viewing area would probably have to be shown via a pop-up overlapping window; otherwise the user would have to toggle between the two views. If the system were to suggest terms for relevance feedback, one of the existing views would have to be supplanted with this information or a pop-up window would have to be used to display the candidate terms. The system does not provide detailed information for source selection, although this could be achieved in a very simple way with a pop-up menu of choices from the control panel.

The SuperBook Layout

The layout of the InfoGrid is quite similar to that of SuperBook (see Section 10.6). The main difference is that SuperBook retains the table of contents-like display in the main lefthand pane, along with indicators of how many documents containing search hits occur in each level of the outline. Like InfoGrid, the main pane of the righthand side is used to display selected documents. Query formulation is done just below the table of contents view (although in earlier versions this appeared in a separate window). Terms related to the users' query are shown in this window as well. Large images appear in pop-up overlapping windows.

The SuperBook layout is the result of several cycles of iterative design [107]. Earlier versions used overlapping windows instead of a monolithic layout, allowing users to sweep out a rectangular area on the screen in order to create a new text box. This new text box had its own set of buttons that allowed users to jump to occurrences of highlighted words in other documents or to the table of contents. SuperBook was redesigned after noting results of experimental studies [74, 121] showing that users can be more efficient if given fewer, well-chosen interaction paths, rather than allowing wide latitude. A recent study of auditory interfaces found that although users were more efficient with a more flexible interface, they nevertheless preferred the more rigid, predictable interface [182].) The designers also took careful note of log files of user interactions. Before the redesign, users had to choose to view the overall frequency of a hit, move the mouse to the table of contents window, click the button and wait for the results to be updated. Since this pattern was observed to occur quite frequently, in the next version of the interface, the system was redesigned to automatically perform this sequence of actions immediately after a search was run.

The SuperBook designers also attempted a redesign to allow the interface to fit into smaller displays. The redesign made use of small, overlapping windows. Some of the interaction sequences that were found useful in this more constrained environment were integrated into later designs for large monolithic displays.

The DLITE Interface

The DLITE system [41, 40] makes a number of interesting design choices. It splits functionality into two parts: control of the search process and display of results (see Figure 10.30). The control portion is a graphical direct manipulation display with animation. Queries, sources, documents, and groups of retrieved documents are represented as graphical objects. The user creates a query by filling out the editable fields within a query constructor object. The system manufactures a query object, which is represented by a small icon which can be dragged and dropped onto iconic representations collections or search services. If a service is active, it responds by creating an empty results set object and attaching the query to this. A set of retrieval results is represented as a circular pool, and documents within the result set are represented as icons distributed along the perimeter of the pool. Documents can be dragged out of the results set pool and dropped into other services, such as a document summarizer or a language translator. Meanwhile, the user can make a copy of the query icon and drop it onto another search service. Placing the mouse over the iconic representation of the query causes a "tool-tips" window to pop up to show the contents of the underlying query. Queries can be stored and reused at a later time, thus facilitating retention of previously successful search strategies.

A flexible interface architecture frees the user from the restriction of a rigid order of commands. On the other hand, as seen in the SuperBook discussion, such an architecture must provide guidelines, to help get the user started, give hints about valid ways to proceed, and prevent the user from making errors. The graphical portion of the DLITE interface makes liberal use of animation to help guide the user. For example, if the user attempts to drop a query in the document summarizer icon – an illegal operation – rather than failing and giving the user an accusatory error message

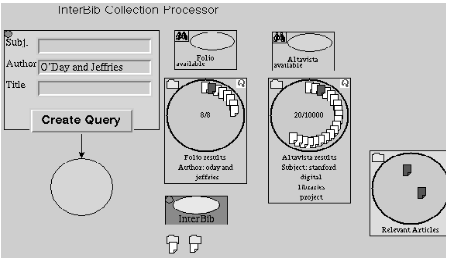

Figure 10.30: The DLITE interface (from Cousins 1997).

[39], the system takes control of the object being dropped, refusing to let it be placed on the representation for the target application, and moves the object left, right, and left again, mimicking a "shake-the-head-no" gesture. Animation is also used to help the user understand the state of the system, for example, in showing the progress of the retrieval of search results by moving the result set object away from the service from which it was invoked.

DLITE uses a separate web browser window for the display of detailed information about the retrieved documents, such as their bibligraphic citations and their full text. The browser window is also used to show Scatter/Gather-style cluster results and to allow users to select documents for relevance feedback. Earlier designs of the system attempted to incorporate text display into the direct manipulation portion, but this was found to be infeasible because of the space required [40]. Thus DLITE separates the control portion of the information access process from the scanning and reading portion. This separation allows for reusable query construction and service selection, while at the same time allowing for a legible view of documents and relationships among retrieved documents. The selection in the display view is linked to the graphical control portion, so a document viewed in the display could be used as part of a query in a query constructor.

DLITE also incorporates the notion of a workspace, or "workcenter," as it is known in this system. Different workspaces are created for different kinds of tasks. For example, a workspace for buying computer software can be equiped with source icons representing good sources of reviews of computer software, good web sites to search for price information and link to the user's online credit service.

The SketchTrieve Interface

The guiding principle behind the SketchTrieve interface [82] is the depiction of information access as an informal process, in which half-finished ideas and partly explored paths can be retained for later use, saved and brought back to compare to later interactions, and the results can be combined via operations on graphical objects and connectors between them. It has been observed [136, 166] that users use the physical layout of information within a spreadsheet to organize information. This idea motivates the design of SketchTrieve, which allows users to arrange retrieval results in a side-by-side manner to facilitate comparison and recombination (see Figure 10.31).

Figure 10.31: The SketchTrieve interface (from Hendry and Harper 1997).

The notion of a canvas or workspace for the retention of the previous context should be adopted more widely in future. Many issues are not easily solved, such as how to show the results of a set of interrelated queries, with minor modifications based on query expansion, relevance feedback, and other forms of modification. One idea is to show sets of related retrieval results as a stack of cards within a folder and allow the user to extract subsets of the cards and view them side by side, as is done in SketchTrieve, or compare them via a difference operation.

10.8.4 Examples of Poor Use of Overlapping Windows

Sometimes conversion from a command-line based interface to a graphical display can cause problems. Hancock-Beaulieu et al. [70] describe poor design decisions made in an overlapping windows display for a bibliographic system. (An improvement was found with a later redesign of the system that used a monolithic interface [69].) Problems can also occur when designers make a "literal" transformation from a TTY interface to a graphical interface. The consequences can be seen in the current Lexis-Nexis interface, which does not make use of the fact that window systems allow the user to view different kinds of information simultaneously. Instead, despite the fact that it occupies the entire screen, the interface does not retain window context when the user switches from one function to another. For example, viewing a small amount of meta-data about a list of retrieved titles causes the list of results to disappear, rather than overlaying the information with a pop-up window or rearranging the available space with resizable tiles. Furthermore, this meta-data is rendered in poorly-formatted ASCII instead of using the bitmap capabilities of a graphical interface. When a user opts to see the full text view of a document, it is shown in a small space, a few paragraphs at a time, instead of expanding to fill the entire available space.

Another bad design decision occurs when the user starts the system up. The user must fill in every field of a query window, including selecting a source or collection, before the interface will connect over the network to the main system. If the user does not have a valid account, notification of this occurs only after an entire query form has been filled out. As discussed in Section 10.4, if the user does not know which source to start with, they must still specify a source for the initial query before the system will connect and indicate what other sources are available. When the available sources are displayed, they are shown in a small window that cannot be resized, and no information other than the name of the source is provided to explain what it might contain. The mechanism for selecting the source for later use in a query is also quite awkward.

10.8.5 Retaining Search History

Section 10.3 discusses information seeking strategies and behaviors that have been observed by researchers in the field. This discussion suggests that the user interface should show what the available choices are at any given point, as well as what moves have been made in the past, short-term tactics as well as longer term strategies, and allow the user to annotate the choices made and information found along the way. Users should be able to bundle search sessions as well as save individual portions of a given search session, and flexibly access and modify each. There is also increasing interest in incorporating personal preference and usage information both into formulation of queries and use of the results of search [60].

For the most part these strategies are not supported well in current user interfaces; however some mechanisms have been introduced that begin to address these needs. In particular, mechanisms to retain prior history of the search are useful for these tasks. Some kind of history mechanism has been made available in most search systems in the past. Usually these consist of a list of the commands executed earlier. More recently, graphical history has been introduced, that allow tracking of commands and results as well. Kim and Hirtle [93] present a summary of graphical history presentation mechanisms. Recently, a graphical interface that displays web page access history in a hierarchical structure was found to require fewer page accesses and require less time when returning to pages already visited [83]

An innovation of particular interest for information access interfaces is exemplified by the saving of state in miniature form in a "slide sorter" view as exercised by VISAGE system for information visualization [160] (see Figure 10.32). The VISAGE application has the added advantage of being visual in nature and so individual states are easier to recognize. Although intended to be used as a presentation creation facility, this interface should also be useful for retaining search action history.

10.8.6 Integrating Scanning, Selection and Querying

User interfaces for information access in general do not do a good job of supporting strategies, or even of sequences of movements from one operation to the next. Even something as simple as taking the output of retrieval results from one query and using them as input to another query executed in a later search session is not well supported in most interfaces.

Hertzum and Frokjaer [5] found that users preferred an integration of scanning and query specification in their user interfaces. They did not, however, observe better results with such interactions. They hypothesized that if interactions are too unrestricted this can lead to erroneous or wasteful behavior, and interaction between two different modes requires more guidance. This suggests that more flexibility is needed, but within constraints (this argument was also made in the discussion of the SuperBook system in Section 10.6).

There are exceptions. The new Web version of the Melyvl system provides ways to take the output of one query and modify it later for re-execution (see Figure 10.33). The workspace-based systems such as DLITE and Rooms allow storage and reuse of previous state. However, these

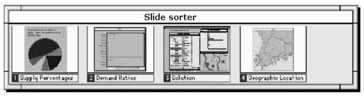

Figure 10.32: The VISAGE interaction history visualization (from Roth et al. 1997).

systems do not integrate the general search process well with scanning and selection of information from auxiliary structures. Scanning, selection, and querying needs to be better integrated in general. This discussion will conclude with an example of an interface that does attempt to tightly couple querying and browsing.

The Cat-a-Cone interface integrates querying and browsing of very large category hierarchies with their associated text collections. The prototype system uses 3D+animation interface components from the Information Visualizer [30], applied in a novel way, to support browsing and search of text collections and their category hierarchies. A key component of the interface is the separation of the graphical representation of the category hierarchy from the graphical representation of the documents. This separation allows for a fluid, flexible interaction between browsing and search, and between categories and documents. It also provides a mechanism by which a *set* of categories associated with a document can be viewed along with their hierarchical context. See Figure 10.34.

Another key component of the design is assignment of first-class status to the representation of text content. The retrieved documents are stored in a 3D+animation book representation [30] that allows for compact display of moderate numbers of documents. Associated with each retrieved document is a page of links to the category hierarchy and a page of text showing the document contents. The user can "ruffle" the pages of the book of retrieval results and see corresponding changes in the category hierarchy, which is also represented in 3D+animation. All and only those parts of the category space that reflect the semantics of the retrieved document are shown with that document.

The system allows for several different kinds of starting points. Users can start by typing in a name of a category and seeing which parts of the category hierarchy match it. For example, Figure 10.35 shows the results of searching on "Radiation" over the MeSH terms in this subcollection. The

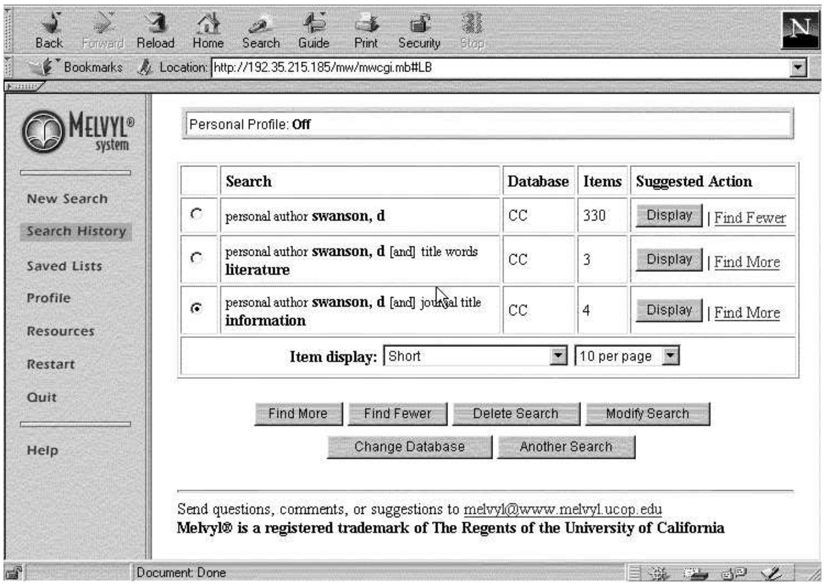

Figure 10.33: A view of query history revision in the Web-based version of the Melvyl bibliographic catalog.

word appears under four main headings (*Physical Sciences, Diseases, Diagnostics*, and *Biological Sciences*). The hierarchy immediately shows why "Radiation" appears under *Diseases* – as part of a subtree on occupational hazards. Now the user can select one or more of these category labels as input to a query specification.

Another way the user can start is by simply typing in a free text query into an entry label. This query is matched against the collection. Relevant documents are retrieved and placed in the book format. When the user "opens" the book to a retrieved document, the parts of the category hierarchy the correspond to that retrieved documents are shown in the hierarchical representation. Thus multiple intersecting categories can be shown simultaneously, in their hierarchical context. Thus this interface fluidly combines large, complex metadata, starting points, scanning and querying into one interface.

Recall the evaluation of the Kohonen feature map representation discussed in Section 10.4. The experimenters found that some users expressed a desire for a visible hierarchical organization, others wanted an ability to zoom in on a subarea to get more detail, and some users disliked having to look through the entire map to find a theme, desiring an alphabetical ordering instead. The subjects like the ease of being able to jump from one area to another without having to back up (as is required in Yahoo) and liked the fact that the maps have varying levels of granularity.

These results all support the design decisions made in the Cat-a-Cone. Hierarchical representation of term meanings is supported, so users can choose which level of description is meaningful to them. Furthermore, different levels of description can be viewed simultaneously, so more familiar concepts can be viewed in more detail, and less familiar at a more general level. An alphabetical ordering of the categories coupled with a regular expression search mechanism allows for straight-

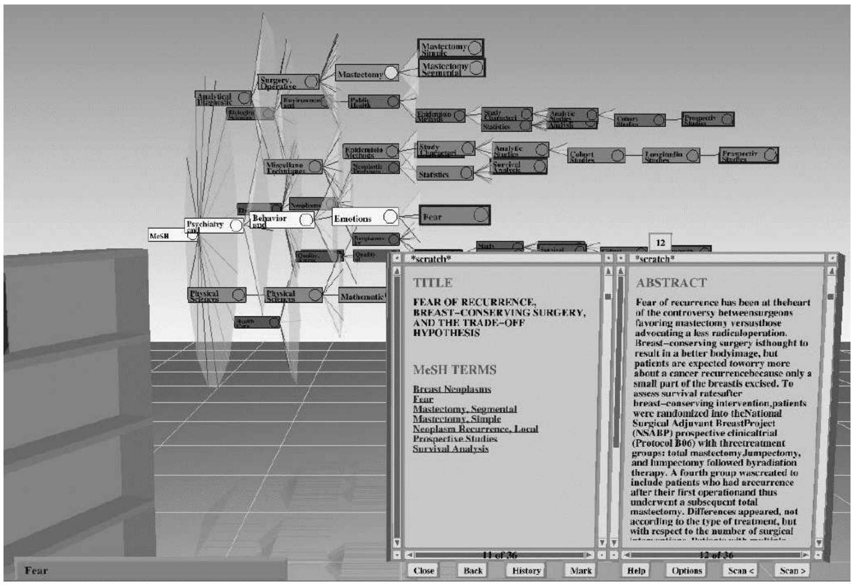

Figure 10.34: The Cat-a-Cone interface for integrating category and text scanning and search (Hearst and Karadi 1997).

forward location of category labels. Retrieved documents are represented as first-class objects, so full text is visible, but in a compact form. Category labels are disambiguated by their ancestor/descendant/sibling representation. Users can jump easily from one category to another and can in addition query on multiple categories simultaneously (something that is not a natural feature of the maps). The Cat-a-Cone has several additional advantages as well, such as allowing a document to be placed at the intersection of several categories, and explicitly linking document contents with the category representation.

10.9 Conclusions

The field of human-computer interaction is a broad one, and this chapter touches on only a small subset of pertinent issues. For further information, see the excellent texts on user interface design by Shneiderman [168], information seeking behavior by Marchionini [123], and digital libraries by Lesk [111]. An excellent book on visual design is that of Mullet and Sano [133]. Tufte has written thought-provoking and visually engaging books on the power of information visualization [178, 179] and a collection of papers on information visualization has been edited by Card et al. [27].

This chapter has discussed many ideas for improving the human-computer interaction experience for information seekers. This is the most rapidly developing area of information access today, and improvements in the interface are likely to lead the way toward better search results and better-enabled information creators and users. Research in the area of human-computer interaction is difficult because the field is relatively new, and because it can be difficult to obtain strong results when running user studies. These challenges should simply encourage those who really want to

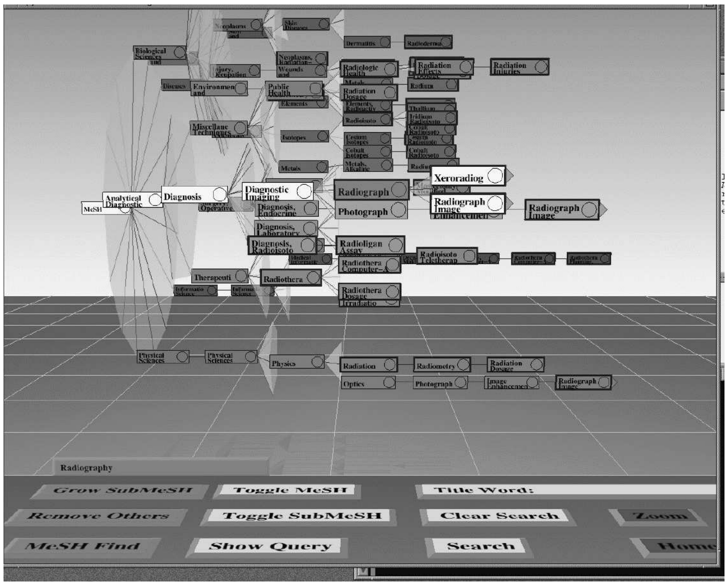

Figure 10.35: An interface for a starting point for searching over category labels (Hearst and Karadi 1997).

influence the information access systems of tomorrow.

Acknowledgements The author gratefully acknowledges the generous and helpful comments on the contents of this chapter by Gary Marchionini and Ben Shneiderman, the excellent administrative assistance of Barbara Goto, and the great faith and patience of Ricardo Baeza-Yates and Berthier Ribeiro-Neto.

Bibliography

[1] IJsbrand Jan Aalbersberg. Incremental relevance feedback. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Interaction in Information Retrieval, pages 11–22, 1992.

[2] James Allan. Relevance feedback with too much data. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Feedback Methods, pages 337–343, 1995.

[3] Robert B. Allen, Pascal Obry, and Michael Littman. An interface for navigating clustered document sets returned by queries. In *Proceedings of ACM COOCS: Conference on Organizational Computing Systems*, Milpitis, CA, November 1993.

[4] Eser Kandogan andBen Shneiderman. Elastic windows: Evaluation of multi-window operations. In *Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems*, volume 1, pages 250–257, 1997.

[5] Morten Hertzum andErik Frokjaer. Browsing and querying in online documentation: A study of user interfaces and the interaction process. *ACM Transactions on Computer-Human Interaction*, 3(2):136–161, 1996.

[6] P. Anick, J. Brennan, R. Flynn, D. Hanssen, B. Alvey, and J. Robbins. A direct manipulation interface for boolean information retrieval via natural language query. In *Proceedings of the 13th Annual International ACM/SIGIR Conference*, pages 135–150, Brussels, Belgium, 1990.

[7] Peter G. Anick. Adapting a full-text information retrieval system to the computer troubleshooting domain. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 349–358, 1994.

[8] Ricardo Baeza-Yates. Visualizing large answers in text databases. In *Int. Workshop on Advanced User Interfaces (AVI'96)*, pages 101–107, Gubbio, Italy, May 1996. ACM Press.

[9] Michelle Q. Wang Baldonado and Terry Winograd. Sensemaker: An information-exploration interface supporting the contextual evolution of a user's interests. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, May 1997. ACM.

[10] Michelle Q. Wang Baldonado and Terry Winograd. Hi-cites: Dynamically-created citations with active highlighting. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Los Angeles, CA, March 1998. ACM.

[11] Liam Bannon, Allen Cypher, Steven Greenspan, and Melissa L. Monty. Evaluation and analysis of users' activity organization. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 54–57, 1983.

[12] Brian T. Bartell, Garrison W. Cottrell, and Richard K. Belew. Automatic combination of multiple ranked retrieval systems. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Statistical Models, pages 173–181, 1994.

[13] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of AAAI 98*, pages 714–720, Madison, WI, July 1998.

[14] Marcia J. Bates. The design of browsing and berrypicking techniques for the on-line search interface. *Online Review*, 13(5):407–431, 1989.

[15] Marcia J. Bates. Where should the person stop and the information search interfaces start? *Information Processing & Management*, 26(5), 1990.

[16] Benjamin B. Bederson, James D. Hollan, Ken Perlin, Jonathan Meyer, David Bacon, and George Furnas. Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. *Journal of Visual Languages and Computing*, 7(1):3–31, 1996.

[17] Ric Belew. Rave reviews: Acquiring relevance assessments from multiple users. In Marti A. Hearst and Haym Hirsh, editors, *Working Notes of the AAAI Spring Symposium on Machine Learning in Information Access*, Stanford, CA, March 1996. AAAI.

[18] N. Belkin, P. G. Marchetti, and C. Cool. Braque – design of an interface to support user interaction in information retrieval. *Information Processing and Management*, 29(3):325–344, 1993.

[19] Eric A. Bier, Maureen C. Stone, Ken Pier, Ken Fishkin, Thomas Baudel, Matt Conway, William Buxton, and Tony DeRose. Toolglass and magic lenses: The see-through interface. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 2 of *VIDEOS: Part II – Techniques for Improved Human-Computer Interaction*, pages 445–446, 1994.

[20] Patricia A. Billingsley. Taking panes: Issues in the design of windowing systems. In Martin Helander, editor, *Handbook of Human-Computer Interaction*, pages 413–436. Springer Verlag, 1988.

[21] Christine L. Borgman. Why are online catalogs hard to use? lessons learned from information retrieval studies. *Journal of the American Society for Information Science*, 37(6):387–400, 1996.

[22] Christine L. Borgman. Why are online catalogs still hard to use? *Journal of the American Society for Information Science*, 47(7):493–503, 1996.

[23] James Boyle, William Ogden, Steven Uhlir, and Patricia Wilson. Qmf usability: How it really happened. In *Proceedings of IFIP INTERACT'84: Human-Computer Interaction*, pages 877–882, 1984.

[24] Chris Buckley, Gerard Salton, and James Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Routing, pages 292–300, 1994.

[25] Robin Burke, Kristian Hammond, Vladimir Kulukin, Steven Lytinen, Noriko Tomuro, and Scott Schoenberg. Experiences with the FAQ system. *AI Magazine*, 18(2):57–66, 1997.

[26] S. K. Card, M. Pavel, and J. E. Farrell. Window-based computer dialogues. In *Proceedings of Interact '84, First IFIP Conference on Human-Computer Interaction*, pages 355–359, 1984.

[27] Stuart K. Card, Jock D. MacKinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think.* Morgan Kaufman Publishers, 1998.

[28] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction.* L. Erlbaum Associates, 1983.

[29] Stuart K. Card, George G. Robertson, and Jock D. Mackinlay. The information visualizer, an information workspace. In *Proceedings of the ACM CHI '91 Human Factors in Computing Systems*, pages 181–188, 1991.

[30] Stuart K. Card, George G. Robertson, and William York. The webbook and the web forager: An information workspace for the world-wide web. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, Canada, April 1996.

[31] S. Carliner. Designing wizards. *Training & Development*, 52(7):62–63, 1998.

[32] Claudio Carpineto and Giovanni Romano. Information retrieval through hybrid navigation of lattice representations. *International Journal of Human-Computer Studies*, 45(5):553–578, 1996.

[33] Matthew Chalmers and Paul Chitson. Bead: Exploration in information visualization. In *Proceedings of the 15th Annual International ACM/SIGIR Conference*, pages 330–337, Copenhagen, Denmark, 1992.

[34] Shan-Ju Chang and Ronald E. Rice. Browsing: A multidimensional framework. *Annual Review of Information Science and Technology*, 28:231–276, 1993.

[35] Hsinchen Chen, Andrea L. Houston, Robin R. Sewell, and Bruce R. Schatz. Internet browsing and searching: User evaluations of category map and concept space techniques. *Journal of the American Society for Information Sciences (JASIS)*, 49(7), 1998.

[36] Michael Chen and Marti A. Hearst. Presenting web site search results in context: A demonstration. In *Proceedings of the 20th Annual International ACM/SIGIR Conference*, 1998. To appear.

[37] Charles L. A. Clarke, Grodon V. Cormack, and Forbes J. Burkowski. Shortest substring ranking (multitext experiments for TREC-4). In Donna Harman, editor, *Proceedings of the Fourth Text Retrieval Conference TREC-4*. National Institute of Standards and Technology Special Publication, 1996.

[38] Jeff Conklin. Hypertext: An introduction and survey. *Computer Magazine of the Computer Group News of the IEEE Computer Group Society*, 20(9), September 1987.

[39] Alan Cooper. *About Face: The Essentials of User Interface Design.* IDG Books, 1995.

[40] Steve B. Cousins. *Reification and Affordances in a User Interface for Interacting with Heterogeneous Distributed Applications.* PhD thesis, Stanford University, May 1997.

[41] Steve B. Cousins, Andreas Paepcke, Terry Winograd, Eric A. Bier, and Ken Pier. The digital library integrated task environment. In *Proceedings of the ACM Conference on Digital Libraries*, Philadelphia, PA, July 1997. ACM.

[42] Douglass R. Cutting, David Karger, and Jan Pedersen. Constant interaction-time Scatter/Gather browsing of very large document collections. In *Proceedings of the 16th Annual International ACM/SIGIR Conference*, pages 126–135, Pittsburgh, PA, 1993.

[43] Douglass R. Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM/SIGIR Conference*, pages 318–329, Copenhagen, Denmark, 1992.

[44] Allen Cypher, editor. *Watch What I Do: Programming by Demonstration.* MIT Press, Cambridge, 1993.

[45] Jr. D. Austin Henderson and Stuart K. Card. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics*, 5(3):211–243, 1986.

[46] Karen M. Drabenstott and Marjorie S. Weller. The exact-display approach for online catalog subject searching. *Information Processing and Management*, 32(6):719–745, 1996.

[47] Mark D. Dunlop. The effect of accessing nonmatching documents on relevance feedback. *ACM Transactions on Information Systems*, 15(2):137–153, 1997.

[48] Deborah M. Edwards and Lynda Hardman. 'lost in hyperspace': Cognitive mapping and navigation in a hypertext environment. In Ray McAleese, editor, *HYPERTEXT I: Theory into Practice*, pages 105–125. Ablex Publishing Corporation, 1988.

[49] Dennis E. Egan. Individual differences in human-computer interaction. In Martin Helander, editor, *Handbook of Human-Computer Interaction*, pages 543–568. Springer Verlag, 1988.

[50] Dennis E. Egan, Michael E. Lesk, R. Daniel Ketchum, Carol C. Lochbaum, Joel R. Remde, Louis M. Gomez, and Thomas K. Landauer. Hypertext for the electronic library? core sample results. In *Proceedings of the ACM Hypertext Conference*, pages 299–312, May 1991.

[51] Dennis E. Egan, Joel R. Remde, Louis M. Gomez, Thomas K. Landauer, Jennifer Eberhardt, and Carol C. Lochbaum. Formative design evaluation of SuperBook. *Transaction on Information Systems*, 7(1), 1989.

[52] Dennis E. Egan, Joel R. Remde, Thomas K. Landauer, Carol C. Lochbaum, and Louis M. Gomez. Behavioral evaluation and analysis of a hypertext browser. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 205–210, May 1989.

[53] Stephen G. Eick. Graphically displaying text. *Journal of Computational and Graphical Statistics*, 3(2):127–142, June 1994.

[54] Stephen G. Eick and Gary J. Wills. High interaction graphics. *European Journal of Operations Research*, 81(3):445–459, March 1995.

[55] D. Ellis. A behavioural model for information retrieval system design. *Journal of Information Science*, 15:237–247, 1989.

[56] Gerhard Fischer and Helga Nieper-Lemke. Helgon: Extending the retrieval reformulation paradigm. In *Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems*, pages 357–362, 1989.

[57] Ken Fishkin and Maureen C. Stone. Enhanced dynamic queries via movable filters. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1 of *Papers: Information Visualization*, pages 415–420, 1995.

[58] Richard H. Fowler, Wendy A. L. Fowler, and Bradley A. Wilson. Integrating query, thesaurus, and documents through a common visual representation. In *Proceedings of the 14th Annual International ACM/SIGIR Conference*, pages 142–151, Chicago, 1991.

[59] Edward A. Fox, Deborah Hix, Lucy T. Nowell, Dennis J. Brueni, William C. Wake, Lenwwod S. Heath, and Durgesh Rao. Users, user interfaces, and objects: Envision, a digital library. *Journal of the American Society for Information Science*, 44(8):480–491, 1993.

[60] Eric Freeman, , and Scott Fertig. Lifestreams: Organizing your electronic life. In Robin Burke, editor, *Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, Cambridge, MA, November 1995. AAAI.

[61] H. P. Frei and D. Stieger. The use of semantic links in hypertext information retrieval. *Information Processing & Management*, 31(1):1–13, 1994.

[62] George W. Furnas and Jeff Zacks. Multitrees: Enriching and reusing hierarchical structure. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 2, page 223, 1994.

[63] Richard Furuta, Frank M. Shipman III, Catherine C. Marshall, Donald Brenner, and Hao wei Hsieh. Hypertext paths and the world-wide web: Experiences with walden's paths. In *Proceedings of the Eigth ACM Conference on Hypertext*, Southampton, UK, 1997.

[64] Stephan Green, Gary Marchionini, Catherine Plaisant, and Ben Shneiderman. Previews and overviews in digital libraries: Designing surrogates to support visual information seeking. Technical Report Department of Computer Science CS-TR-3838, University of Maryland, 1997.

[65] Sharon L. Greene, Susan J. Devlin, Philip E. Cannata, and Louis M. Gomez. No ifs, ands, or ors: A study of database querying. *International Journal of Man-Machine Studies*, 32(3):303–326, 1990.

[66] Catherine Guinan and Alan F. Smeaton. Information retrieval from hypertext using dynamically planned guided tours. In *Proceedings of ourth ACM Conference on Hypertext*, pages 122–130, 1992.

[67] Frank G. Halasz, Thomas P. Moran, and Randall H. Trigg. Notecards in a nutshell. In *Proceedings of ACM CHI+GI'87 Conference on Human Factors in Computing Systems and Graphics Interface*, pages 45–52, 1987.

[68] Micheline Hancock-Beaulieu. User friendliness and human-computer interaction in online library catalogs. *Program*, 46(1):29–37, 1992.

[69] Micheline Hancock-Beaulieu. Experiments on interfaces to support query expansion. *Journal of Documentation*, 53(1):8–19, 1997.

[70] Micheline Hancock-Beaulieu, Margaret Fieldhouse, and Thien Do. An evaluation of interactive query expansion in an online library catalogue with a graphical user interface. *Journal of Documentation*, 51(3):225–243, 1995.

[71] Micheline Hancock-Beaulieu and Stephen Walker. An evaluation of automatic query expansion in an online library catalogue. *Journal of Documentation*, 48(4):406–421, 1992.

[72] Donna Harman. Relevance feedback revisited. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Interaction in Information Retrieval, pages 1–10, 1992.

[73] Stephen P. Harter. *Online Information Retrieval.* Academic Press, 1986.

[74] Alexander G. Hauptmann and Bert F. Green. A comparison of command, menu-selection and natural-language computer programs. *Behaviour and Information Technology*, 2(2):163–178, 1983.

[75] Marti A. Hearst. Tilebars: Visualization of term distribution information in full text information access. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Denver, CO, May 1995.

[76] Marti A. Hearst. Improving full-text precision using simple query constraints. In *Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval (SDAIR)*, Las Vegas, NV, 1996.

[77] Marti A. Hearst. The use of categories and clusters in organizing retrieval results. In Tomek Strzalkowski, editor, *Natural Language Information Retrieval.* Kluwer Academic Publishers, 1998. To appear.

[78] Marti A. Hearst and Chandu Karadi. Cat-a-cone: An interactive interface for specifying searches and viewing retrieval results using a large category hierarchy. In *Proceedings of the 20th Annual International ACM/SIGIR Conference*, Philadelphia, PA, 1997.

[79] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19th Annual International ACM/SIGIR Conference*, pages 76–84, Zurich, Switzerland, 1996.

[80] Matthias Hemmje, Clemens Kunkel, and Alexander Willett. LyberWorld – a visualization user interface supporting fulltext retrieval. In *Proceedings of the 17th Annual International ACM/SIGIR Conference*, pages 249–259, Dublin, Ireland, July 1994.

[81] R. Hendly, N. Drew, A. Wood, and R. Beale. Narcissus: Visualizing information. In *Proceedings of the IEEE Information Visualization Symposium '95*, pages 90–96, Atlanta, GA, Oct 1995.

[82] David G. Hendry and David J. Harper. An informal information-seeking environment. *Journal of the American Society for Information Science*, 48(11):1036–1048, 1997.

[83] Ron R. Hightower, Laura T. Ring, Jonathan I. Helfman, Benjamin B. Bederson, and James D. Hollan. Grapical multiscale web histories: A study of padprints. In *Proceedings of the Ninth ACM Conference on Hypertext*, 1998.

[84] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of AAAI 98*, Madison, WI, July 1998.

[85] Adele Howe and Danielle Dreilinger. Savvysearch: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.

[86] David A. Hull, Jan O. Pedersen, and Hinrich Schütze. Method combination for document filtering. In *Proceedings of the 19th Annual International ACM/SIGIR Conference*, pages 279–287, Zurich, Switzerland, 1996.

[87] Paul S. Jacobs and Lisa F. Rau. Innovations in text interpretation. *Artificial Intelligence*, 63(1-2):143–191, 1993.

[88] Thorsten Joachims, Dwayne Freitag, and Tom Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, 1997.

[89] Brian Johnson and Ben Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceedings of the 2nd International IEEE Visualization Conference*, pages 284–291, San Diego, 1991.

[90] Steve Jones and S. McInnes. Graphical query specification and dynamic result previews for a digital library. In *Proceedings of UIST 98, ACM Symposium on User Interface Software and Technology*, November 1998.

[91] Henry Kautz, Bart Selman, and Mehul Shah. The hidden web. *AI Magazine*, 18(2):27–36, 1997.

[92] P. R. Keller and M. M. Keller. *Visual Cues: Practical Data Visualization.* IEEE Computer Society Press, 1993.

[93] Hanhwe Kim and Stephen C. Hirtle. Spatial metaphors and disorientation in hypertext browsing. *Behaviour and Information Technology*, 14(4):23, 1995.

[94] Adrienee J. Kleiboemer, Manette B. Lazear, and Jan O. Pedersen. Tailoring a retrieval system for naive users. In *Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval (SDAIR)*, Las Vegas, NV, 1996.

[95] Jon Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[96] Jurgen Koenemann and Nicholas J. Belkin. A case for interaction: A study of interactive information retrieval behavior and effectiveness. In *Proceedings of ACM CHI 96 Conference on Human Factors in Computing Systems*, volume 1 of *PAPERS: Interactive Information Retrieval*, pages 205–212, 1996.

[97] Janet L. Kolodner. *Case-based Reasoning.* Morgan Kaufmann Publishers, 1993.

[98] Robert R. Korfhage. To see or not to see – is that the query? In *Proceedings of the 14th Annual International ACM/SIGIR Conference*, pages 134–141, Chicago, 1991.

[99] Flip Korn and Ben Shneiderman. Navigating terminology hierarchies to access a digital library of medical images. Technical Report HCIL-TR-94-03, University of Maryland, 1995.

[100] S. M. Kosslyn. Understanding charts and graphs. *Applied Cognitive Psychology*, 3:185–226, 1989.

[101] Robyn Kozierok and Pattie Maes. A learning interface agent for scheduling meetings. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pages 81–88, New York, NY, 1993.

[102] Julian Kupiec. MURAX: A robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th Annual International ACM/SIGIR Conference*, pages 181–190, Pittsburgh, PA, 1993.

[103] Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM/SIGIR Conference*, pages 68–73, Seattle, WA, 1995.

[104] K. L. Kwok, L. Grunfeld, and D. D. Lewis. Trec-3 ad-hoc, routing retrieval and thresholding experiments. In *Proceedings of the Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, 1995.

[105] Eric Lagergren and Paul Over. Comparing interactive information retrieval systems across sites: The trec-6 interactive track matrix experiment. In *Proceedings of the 21st Annual International ACM/SIGIR Conference*, pages 164–172, 1998.

[106] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 401–408, Denver, CO, May 1995.

[107] Thomas K. Landauer, Dennis E. Egan, Joel R. Remde, Michael Lesk, Carol C. Lochbaum, and Daniel Ketchum. Enhancing the usability of text through computer delivery and formative evaluation: the superbook project. In C. McKnight, A. Dillon, and J. Richardson, editors, *Hypertext: A Psychological Perspective*, pages 71–136. Ellis Horwood, 1993.

[108] Jill H. Larkin and Herbert A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[109] Ray R. Larson. Bibliometrics of the world wide web: An exploratory analysis of the intellectual structure of cyberspace. In *ASIS '96: Proceedings of the 1996 Annual ASIS Meeting*, 1996.

[110] David B. Leake, editor. *Case-based reasoning: experiences, lessons, & future directions.* AAAI press, Menlo Park, CA, 1996.

[111] Michael Lesk. *Practical Digital Libraries; Books, Bytes, & Bucks.* Morgan Kaufman, 1997.

[112] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[113] Henry Lieberman. Letizia: an agent that assists web browsing. In *Proceedings of 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929, 1995.

[114] Xia Lin, Dagobert Soergel, and Gary Marchionini. A self-organizing semantic map for information retrieval. In *Proceedings of the 14th Annual International ACM/SIGIR Conference*, pages 262–269, Chicago, 1991.

[115] Henry J. Lowe and G. Octo Barnett. Understanding and using the medical subject headings (MeSH) vocabulary to perform literature searches. *Journal of the American Medical Assocation (JAMA)*, 271(4):1103–1108, 1994.

[116] Clifford Lynch. The next generation of public access information retrieval systems for research libraries – lessons from 10 years of the melvyl system. *Information Technology and Libraries*, 11(4):405–415, 1992.

[117] Y. S. Maarek and A.J. Wecker. The librarian's assistant: Automatically assembling books into dynamic bookshelves. In *Proceedings of RIAO '94; Intelligent Multimedia Information Retrieval Systems and Management*, October 1994.

[118] Yoelle S. Maarek and Israel Z. Ben Shaul. Webcutter: A system for dynamic and tailorable site mapping. In *Proceedings of the Sixth International World Wide Web Conference*, 1997.

[119] Jock Mackinlay, George Robertson, and Stuart K. Card. The perspective wall: Detail and context smoothly integrated. In *Proceedings of the ACM CHI '91 Human Factors in Computing Systems*, pages 173–179, 1991.

[120] Jock D. Mackinlay, Ramana Rao, and Stuart K. Card. An organic user interface for searching citation links. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1 of *Papers: Information Access*, pages 67–73, 1995.

[121] A. MacLean, P. J. Barnard, and M. D. Wilson. Evaluating the human interface of a data entry system: User choice and performance measures yield different tradeoff functions. In *Proceedings of the HCI'85 Conference on People and Computers: Designing the Interface*, pages 172–185, 1985.

[122] Pattie Maes and Robyn Mozierok. Learning interface agents. In *Proceedings of AAAI 93*, pages 459–465, Washington, D.C., July 1993.

[123] Gary Marchionini. *Information Seeking in Electronic Environments*. Cambridge University Press, 1995.

[124] Karen Markey, Pauline Atherton, and Claudia Newton. An analysis of controlled vocabulary and free text search statements in online searches. *Online Review*, 4:225–236, 1982.

[125] Ray McAleese, editor. *Hypertext I: Theory into Practice*. Ablex Publishing Corporation, 1988.

[126] B. McCune, R. Tong, J.S. Dean, and D. Shapiro. Rubric: A system for rule-based information retrieval. *IEEE Transactions on Software Engineering*, 11(9), 1985.

[127] Charles T. Meadow, Barbara A. Cerny, Christine L. Borgman, and Donald O. Case. Online access to knowledge: System design. *Journal of the American Society for Information Science*, 40(2):86–98, 1989.

[128] Filippo Menczer and Richard K. Belew. Adaptive information agents in distributed textual environments. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (AGENTS-98)*, pages 157–164, May 1998.

[129] Beth Meyer, Richard A. Sit, Victoria A. Spaulding, Sherry E. Mead, and Neff Walker. Age group differences in world wide web navigation. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of ACM CHI'97 Conference on Human Factors in Computing Systems*, pages 157–164, March 1997.

[130] A. Michard. Graphical presentation of Boolean expressions in a database query language: design notes and an ergonomic evaluation. *Behaviour and Information Technology*, 1(3), 1982.

[131] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[132] Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In *Proceedings of 21st Annual International Conference on Research and Development in Information Retrieval, SIGIR 98*, Melbourne, Australia, 1998.

[133] Kevin Mullet and Darell Sano. *Designing Visual Interfaces: Communication-Oriented Techniques*. SunSoft Press, 1995.

[134] Brad Myers. *Creating User Interfaces by Demonstration*. Academic Press, New York, 1988.

[135] Brad Myers. A taxonomy of window manager user interfaces. *IEEE Computer Graphics and Applications*, pages 65–84, Sept 1988.

[136] Bonnie A. Nardi. *A Small Matter of Programming: Perspectives on End User Computing*. MIT Press, 1993.

[137] A. Nation. Visualizing websites using a hierarchical table of contents browser: Webtoc. In *Proceedings of the Third Conference on Human Factors and the Web*, Denver, CO, 1997.

[138] Jakob Nielsen. *Usability Engineering*. Academic Press, 1993.

[139] D. A. Norman. *The Psychology of Everyday Things*. Basic Books, New York, 1988.

[140] Vicki L. O'Day and Robin Jeffries. Orienteering in an information landscape: how information seekers get from here to there. In *Proceedings of the INTERCHI '93*, Amsterdam, April 1993. IOS Press.

[141] R. N. Oddy. Information retrieval through man-machine dialogue. *Journal Documentation*, 33:1–14, 1977.

[142] Kenton O'Hara and Abigail Sellen. A comparison of reading paper and on-line documents. In *Proceedings of ACM/SIGCHI 97*, Atlanta, GA, March 1997.

[143] Michael Pazzani, Daniel Billsus, and Jack Muramatsu. Syskill & webert: Identifying interesting web sites. In *Proceedings of the Thirteenth Annual National Conference on Artificial Intelligence (AAAI '96)*, Portland, OR, August 1996.

[144] Gert Schmeltz Pedersen. A browser for bibliographic information retrieval, based on an application of lattice theory. In *Proceedings of the 16th Annual International ACM/SIGIR Conference*, pages 270–279, Pittsburgh, PA, 1993.

[145] Lori Phelps. Active documentation: Wizards as a medium for meeting user needs. In *ACM 15th International Conference on Systems Documentation*, pages 207–210, 1997.

[146] Peter Pirolli, James Pitkow, and Ramana Rao. Silk from a sow's ear: Extracting usable structures from the web. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, Canada, May 1996. ACM.

[147] Peter Pirolli, Patricia Schank, Marti A. Hearst, and Christine Diehl. Scatter/gather browsing communicates the topic structure of a very large text collection. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, Canada, May 1996. ACM.

[148] Catherine Plaisant, Tom Bruns, Ben Shneiderman, and Khoa Doan. Query previews in networked information systems: the case of EOSDIS. In *Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems*, volume 2 of *Formal Video Program*, pages 202–203, 1997.

[149] Catherine Plaisant, David Carr, and Ben Shneiderman. Image-browser taxonomy and guidelines for designers. *IEEE Software*, 12(2):21–32, 1995.

[150] A Steven Pollitt. Interactive information retrieval based on faceted classification using views. In *Proceedings of the 6th International Study Conference on Classification (FID/CR)*, University College, London, June 1997.

[151] Wanda Pratt. Dynamic organization of search results using the umls. In *American Medical Informatics Association Fall Symposium*, 1997.

[152] R. Rao and S. K. Card. The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. ACM, April 1994.

[153] R. Rao, S. K. Card, H.D. Jellinek, J. D. Mackinlay, and G. G. Robertson. The information grid: A framework for building information retrieval and retrieval-centered applications. In *Proceedings of the ACM Symposium on User Interface Software and Technology.* ACM Press, Nov 1992.

[154] David F. Redmiles. Supporting software reuse through examples. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, Doctoral Consortium, page 492, 1991.

[155] Earl Rennison. Galaxy of news: An approach to visualizing and understanding expansive news landscapes. In *Proceedings of UIST 94, ACM Symposium on User Interface Software and Technology*, pages 3–12, New York, 1994.

[156] Paul Resnick and Hal Varian. Introduction: Special issue on collaborative filtering. *Communications of the ACM*, 40(3):56–58, March 1997.

[157] George C. Robertson, Stuart K. Card, and Jock D. MacKinlay. Information visualization using 3D interactive animation. *Communications of the ACM*, 36(4):56–71, 1993.

[158] David E. Rose, Richard Mander, Tim Oren, Dulce B. Ponceleon, Gitta Salomon, and Yin Yin Wong. Content awareness in a file system interface: Implementing the 'pile' metaphor for organizing information. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Interface Issues, pages 260–269, 1993.

[159] Mary Beth Rosson, John M. Carroll, and Rachel K. E. Bellamy. Smalltalk scaffolding: A case study of minimalist instruction. In *Proceedings of ACM CHI'90 Conference on Human Factors in Computing Systems*, Aids to Understanding Programs, pages 423–429, 1990.

[160] Steven F. Roth, Mei C. Chuah, Stephan Kerpedjiev, John A. Kolojejchick, and Peter Lucas. Towards an information visualization workspace: Combining multiple means of expression. *Human-Computer Interaction*, 12(1-2):131–185, 1997.

[161] Daniel M. Russell, Mark J. Stefik, Peter Pirolli, and Stuart K. Card. The cost structure of sensemaking. In *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems*, Conceptual Analysis of Users and Activity, pages 269–276, 1993.

[162] Gerard Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer.* Addison-Wesley, Reading, MA, 1989.

[163] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *JASIS*, 41(4):288–297, 1990.

[164] Tefko Saracevic, Paul Kantor, Alice Y. Chamis, and Donna Trivison. A study of information seeking and retrieving. i. background and methodology. *Journal of the American Society for Information Science*, 39(3):161–176, 1988.

[165] Upendra Sharanand and Pattie Maes. Social information filtering: Algorithms for automating word of mouth. In *Proceedings of ACM/CHI 95*, Denver, CO, 1995.

[166] Frank M. Shipman, III, Catherine C. Marshall, and Thomas P. Moran. Finding and using implicit structure in human-organized spatial layouts of information. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 346–353, 1995.

[167] Ben Shneiderman. The eyes have it: A task by data type taxonomy. In *Proceedings of Visual Languages 96*, Boulder, CO, 1996.

[168] Ben Shneiderman. *Designing the user interface: strategies for effective human-computer interaction.* Addison-Wesley, Reading, MA, 1997.

[169] Ben Shneiderman, Donald Byrd, and W. Bruce Croft. Sorting out searching: A user-interface framework for text searches. *Communications of the ACM*, 41(4):95–98, 1998.

[170] Anselm Spoerri. InfoCrystal: A visual tool for information retrieval & management. In *Proceedings of Information Knowledge and Management '93*, Washington, D.C., Nov 1993.

[171] Craig Stanfill and Brewster Kahle. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[172] Tomek Strzalkowski, editor. *Natural Language Information Retrieval.* Kluwer Academic Publishers, 1999. To appear.

[173] Elaine Svenonius. Unanswered questions in the design of controlled vocabularies. *Journal of the American Society for Information Science*, 37(5):331–340, 1986.

[174] P. Thompson, H. Turtle, B. Yang, and J. Flood. Trec-3 ad-hoc retrieval and routing experiments using the win system. In *Proceedings of the Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, 1995.

[175] R. H. Thompson and B. W. Croft. Support for browsing in an intelligent text retrieval system. *International Journal of Man [sic] -Machine Studies*, 30(6):639–668, 1989.

[176] Anthony Tomasic, Luis Gravano, Calvin Lue, Peter Schwarz, and Laura Haas. Data structures for efficient broker implementation. *ACM Transactions on Information Systems*, 15(3):223–253, 1997.

[177] Geoffrey Towell, Ellen M. Voorhees, Narendra K. Gupta, and Ben Johnson-Laird. Learning collection fusion strategies for information retrieval. In *Proc. 12th International Conference on Machine Learning*, pages 540–548. Morgan Kaufmann, 1995.

[178] Edward Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Chelshire, CT, 1983.

[179] Edward Tufte. *Envisioning Information.* Graphics Press, Chelshire, CT, 1990.

[180] Howard Turtle and W. Bruce Croft. Inference networks for document retrieval. In *Proceedings of the 13th Annual International ACM/SIGIR Conference*, pages 1–21, 1990.

[181] L. A. Tweedie, R. Spence, D. Williams, and R. Bhogal. The attribute explorer. In *Proceedings of the Video Track of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Boston, MA, April 1994. ACM.

[182] Marilyn A. Walker, Jeanne Fromer, Giuseppe Di Fabbrizio, Craig Mestel, and Don Hindle. What can I say: Evaluating a spoken language interface to email. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, Los Angeles, CA, March 1998. ACM.

[183] John A. Waterworth and Mark H. Chignell. A model of information exploration. *Hypermedia*, 3(1):35–58, 1991.

[184] H. D. White and K. W. Mccain. Bibliometrics. *Annual Review of Information Science and Technology*, 24:119–186, 1989.

[185] Robert Wilensky, Yigal Arens, and David N. Chin. Talking to UNIX in English: An overview of UC. *Communications of the ACM*, 27(6), 1984.

[186] Peter Willett. Recent trends in hierarchical document clustering: A critical review. *Information Processing and Management*, 24(5):577–597, 1988.

[187] Michael David Williams. What makes rabbit run? *International Journal of Man-Machine Studies*, 21(4):333–352, 1984.

[188] James A. Wise, James J. Thomas, Kelly Pennock, David Lantrip, Marc Pottier, and Anne Schur. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. In *Proceedings of the Information Visualization Symposium 95*, pages 51–58. IEEE Computer Society Press, 1995.

[189] Ian H. Witten, Craig Nevill-Manning, Roger McNab, and Sally Jo Cunningham. A public library based on full-text retrieval. *Communications of the ACM*, 41(4):71–75, 1998.

[190] Kent Wittenburg and Eric Sigman. Integration of browsing, searching, and filtering in an applet for web information access. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, Late Breaking Track*. ACM, 1997.

[191] Degi Young and Ben Shneiderman. A graphical filter/flow model for Boolean queries: An implementation and experiment. *Journal of the American Society for Information Science*, 44(6):327–339, July 1993.