

Integrating Search and Retrieval with Hypertext

By Edward A. Fox, Qi Fan Chen,
and Robert K. France
Dept. of Computer Science,
Virginia Polytechnic Institute and
State University

Introduction

While hypertext and hypermedia have numerous applications, they cannot solve all our problems relating to information access. Such access can build upon manually or automatically created indices, search algorithms, feedback procedures, and other techniques whose value has been proven on collections much larger than those that can be managed solely by browsing and following links. We propose a reference model for what we believe is an urgent need, that of integrating search and retrieval with hypertext and hypermedia. Further, we describe a prototype system designed around that model which would facilitate access to a wide variety of types of information.

Current Status

In this chapter, we focus on the problem of how to efficiently store and retrieve a wide range of disparate information (including data stored in databases, text bases, media archives, knowledge bases, and other organized collections). Once such col-

lections were managed almost exclusively by mainframes, which ensured centralization and often enforced vendor, and sometimes industry-wide, standards. With the shift to personal computers and workstations, there has been a rapid proliferation of specialized packages, which make interchange of information between software and hardware platforms difficult.

Accumulations of information in forms that are hard to integrate are being created and expanded at a rapid rate. Large online database vendors such as Dialog, Maxwell Online, STN, and others provide access to enormous collections of bibliographic records (usually titles, abstracts, and other citation data) or full text (reference works, newspapers, legal documents). In house, corporate employees are generating electronic mail messages, database files (separately organized for geographic information, CAD/CAM, chemical registry data, and image databanks), and word processing documents. The serious problems of providing democratic access to all this information and of supporting manipulation techniques as versatile and natural as speech or pencil-and-paper methods have yet to be solved.

Prospects for Improvement

The emergence of networked computer systems and optical storage devices have exacerbated the problem of incompatible data formats and, as a result, forced vendors to seek solutions. For example, software designed to access data scattered across a heterogeneous network needs to follow rigorously enforced conventions.

Optical publishing systems increase the amount of storage capacity available at a reasonable cost. This allows in-house access to huge databases of text, graphics, images, and even interactive digital video [Fox, 1988d, 1989d, and 1990a]. The huge amount of information that can be stored optically can make particular facts difficult to find. Fortunately, locally stored information can be organized and more easily accessed with more modern software than is provided with on-line services such as Dialog. This allows more advanced information retrieval techniques to be implemented [Fox, 1986a].

Our work is motivated by an analysis of current potentials and problems with information access. In the first part of this chapter, we describe a framework for integrating hypertext systems with techniques for discovering information quickly, leading readers to particular nodes efficiently even in very large, unfamiliar hyperbases. In the second section, we consider some of the capabilities that must be provided by information systems, pointing out which are provided by various types of existing systems. The third section discusses progress made in that direction and its sequel provides further background. The fourth section discusses the VPI&SU CODER project, which provides a means of exploring a broad class of information access approaches. The final section of the chapter outlines how we are extending CODER for hypermedia and interoperability, and includes recommendations for conceptual and practical integration of information retrieval with hypertext/hypermedia.

Necessary Capabilities

Computer systems which are intended to facilitate user's access to information should:

- Include help systems that can explain how to perform a search and what is done as part of that search;
- Adapt to user requirements;
- Help users describe and elaborate their information needs;
- Be built in a flexible and modular fashion so that they can be easily extended. [Belkin, et al., 1983; Belkin, et al., 1987a]

A user-centered, mixed initiative interaction (where both user and computer operate concurrently and can each seize control of input or output activities at almost any time) is preferred, so that:

- the user can request an explanation or follow a new train of thought at any time,
- the computer can request clarification or display tentative results as appropriate to further what it believes are the user's current goals.

Since these goals often relate to finding relevant items in some information repository, it is important to pay special attention to efficient and effective searching, as is discussed in the next subsection.

Finding Barn Doors and Stringing Pearls

The current generation of information retrieval systems rely on various search techniques to find relevant items. These can be viewed at both the abstract and system-specific levels [Bates, 1981]. One approach is to seek information by first looking for a neighborhood where useful items can be found and then selecting related or linked items from among the items at hand.

One can think of first finding the *barn door* and then, after glimpsing something of interest, i.e., a valuable *pearl*, following leads to closely related items. This is analogous to searching and then browsing, or using information retrieval techniques and then applying hypertext methods.

Most information being accessed has been processed to some degree in order to make access easier. Outlines, tables of contents, and indices (e.g., for author names, or specially selected descriptors) are usually developed by humans according to some theory or organizing principles. Readers who can relate to existing organizational systems can often find items of interest; computer menus are sometimes applied to map these structures to available regions of display screens. Related items may be found if they occur nearby, if an index points to a series of items, or if explicit cross references are provided.

Unfortunately, the searcher's perspective and/or vocabulary does not always match that of the author or indexer. This is particularly important when a complex idea is involved and when a very large collection of items is being searched. Here pre-coordinated systems (where some of the most important combinations of ideas are pre-combined and added to indices, sometimes as phrases like "information retrieval") may fail and force readers to build post-coordinated queries (like "multimedia and retrieval"). Narrower queries can be formed using proximity operators. These include:

- searches for adjacent words (two words placed side by side);
- two or more words located in a single sentence;
- within field combinations (e.g., two words located within a title line);
- within a paragraph.

Boolean query construction (i.e., using ANDs, ORs and/or NOTs to specify matches) can be exceedingly difficult, and indeed ten-to-one variations have been observed in the ability of different individuals (and even of the same individual at different times) to form good queries. Frequently, repeated efforts are required to form queries that do not retrieve many nonrelevant items (i.e., have high precision) and yet do retrieve a significant fraction of the relevant items (i.e., have high recall).

Finding relevant items often involves one or more of the following steps:

- Translating an information need into the organization of concepts represented by a table of contents, index or thesaurus.
- Adopting a scheme to descend through some hierarchical organization of topics or descriptors to find specific section(s) of interest.
- Combining words or terms used within an index or thesaurus or forming the free text into a formal Boolean query.
- Refining the query by adding synonyms or related terms (broadening), by increasing specificity (narrowing) or by reorganizing the query structure.
- Visually examining all selected items to find those of value.

Sometimes the task at hand makes the choice of search method obvious. If you are looking for a single known item, a Boolean search will work best. Looking for multiple items on the same topic calls for browsing. Attempts to retrace a previously viewed presentation suggests the use of links. Sometimes a combination of methods works best.

Unified Metaphors

The steps involved in searching and browsing hyperdocuments should be hidden behind a task-oriented interface that lets readers search at a conceptual, descriptive level instead of at either a word-oriented or a procedural level. Boolean queries, for instance, fail to fulfill readers' information access needs because they focus

attention on the wrong objects and force users to think in terms of the mechanics of search algorithms. Conversely, properly implemented browsing systems allow readers to explore a range of conceptual dimensions easily and intuitively. A graphical presentation coupling the spatial metaphor of nearness to the hypermedia metaphor of links supports such explorations using simple pointing operations. Most importantly, both searching and browsing should be subsumed by a set of consistent metaphors and abstractions that encourage the integration of these access techniques.

Using naturally occurring metaphors as the backbone of the user interface offers several advantages. Unfortunately, while paper sketches and natural language input may be easy for humans to work with, they are often beyond the scope of all but the most expensive of today's computer systems.

Another possibility is to have a standardized unified digital representation and manipulation system. We all take for granted the fact that telephones connect people all around the world, and that they do this because of international agreements on hardware and software standards. We can envision a collection of standards for hardware, software, and information structures or data formats [see Devlin, 1991] built around both existing and exciting new technologies for digital encoding. This requires management of multimedia including: text, graphics, images, speech, audio, and video. It involves interconnection of multiple computers and diverse networks, and use of a variety of peripheral and input/output devices. Finally, it requires *intelligent* processing to improve efficiency and effectiveness and to ensure smooth human-computer interaction.

Progress Toward Unified Access

Networking

A great deal of progress has been made towards integration of computer networks [Tannenbaum, 1981]. Thousands of computers are internetworked, able to exchange electronic mail. Many are connected to allow even higher levels of integration: passing of files, remote log-ins, and even communication between programs. In the area of library information retrieval, the Z39.50 standard has been developed so that a user of one library system can cause that system to have a query processed on another system, and then indirectly receive the search results. Various organizations, including universities such as VPI&SU, are developing and testing prototype systems for information interchange that follow Z39.50 specifications. In addition, the X.500 standard has enabled many prototype systems to combine into a partial global directory system—a computer analog of the telephone white pages.

Much of the success in internetworking can be credited to the use of a layered approach to software, whereby physical connection, data connection, session management, presentation handling, and application program operation have separate interfaces and specifications. The *International Organization for Standardization*

(ISO) *Open Systems Integration* (OSI) model uses physical, data link, network, transport, session, presentation, and application levels to encourage interoperability of software on heterogeneous computers [Zimmerman, 1980]. OSI has been the basis for Z39.50, X.500, and many other related standards.

This layering approach also applies to storage systems. On the one hand, there are levels of physical units, operating system-identified devices, and network wide file systems. On the other hand, there are software divisions, such as the proposed standard of the Air Traffic and Air Transport Associations, which separates the search engine component of CD-ROM software (which itself builds upon the volume and file description layer specified by the ISO 9660 standard) from the user interface layer.

Object Model

While computer users generally search for relevant concepts, computer scientists have tried to meet user's needs by studying and building specialized processing systems for particular classes of information. Thus, the term "data" is often associated with database management systems, "information retrieval" is often limited to text databases, and "knowledge bases" are viewed as the proper repositories for collections of complex list structures and rules.

To unify these approaches, object-oriented databases have been proposed that might contain and support processing of any class of information. In hypertext systems that build upon these object bases, completing a search or following a link would lead to presentation of any of various objects: a screen of text, a bitmap display, an audio segment, a spreadsheet entry, an individual plane of a complex image, an explanation of some expert rules, or a tabular report.

However, to date only limited progress has been made in integrating database management systems (which at present usually follow the relational model) with text retrieval systems (which frequently employ inverted files and require access by way of Boolean queries). One issue is the differences in items being manipulated; tables of numbers or short character strings have few data types as opposed to the variety encountered in multimedia systems, and database structures are much more regular than those found in the world of compound documents (e.g., books, magazines, letters, reports, and encyclopedias). A possible solution is to use abstract data types (e.g., sets, lists, vectors) as elements of a relational DBMS [Fox, 1981]. Some of these ideas were used in the redesign of the SMART system for the UNIX environment [Fox, 1983a], which eventually led to a version of that system that is widely used by document retrieval researchers. Another approach is to directly use the relational model, with performance tuning and limited extensions where needed, to handle bibliographic records [Lynch, 1987].

Artificial intelligence (AI) research extends these efforts to solving the problems of knowledge representation. Initially, languages like LISP and Prolog focused on

symbol and list manipulation. Real world objects were associated with *atoms* that could be located by their names. Property lists (sets of property type/value pairs) were attached to these atoms.

But to better match human ability to deal with default values, groupings of key attributes with various types of objects, and taxonomic classification of similar classes of objects, the concept of a *frame* was developed. This concept has proven useful for information retrieval [Fox, 1987b, Weaver, et al., 1989].

Essentially, a frame class (e.g., U.S. postal address) has various slots for important aspects of the object (e.g., state, zip code), and may inherit from more general classes (e.g., postal address with slots for name and locality), as well as be instantiated for an individual (e.g., John Smith's U.S. postal address). Frames can be grouped together into extremely regular knowledge bases.

A semantic network is an alternate knowledge representation, closer in orientation to hypertext [Findler, 1979, Morgado, 1986]. A semantic network can be envisioned as a graph where nodes are used to represent objects, and links to indicate significant, meaningful relationships. One well-known semantic network system, SNePS [Shapiro, 1989], supports knowledge representation and inference (i.e., proving or drawing conclusions based on knowledge recorded previously in the system). In SNePS a distinction is often made between *path-based inference*, whereby the reader follows a succession or chain of links from some node to identify some relationship, and *node-based inference*, whereby logical propositions are represented in the network and are combined with other propositions in varied regions of the network in accord with the rules of logic. Path-based inference can be very efficient, and corresponds to following links in hypertexts. Semantic networks can also support *spreading activation*, where several concepts are located in the network, all paths of length 1, 2, etc. from each are explored until paths from several concepts meet—a type of parallel search for important associations. Spreading activation has been used in the GRANT system to retrieve documents related to various topics, projects, and grant proposals [Cohen and Kjeldsen, 1987].

Semantic networks are useful for handling the inter-relationships in language. At VPI&SU, for example, we have taken machine-readable dictionaries, extracted and restructured the important data [Fox, 1986b], archived them in the form of Prolog facts, and loaded them into a semantic network to facilitate further processing and access [Fox, 1988a; France, et al., 1989a]. In so doing we have found it helpful to build an elaborate taxonomy of relations (i.e., links) common between word senses in our lexicon. One goal is to extend earlier experiments that indicated improvement in retrieval effectiveness could result by finding words that are lexically related to query terms [Fox, 1988e]. Thus, the system knows enough about the meaning of a query to search for "captain" if a query asks for "ship's officers," or "general" when "army leader" is sought. Others have considered networks as important support structures for probabilistic (Bayesian) inference systems that prove when a document is relevant to a query [Croft and Turtle, 1989]. Ultimately, semantic networks can be combined with hypertext and hypermedia to lead to a uniform representation of objects, as discussed below.

Integration of Various Search Techniques

Research suggests that the best retrieval performance results when many different forms of information about documents are utilized by a variety of search methods [Belkin and Croft, 1987b]. Integration of various approaches has this same goal [Fox, 1987b]. Thus, one might merge the results from using Boolean, vector, and probabilistic searches for a given information statement. Any of these search methods can also be modified by the use of feedback, where a reader indicates what documents or sections thereof are relevant, and the computer uses all the information it has available about that sample to perform another, better search. One can even throw in use of AI techniques, as discussed in the section on our work with CODER. Reference models provide a clear framework for integration and thus have become popular in regard to networking, where different cabling, interconnection, specialized hardware, firmware, and layers of software allow users of similar applications on different computers to collaborate.

Reference Models for Hypertext

In order to develop interoperable information systems and a unified representation model, a reference model for information management systems must first be developed. Work on hypertext reference models has arisen in part as a result of standardization efforts for hypertext/hypermedia [see Chapter 26 by Devlin on standards]. We have found the Dexter model [Halasz and Schwartz, 1990] and the *r-model* [Furuta and Stotts, 1989; Furuta and Stotts, 1990; Stotts and Furuta, 1989] to be particularly insightful. However, interoperable information systems should properly include not only hypertext/hypermedia, but also searching, networking, and other applications and levels of manipulation.

A More Complete Reference Model

Our proposal for a reference model that integrates hypertext and other sources of information is shown in Figure 21.1, and explained in more detail in subsequent sections. Essentially it combines the seven OSI layers with seven other layers relating to hypertext/hypermedia and other types of information systems.

The bottom four layers are OSI layers that together support the secure transport of messages. Atop the transport layer is a layer that provides essential support for files and process communication (messages). Processes operate above this layer to support high-level communication between machines, including necessary translations between data representations. These six layers complete an extended groundwork for communication among machines, languages, and environments at a highly abstract level.

The node layer comes next, supporting atomic, structured, and multimedia objects. The anchor layer allows points or spans inside nodes to be addressed in ways appropriate for each node type. Links connect anchors, thus providing a

Typical Contents

Layer	
Application	hypertext, hypermedia, information retrieval, image management, authoring, training, tutoring, CAD/CAM, interactive digital video
Presentation	devices: windows, pointing devices, speakers; media: text, audio, video; operations: generalization, editing, translation, access, browsing, selection, picking, querying, sequencing.
Session	base selection, user identification, history management, versioning.
Base	base: knowledge, information, data; operations: search, inference.
View	graphs, lists, sets, vectors, relations, frames.
Link	link Ids, labels.
Anchor	anchor Ids, span descriptions.
Node	textual strings, integers, object Ids.
Communication	processes.
Physical machine	file systems, storage media, message support.
Transport	ISO lower level equivalent
Network	ISO lower level equivalent
DataLink	ISO lower level equivalent
Physical	ISO lower level equivalent

Figure 21.1 Information Management System Reference Model.

layer that is essentially a graph or network of information items. The view layer allows various aggregations and associations as well as common data and information and knowledge structures to be manipulated (e.g., have a view of relations, a view of vectors, a view of frames, a view as linked collections of nodes). The base layer allows coordinated access to data, information and knowledge bases—through search, navigation, and related operations. Higher levels correspond to upper level OSI layers, but are organized to support common information access, hypertext/hypermedia, and presentation/application programming activities.

We view this model as providing a framework for integrating information management systems. Our aim is to validate this framework, to prove it adequate for a wide variety of activities. Towards the end of this chapter, we explain how our work with the CODER system is being extended and layered along the lines of this integrated model.

Previous Work

In the broadest sense, a reference model for information management should encompass work in electronic publishing, document representation, text analysis, storage, information retrieval, and hypertext/hypermedia.

Electronic Publishing

In connection with ACM Press Database and Electronic Products [Fox, 1988c], we have ventured into a variety of electronic publishing ventures [see also Fox, Rous, Marchionini, 1991]. In *Hypertext on Hypertext* we dealt with three versions of a single hypertext, in order to accommodate user preferences for equipment and operating systems. Further, in one of the versions [Shneiderman, 1988], the original hypertext system was extended to include more comprehensive searching capabilities, in part because of our experiences with online and CD-ROM publishing. These experiences also pointed out the need for standards and integrated or interoperable systems to allow wide-scale electronic publishing [Fox, 1990b].

For example, while developing the Virginia Disc series of CD-ROMs, we purposely provided data and information bases in multiple forms to allow comparison of approaches and software packages. The *Master Gardener Handbook* on Virginia Disc One [Fox, 1990c] is available in hypertext form, or can be searched with a bibliographic retrieval system. Though the former is handy for reading and browsing, the latter is more convenient for finding specific facts. Similar comparisons can be made between alternative retrieval systems (Personal Librarian applied to searching the *King James Bible* at either the chapter or verse level).

Related Work

The most popular database management software today—relational DBMS—is based on the simple mathematical concept of a *relation*. A database relation can be thought of as a flat table, but is much more precisely defined—a relation is a set of individual items, called “tuples” and sometimes thought of as rows, each containing values for a fixed collection of attributes, usually associated with columns of the table. Queries in the relational model are answered by combining these relations using a restricted set of operations that depend on the flat and uniform structure of the tuples. Traditional information retrieval also uses the notion of sets [Artandi, 1971], but regards each item as a nonuniform—and often

structured—collection of attributes. Information retrieval queries combine desired attributes using sentence-forming operators. Thus, a Boolean query specifies a set of documents by combining simpler queries (ultimately, individual indexed words or concepts) using logical operators.

Other systems of operators are possible. For instance, significantly more effective retrieval results from a simple change related to human perception of language and Boolean logic—making the *AND* and *OR* operators softer or less strict. One realization of this change, the p-norm model, allows a parameter, p , to vary so there is a continuum between strict *AND*s and strict *OR*s [Salton, et al., 1983]. This model allows retrieval results to be ranked in descending order of presumed value, and gives users flexibility to indicate relative importance of search terms. Further extension is possible, whereby user judgments regarding which previously retrieved documents are relevant can be used to automatically build better Boolean or p-norm queries [Salton, et al., 1985].

Feedback is also fundamental to the probabilistic model [Robertson and Jones, 1976], whereby documents are also ranked according to estimated probability of relevance. Ranking from probabilistic retrieval and various fuzzy set-based schemes [Bookstein, 1985] is of particular value to users so they need not be concerned with retrieval set size, but rather can let the computer carry out pattern matches to find closely related items.

Another enhancement to retrieval results from adding information. This has been demonstrated using the vector space model, where linear algebra can be used to allow descriptions of term occurrences in document collections [Salton, 1989]. Many people are familiar with keyword-based search approaches [called syntactic searches in Littleford, 1991], in which the reader searches for a particular string of characters in a database or uses entries from a *controlled* vocabulary for searching. Keyword-based searches are usually only available as alternatives to rather than complements to citation-based searching (e.g., as is done with the *Science Citation Index*). Yet if term and citation data are both made available [Fox, 1983b], especially when used in feedback searching, better results can be obtained than from either type of data alone [Fox, 1988b]. In the vector approach, readers select documents or groups of documents which can be treated as queries, so the system can search for these matches.

Results with these advanced retrieval methods have been demonstrated with collections of up to tens of thousands of documents. A pilot study indicated applicability to roughly 300,000 documents [Fox, 1987d], and results are now being analyzed from a recent study at VPI&SU with 500,000 library catalog records and 216 users, that compared Boolean, p-norm, vector, and vector with probabilistic feedback approaches. It seems that advanced methods for retrieval can be applied to very large collections.

CODER

Work on the *COMposite Document Expert/extended/effective Retrieval (CODER)* system began in 1985, to develop a testbed for the integration of AI and informa-

tion retrieval methods [Fox, 1986a]. The aim was to develop a next-generation retrieval system: a system that would have greater effectiveness and efficiency than previous systems, and one that would allow comparison and integration of advanced retrieval methods. Our plan is to integrate the various retrieval models into a unified system [Fox, 1989a].

CODER Architecture

As can be seen in Figure 21.2, CODER was conceived as a comprehensive standalone system that could deal directly with end users and could receive texts of bibliographic and full text documents ready for analysis. The architecture follows principles of software engineering, AI prototyping, and object oriented construction [Fox, 1987a].

CODER assumes a coarse-grained parallel operation of the sort now provided by computer networks. A central *spine* holds information for shared use by the analysis and retrieval subsystems. Each subsystem is specified as a collection of modules acting as a *community* and communicating through a central blackboard (a module with *areas* that experts can post to or read postings from). Attached to each blackboard is a strategist, with rules to indicate which modules should be requested to examine which types of postings.

The retrieval subsystem is broken into modules along functional lines, based on studies of user and search intermediary interaction [Belkin, et al., 1983], as a "distributed expert based information system" [Belkin, et al., 1987a]. Thus, there are modules for user modeling, dialog and discourse management, search, planning, morphological analysis, and other operations. These can operate on a single computer, or can be spread across a network of machines on which they run in parallel.

The spine stores document and other text collections, relational data, and more complex knowledge structures. Frames are used extensively to describe documents, real-world entities, sessions, and system users. A lexicon holds English language knowledge derived initially from the *Collins English Dictionary* [Fox, 1986b] and later extended from other sources [France, et al., 1989a]. Domain knowledge suitable for each document collection processed is also stored. For example, the spine includes knowledge about ships and geographical locations for an application involving analyzing and retrieving Navy messages.

CODER Implementation

CODER has evolved from a rough prototype to a system that now shows promise as the basis for a campus information system at VPI&SU. Initial development was keyed to a modest collection of electronic digests about AI, a simple menu style interface, and facts derived from the *Handbook on Artificial Intelligence* [Fox, 1987c]. In this version, names of leading AI researchers were identified, words

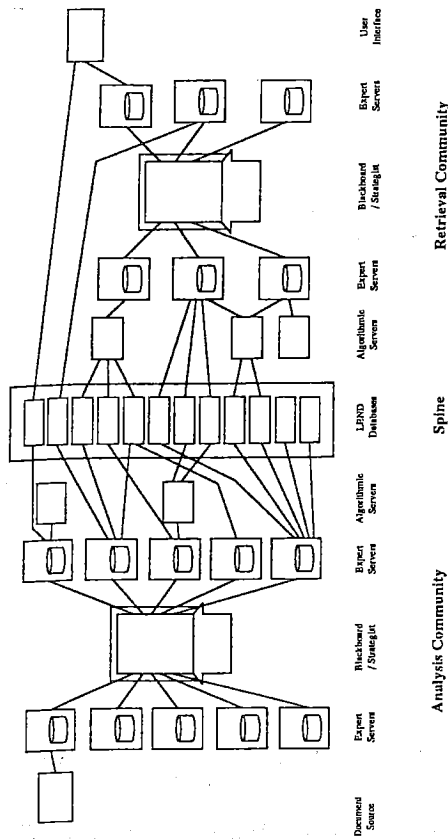


Figure 21.2 CODER system architecture.

associated with their definitions were given in the running text of the book, and a hierarchical taxonomy of AI areas correlated with sections of the text.

Successive applications include new fact bases dealing with ships and geographic locations, a medical thesaurus, and more extensive user interfaces. A more complete language for intermodule communication was developed, so modules written in Prolog, C, and C++ could send messages between computers, including frames, sets, lists, and relations [Weaver, et al., 1989].

Various retrieval techniques have been incorporated, and ultimately CODER should operate by applying several advanced methods [Belkin and Croft, 1987b] in concert or separately on a case-by-case, heuristically determined basis.

CODER/LEND

In extending CODER to handle the results from analyzing multiple machine-readable dictionaries, and to support large document collections, we recognized the benefits of integrating the relevant facts, relationships, and frame structures into a large knowledge base. Since lexical information is so highly inter-related, we chose a semantic network type of representation. However, existing semantic network systems were designed to work with the entire network in primary memory. Thus, we conceived of the *Large External Network Database (LEND)*.

LEND has been under development since 1988, with the aim of providing integrated storage, access, and manipulation for data, information, and knowledge. Specifications have been prepared [France, 1989b; Chen, 1989], and the second implementation is now partially complete. A version of LEND is in use in connection with a campus directory system based on the X.500 standard. However, most work with LEND relates to CODER.

LEND has been designed to provide efficient access to very large knowledge bases. In part this has been made possible by our work with hashing functions. In particular, we have developed very fast algorithms for minimal perfect hashing. Thus, given a static collection of perhaps several million keys (e.g., people's names or social security numbers), we can identify a function that will take any key as a character string, and return the unique address assigned to that key in a hash table which has no wasted space. While most hashing assumes a moderate level of update and deletion, and while a good deal of recent work on hashing focuses on dynamic data sets [Enbody and Du, 1988], our focus has been on static or slowly changing data. Thus, we have demonstrated our methods with CD-ROM [Fox, 1990c] and in connection with LEND. In particular, we have worked with minimal perfect hash functions (where the hash table is fully loaded, and all searches for values in the table take one access) [Fox, 1989b, Fox, 1989c]. Our latest techniques give very close to optimal performance and can find these functions for large key sets very rapidly [Fox, 1990d]. The net effect is to have very rapid access to any object in the network, with lower space requirements and faster performance than those of other common secondary storage access mechanisms [Comer, 1979].

CODER together with LEND now supports analysis and searching for several types of collections. Some 70M bytes of network data has been loaded from dictionaries and document sets into LEND. We aim to demonstrate the utility of this combination through experimentation with sections of a large hypertext reference work dealing with cardiology, a medical thesaurus, our English dictionary data, and hundreds of thousands of bibliographic records.

Beyond these practical issues, however, LEND is the embodiment of our new theory about integrated information access. Though initially building upon concepts of knowledge representation in semantic networks, LEND was extended to also handle hypertext/hypermedia. Indeed, the theory underlying our present work with LEND is that integrated representation of all types of information, including numbers, text, and frames—as objects in a network—should be the foundation for efficient, flexible, and powerful intelligent retrieval systems. As we further combine CODER and LEND, the result should be an integrated information management system, usable for document analysis, query processing, and hypertext.

Recommendations for Integration

In recent years, object-oriented methods have become popular for programming and object-oriented databases have been shown to be useful for multimedia [Kim and Lochovsky, 1989]. Object typing is the basis for handling these varied sorts of objects.

Typed Objects

Our reference model begins with a layer of objects. We assume that objects are typed, that is, that text objects can be differentiated from integers or video seg-

ments and treated appropriately. Appropriate treatment is generally defined algebraically, by specifying the operators that function within a type. Thus, integers come equipped with addition, multiplication, and so forth, while text objects come with operators for concatenation and a test for inclusion. The type system should enforce against such meaningless operations as concatenating two integers or multiplying texts.

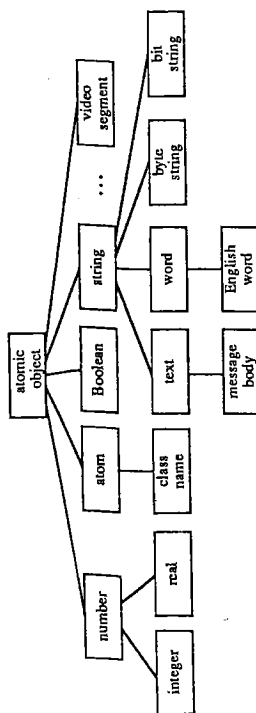
Typing schemes play an important role in many different programming disciplines and representational systems. Classical programming languages are based on flat typing systems, where no relations are implied among the built-in types. The more modern of these include both primitive, atomic types (e.g., integers, reals, and atoms) and composite types (e.g., records, lists, and sets). Type conversion functions may provide morphisms from objects of one type to objects of another. Current work in programming language theory includes extending typing systems to include function types and polymorphous types for composite objects.

Type declarations in programs are quite similar to definitions of *object classes* in object-oriented databases. One defines broad classes, like numbers which have operations like addition, and then narrower classes, like integers, which inherit properties from the broader or parent class. This allows a taxonomy of objects to be developed, with programming to specify operations of new objects limited to those (e.g., next in sequence for integers) not present in previously defined classes. Each class comprises a certain type of object, together with the operations of methods that are appropriate to the objects of that type. Since objects only respond to their appropriate methods, meaningless operations are not permitted. Furthermore, use of this system makes it possible to model objects directly, to support fast prototyping, and to group types with similar methods together, defining structure and operation as it becomes appropriate (See Figure 21.3.).

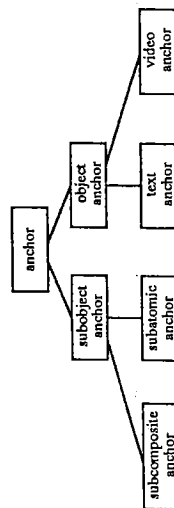
Integrated information systems require a significantly richer set of atomic types than do typical applications in either the classical or object oriented paradigms. Not only is text required, but graphics and images are becoming commonplace in documents, and we are at the threshold of a revolution in handling digital audio and video [Fox, 1989d; Fox, 1990a]. Additionally, information objects like bibliographic records or dictionary entries are best modeled by composite objects with a rich internal structure. For document objects, there are obvious part/whole and subtype/supertype hierarchies implicit in document structure [Guting, et al., 1989] and other hierarchies inherent in media objects. Multiple inheritance hierarchies are useful in this context, since information objects often satisfy more than one description simultaneously. For example, a document that is at once an electronic mail (e-mail) message and a set of bibliographic references inherits attributes from both the e-mail message type (sender, address) and the bibliography type (subject domain, record format).

anchors and Links

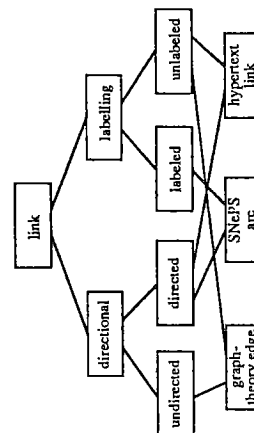
One innovation of hypermedia practice has been the use of *anchors*. When two objects are linked in a hypertext system, the link runs not between entire objects



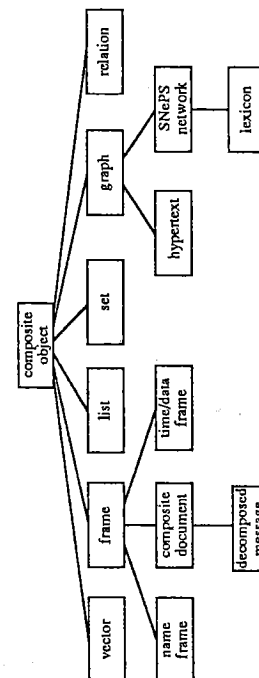
(a)



(b)



(c)



(d)

but from a hot spot in one object to a relevant section in the other. We can formalize this idea and extend it to other classes of objects through the concept of an anchor. An anchor is a segment within an object, where "segment" and "within" are given meaning by the object type. Thus, an anchor in a CAD drawing might attach the key to the diagram, or to the sketch of the left front door of a car (regardless of whether an animation shows the door opening and closing), or a button shown on the controls of a computerized jukebox that allows audio selection or rewind or play.

In the anchor layer, each class of objects that support anchors is provided with methods for creating and resolving anchors; at all higher layers, anchors are abstractions that can only be interpreted with reference to these methods. In many cases, however, operations are further specialized. If a link is unidirectional, then activating an anchor at the start of the link leads to the link itself, while activating the link and thence the anchor at the other end of the link usually leads to presentation of that target anchor and the node it is attached to.

An anchor serves as the terminating point for links, which are associations between active areas within objects. (Where the entire object is intended, the object as a whole constitutes an anchor.) Links can be of many types, depending upon their intended use (see Figure 21.3c). Within each type, links can be further differentiated, for instance by the type of the objects linked, or by label. In semantic networks, links are generally differentiated by their labels. As in the classical programming language approach, however, these implied link types form a flat domain. We suggest an enrichment to allow a hierarchy of link types. This allows the same sort of typing system to be used with links that was used with objects, and also follows earlier work in hypertext, where many link types were found appropriate [Trigg and Weiser, 1986], and where a link taxonomy was described [DeRose, 1989].

Anchors allow reference to objects of parts thereof. Links can use anchors to refer to an entire node, to a portion of an atomic node, or to one of the components of a node collection. As a result, a semantic interpretation can be imposed even in an interoperable system for resolving links [Pearl, 1989].

In information retrieval the concept of nearness between document objects becomes important. Any such measure of proximity must effectively be based upon links between words and their exact appearance in the text. Thus, proximity in an index for this chapter would be equivalent to a pointer to the seventeenth word of this paragraph. With an anchor for that word occurrence, a search could easily lead to the target word, or some surrounding context like sentence or paragraph that might be displayed with highlighting.

With nodes, anchors, and links—the heart of a hyperbase—we have also the key elements for managing other graph-like systems. Graphs are very handy for modeling, computer networks, traffic routes, or lexicons of word associations. But other views of graphs can be built so that manipulations are tailored; such as when head and tail operations are applied to ordered lists. Systems that support defining such views can allow us to handle composite documents, complex knowledge representations, and other perspectives on our data.

Figure 21.3 Type hierarchies for (a) node layer (b) anchor layer (c) link layer (d) view layer.

Views and the Pattern Language

Links can be used to express simple associations between objects. Thus, one type of link might associate a commentary with the text it refers to, while another link might associate a person's name and digitized photograph. Links can associate spreadsheets with totals in a report or words with their definitions.

Often, however, the relations between objects are more complex. For instance, the link between a document and a concept that occurs within it can include a weight denoting how important the concept is to the document. Furthermore, it is often desirable to consider a relation as an object itself. For example, one might want to search to find all the types of links or all links between text and image nodes.

The SNePS semantic network system [Shapiro, 1979] has the means to build such complex objects on a substrate of associative links. In particular though, this capability is usually used to manage short texts for natural language processing, or to describe facts that might otherwise be stated as first-order predicate calculus terms. Similar techniques can be used to model lists, sets, or frames. Frames in turn provide a natural representation tool for composite documents and media objects. In this way, vivid representation systems for a given domain are built from more general tools. We refer to these constructed classes as *view types*.

Even at the view level, a full type system with multiple inheritance is supported. The inheritance hierarchies are more complex, since different perspectives call for different structures and sets of operations, but the typing system itself can be as simple as that used at lower layers.

Views are made of up of classes of view objects. Each such class is created by specifying a mapping of the composite objects in the class into associative structures on the graph level, and by defining the methods for the class in terms of graph operations. This is accomplished in LEND by a pattern language. Related work has been done in connection with AUGMENT [Engelbart, 1962], with graphic queries and a graph language [Consens and Mendelzon, 1989], with taxonomies of graphs for hypermedia [Parunak, 1989], and with paths or trails in hyperbases [Marshall and Irish, 1989; Zellweger, 1989]. The pattern language is used to carry out associative access to objects related by links. Such access patterns are best characterized by following a path of a given or indeterminate length, with possible restrictions on the nodes and arcs that may occur in the path. Our focus in LEND is on efficient access of these basic operations.

LEND operates in two ways. The first is to help carry out typical browsing and searching tasks. LEND automatically maps these tasks into a series of operations it carries out on the underlying graph of information items. In a sense, views are used so higher level tasks can be easily mapped into lower level operations by the system developer, and then automatically carried out as needed.

LEND's second mode of operation is through direct manipulation by users of the underlying graph. Thus, a linguist might want to find all words connected to any word in a sentence by up to three synonym associations. Such an

expert user of LEND must understand the syntax of our language for graph manipulation, given in Figure 21.4. For hypertext developers, the notation is relatively familiar, with nodes used to describe the various objects, but with arcs used instead of links. This pattern language supports a variety of retrieval schemes. Nodes can be found given their unique system assigned identifier, or sets of nodes satisfying a list of criteria can be returned. Thus, there are node conditional expressions, which can be used to specify more complex sub-graphs, as part of more complex pattern language expressions. A set theoretic description of the pattern language operators, based on the concept of unification, is given in Figure 21.5. A semantic description using many sorted algebras is shown in Figure 21.6. The operators in the language do not cover all possible graph access requirements. However, graphs make up a class of view objects (the natural, or base view).

From this class, the user is free to derive new classes of graph objects and to augment them with any further operations desired.

Our pattern language has been prototyped with a Prolog interpreter and has been tested with various types of data. We expect that our pattern language will be quite useful to support access to the graph layer, and that subsequent CODER development will benefit from this powerful formalism and tool, to carry out a wide variety of information management tasks.

```

Pattern      ::= PatternExp |
              compose(PatternExp, PatternExp) |
              or(PatternExp, PatternExp)

PatternExp   ::= Nodes |
              Paths

Paths        ::= Arcs |
              compose(Paths, Paths) |
              iteration(Nodes, Paths, Nodes, N, N)

Arcs         ::= ArcVar |
              newArcs(ArcID) |
              newArcs(Nodes, Labels, Nodes)

Labels       ::= LabelVar |
              LabelString

Nodes        ::= NodeVar |
              newNodes(NodeID)
              restrict(Nodes, NodeCondiExp)

```

Figure 21.4 Pattern Language Syntax in BNF.

```

retr(Node, UG) = {n | n ∈ VUG AND unify(Node, n)}
retr(Arc, UG) = {a | a ∈ EUG AND unify(Arc, a)}
retr(restrict(Nodes, CondiExp), UG) = {n | n ∈ Nodes AND CondiExp(n) evaluates to true in Class(n)}
retr(restrict(Labels, CondiExp), UG) = {[l] | l ∈ Labels AND CondiExp(l) evaluates to true in Class(l)}
retr(compose(Pattern1, Pattern2), UG) = {G ⊆ UG | (exists G1 ⊆ G)(G1 ∈ retr(Pattern1, UG))
AND (exists G2 ⊆ G)(G2 ∈ retr(Pattern2, UG))
AND (for all g ∈ G)(g ∈ G1 or g ∈ G2 )}
retr(or(Pattern1, Pattern2, UG) = retr(Pattern1, UG) UNION retr(Pattern2, UG))
retr(star(Pattern, Node1, Node2), UG) = retr(Pattern, UG) UNION {G ⊆ UG | (for some new variable X)
(exists G1 ⊆ G)(G1 ∈ retr(substitute(Pattern, X, Node2), UG))
AND (exists G2 ⊆ G)(G2 ∈ retr(star(substitute(Pattern, X, Node2)), UG))
AND (for all g ∈ G)(g ∈ G1 or g ∈ G2 )}
retr(iterate(Pattern, Node1, Node2, k), UG) =
IF k = 0 THEN {}
IF k > 1 THEN retr(Pattern, UG)
AND (exists G1 ⊆ G)(G1 ∈ retr(substitute(Pattern, X, Node2), UG))
AND (exists G2 ⊆ G)(G2 ∈ retr(iterate(substitute(Pattern, X, Node1, k-1), UG))
AND (for all g ∈ G)(g ∈ G1 or g ∈ G2 )}

```

NOTES:

UG is the "universal graph"; all nodes and arcs in a given network.

V_{UG} is the set of vertices (nodes) in UG; E_{UG} the set of edges (arcs) in UG.

unify(A, B) = A and B are the same except that either may have variables where the other has constant terms.

substitute(P, X, Y) = a pattern exactly like P except that X has been substituted everywhere for Y.

retr(P, UG) = set of graphs in the universe of graphs satisfying pattern P.

Figure 21.5 Pattern Language Operator Semantics.

Application of the Model in CODER

Most of the types shown in Figure 21.7 are of general utility. Some have been included, though, as examples of atomic and composite object types that have proven useful in CODER. Our particular type system is further detailed in Figure

Functionalities

Functionalities	Operators
1. LabelString	→ Labels
2. NodeID	→ Nodes
3. ArcID	→ Arcs
4. Nodes × Labels × Nodes	→ Arcs
5. Arcs	→ Paths
6. Paths × Paths	→ Paths
7. Nodes × Paths × Nodes × N × N	→ Paths
8. Nodes × NodeCondiExp	→ Nodes
9. Nodes	→ Pattern
10. Paths	→ Pattern
11. Patterns × Patterns	→ Pattern compose, or

Figure 21.6 Pattern Language Operators.

Layer	Class Name	Operations
all layers	general classes	create, destroy, compare, modify, access.
node layer	string text word number Boolean concept constant	concatenate, matchSubstring. vectorize, pattern match. morphological transform, stem. +, -, *, /. AND, OR, NOT. rename.
anchor layer	general anchor text anchor video anchor	anchorID → NodeID, anchorID → AnchorPlace. anchorID → text span. anchorID → video frame span.
link layer	link	LinkID → directionInfo. LinkID → (SourceAnchorID, DestinationAnchorID).
view layer	graph list set vector frame	traversePath, matchSubgraph head, tail, sublist, concatenate, elementInList. insert, membership, subset, union, intersect. addVect, subVect, scaleVect, cosine. setSlot, getSlot, matchFrame.

Figure 21.7 Typical Objects and Operations in Node, Anchor, Link and View layer.

21.7, where some of the methods for some of the objects are listed. Many of these have already been programmed and tested; work on others is proceeding.

More useful types for *information retrieval* (IR) objects will be added. For example, we will have graph objects, collection objects, etc. Since LEND is object-oriented, we have the freedom to expand the hierarchy and, at the same time, recognize desired performance for each class of objects. Currently, we are forming a SNePS network view from the underlying graph object class. Our purpose is to make LEND efficiently execute primitive operations required by SNePS's path-based inference on large externally stored data sets. We also plan to implement a view based on the F³L common representation language [Weaver, et al., 1989]. F³L's frames, facts, and functions have been widely used both in CODER and its knowledge bases. Once they have been transformed into a LEND view, all of CODER's existing knowledge bases can be loaded directly into LEND's integrated memory.

Another important application of the reference model is to conceptually and computationally model different IR systems in an efficient fashion using nodes and links. This will be beneficial to comparative research in IR techniques. It will also support research in IR from the AI perspective, both because it provides a way to merge classical document models with the semantic network or frame models mentioned above, and because data represented in the graph model can be divorced from its format and viewed reflexively. If this uniform representation of data can be supported with high access performance, then we will also realize some of the advantages of database technology (data sharing, simple management) with document and media information. It is these advantages that have made database management systems so popular in processing uniform data.

In Figure 21.8 we extend Figure 21.1 to show how the general model can apply to operations we are building into CODER. The first column gives the layer, the second lists parts of the CODER system that fall into that layer, and the third column describes in a more general situation what objects (types, operations) would occur at that level. Our reference model thus both expands upon earlier reference models, and relates to an actual system that has been under prototype development for several years. While further theoretical work and development is underway, CODER/LEND and our reference model should already provide some insight into future systems that integrate information retrieval and hypertext/hypermedia operations along with a variety of data, information, and knowledge.

Layer	CODER Modules	Typical Contents
Application	CODER/ALList, CODER/Cardiology, CODER/Navy.	hypertext, hypermedia, information retrieval, image management, authoring, training, tutoring, CAD/CAM, interactive digital video
Presentation	interface, dialogue.	devices: windows, pointing devices, speakers; media: text, audio, video; operations: generalization, editing, translation, access, browsing, selection, picking, querying, sequencing.
Session	discourse, blackboard, strategist, planner.	base selection, user identification, history management, versioning.
Base	search, lexicon browser.	base: knowledge, information, data; operations: search, inference.
View	list mgr, set mgr, F3L class mgrs.	graphs, lists, sets, vectors, relations, frames.
Link	LEND link mgr.	link ids, labels.
Anchor	LEND span mgr.	anchor ids, span descriptions.
Node	string mgr, text mgr., integer mgr.	textual strings, integers, object ids.
Communication	F3L communication & process mgr.	processes.
Physical machine	UNIX files, messages.	file systems, storage media, message support.
Transport	TCP/IP	ISO lower level equivalent
Network	TCP/IP	ISO lower level equivalent
DataLink	TCP/IP	ISO lower level equivalent
Physical	TCP/IP	ISO lower level equivalent

Figure 21.8 Use of Information Management System Reference Model in CODER.

About the Authors

Edward Fox

Edward A. Fox received his B.S. from MIT in 1972 in Electrical Engineering, and his MS and Ph.D. degrees from Cornell University. Dr. Fox is now an Associate Director for Research in the Computing Center, and an Associate Professor in the Department of Computer Science at Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106. He has served as principal investigator or co-principal investigator on more than twenty grants and contracts, sponsored by the National Science Foundation and other sponsors.

Dr. Fox has worked with the Association for Computing Machinery (ACM) in a variety of volunteer capacities. He is a member of the Publications Board, editor-in-chief of ACM Press Database and Electronic Products, vice chairman of the ACM Special Interest Group on Information Retrieval (SIGIR), and associate editor for "ACM Transactions on Information Systems." He has authored or co-authored several articles appearing in "Communications of the ACM" and was executive producer and script writer for "Interactive Digital Video," an hour-long documentary supplement to the July 1989 "Communications" special issue on Interactive Technologies, for which he served as guest editor.

Dr. Fox is editor and project manager for the Virginia Disc series of CD-ROMs, and is author of a variety of journal articles, book chapters, and other publications relating to information storage and retrieval. He serves on the editorial boards of "Information Processing and Management" and "The CD-ROM Professional," on the research advisory council for Chemical Abstracts Service, and has been on a variety of program committees and panels.

Qi Fan Chen

Qi Fan Chen is a Ph.D. candidate at Virginia Polytechnic Institute and State University in the Department of Computer Science. His dissertation work on "An Object-Oriented Network Database System for Information Retrieval Applications" deals with development of LEND and its integration with CODER, as discussed in this chapter. He has co-authored several articles relating to the CODER system and the construction of minimal perfect hash functions to support efficient access to hypermedia and other networked information.

Robert K. France

Robert K. France is a research scientist at Virginia Polytechnic Institute and State University in the Computer Science Department. His MS thesis spelled out the design of CODER, and he has thus been involved in the development of CODER.

since its inception. Mr. France has co-authored a variety of articles, dealing with the architecture, development, knowledge representation methods, and other aspects of this innovative system.

References

- Artandi, S. (1971). Document Retrieval and the Concept of Sets. *Journal of the American Society for Information Science*, July-August, 289-290.
- Bates, M. J. (1981). *Search Techniques*. Annual Review of Information Science and Technology, Martha E. Williams (Ed.), ASIS/Elsevier Science Publishers B. V., Amsterdam, Vol. 16, 139-170.
- Belkin, N. J., Seeger, T., and Wersig, G. (1983). Distributed Expert Problem Treatment as a Model for Information System Analysis and Design. *Journal of Information Science* 5:153-167.
- Belkin, N. J., Borgman, C., Brooks, H., Bylander, T., Croft, W., Daniels, P., Deerwester, S., Fox, E., Ingwersen, P., Rada, R., Sparck Jones, K., Thompson, R., and Walker, D. (1987). *Distributed Expert-Based Information Systems: An Interdisciplinary Approach*. IP&M, 23(5):395-409.
- Belkin, N. J., and Croft, W. B. (1987). Retrieval Techniques. *Annual Review of Information Science and Technology*, Martha E. Williams (Ed.), ASIS / Elsevier Science Publishers B. V., Amsterdam, Vol. 22, 109-145.
- Bookstein, A. (1985). Probability and Fuzzy-Set Applications to Information Retrieval. *Annual Review of Information Science and Technology*, Martha E. Williams (Ed.), American Society for Information Science / Elsevier Science Publishers B. V., Amsterdam, Vol. 20, 117-152.
- Bush, V. (1945). "As We May Think." *Atlantic Monthly*, 176, July, 101-108.
- Chen, Q. F. (1989). *Proposed Specification for an Associative Network Database*. Draft report, Computer Science Department, Virginia Polytechnic Institute and State University, Blacksburg, VA.
- Chen, Q. F. (1990). *An Object-Oriented Network Database System for Information Retrieval Applications*. Dissertation Proposal, Computer Science Department, Virginia Polytechnic Institute and State University.
- Cohen, P. R., and Kjeldsen, R. (1989). "Information Retrieval by Constrained Spreading Activation in Semantic Networks." *Information Processing and Management*, Vol. 23, No. 4:255-268.
- Comer, D. (1979). The Ubiquitous B-tree. *ACM Computing Surveys* 11(2):121-137, June.
- Consens, M., and Mendelzon, A. (1989). "Expressing Structural Hypertext Queries in GraphLog." *Hypertext '89 Proceedings*, Nov. 5-8, 1989, Pittsburgh, Special Issue—ACM SIGCHI Bulletin, 269-292.
- Croft, W. B., and Turtle, H. (1989). "A Retrieval Model Incorporating Hypertext Links." *Hypertext '89 Proceedings*, Nov. 5-8, 1989, Pittsburgh, Special Issue—ACM SIGCHI Bulletin, 213-24.
- Deogun, J. S., and Raghavan, V. J. (1988). "Integration of Information Retrieval and Database Management Systems." *Information Processing and Management*, Vol. 24, No. 3, 303-313.
- DeRose, S. (1989). "Expanding the Notion of Links." *Hypertext '89 Proceedings*, Nov. 5-8, 1989, Pittsburgh, Special Issue—ACM SIGCHI Bulletin, 249-258.
- Devlin, Joseph (1991). "Standards for Hypertext." *Hypertext / Hypermedia Handbook*, Emily Berk and Joseph Devlin (Eds.), McGraw-Hill, New York, NY.
- Embod, R. J., and Du, H. C. (1988). "Dynamic Hashing Schemes." *ACM Computing Surveys* 20:85-113.
- Engelbart, D. (1962). *Augmenting Human Intellect: A Conceptual Framework*. SRI Technical Report AFOSR-3223, Contract AF 49(638)-1024, October.
- Findler, Nicholas (1979). *Associative Networks: Representation and Use of Knowledge by Computers*. Academic Press, Orlando, FL.
- Fox, E. (1981). "Implementing SMART for Minicomputers Via Relational Processing with Abstract Data Types." *Joint Proc. of SIGSMALL Symp. on Small Systems and SIGMOD Workshop on Small Data Base Systems*, Oct. 1981, Orlando, FL, ACM SIGSMALL Newsletter, 7(2):119-129.
- Fox, E. (1983). *Characterization of Two New Experimental Collections in Computer and Information Science Containing Textual and Bibliographic Concepts*. TR 83-561, Cornell Univ. Dept. of Computer Science, Sept., Ithaca, NY.
- Fox, E. (1986). "Information Retrieval: Research Into New Capabilities." Steve Lambert and Suzanne Ropiquet, editors, *CD-ROM: The New Papyrus*, Microsoft Press, Inc. 143-174.
- Fox, E., Wohlwend, R., Sheldon, P., Chen, Q. F., and France, R. (1986). *Building the CODER Lexicon: The Collins English Dictionary and Its Adverb Definitions*, TR-86-23, October, Virginia Polytechnic Institute and State University Computer Science Dept., Blacksburg, VA.
- Fox, E., and France, R. (1987). "Architecture of an Expert System for Composite Document Analysis, Representation and Retrieval." *International Journal of Approximate Reasoning*, 1(2):151-175.
- Fox, E., and Chen, Q. F. (1987). "Text Analysis in the CODER System." *Proceedings Fourth Annual USC Computer Science Symposium: Language and Data in Systems*, April 8, Columbia, SC, 7-14.
- Fox, E. (1987). "Development of the CODER System: A Testbed for Artificial Intelligence Methods in Information Retrieval." *IP&M*, 23(4):341-366.
- Fox, E. (1987). "Testing the Applicability of Intelligent Methods for Information Retrieval." *Information Services and Use*, 7:119-138.
- Fox, E., Nutter, J., Ahlswede, T., Evens, M., and Markowitz, J. (1988). "Building a Large Thesaurus for Information Retrieval." *Proceedings Second Conference on Applied Natural Language Processing*, Feb. 9-12, Austin, Texas, 101-108.
- Fox, E., Nunn, G., and Lee, W. (1988). "Coefficients for Combining Concept Classes in a Collection." *Proc. 11th Int'l Conf. on Research and Development in Information Retrieval*, June 13-15, Grenoble, France, 291-307.
- Fox, E. (1988). *ACM Press Database and Electronic Products—New Services for the Information Age*. Commun. of the ACM, 31(8):948-951.
- Fox, E. (1988). Optical Disks and CD-ROM: Publishing and Access. *Annual Review of Information Science and Technology*, Martha E. Williams (Ed.), ASIS / Elsevier Science Publishers B. V., Amsterdam, Vol. 23, 85-124.
- Fox, E. (1988). "Improved Retrieval Using a Relational Thesaurus for Automatic Expansion of Boolean Logic Queries." *Relational Models of the Lexicon: Representing Knowledge in Semantic Networks*, Martha Walton Evens (Ed.), Cambridge Univ. Press, Cambridge, UK, 199-210.
- Fox, E. (1989). "Research and Development of Information Retrieval Models and their Application." *Information Processing and Management* (IP&M), 25(1):1-5.
- Fox, E., Chen, Q. F., Heath, L., and Datta, S. (1989). "A More Cost Effective Algorithm for Finding Perfect Hash Functions." *Proc. ACM 1989 Computer Science Conference*, February 21-23, Louisville, KY, 114-122.
- Fox, E., Heath, L., and Chen, Q. F. (1989). *An O(n log n) Algorithm for Finding Minimal Perfect Hash Functions*, TR-89-10, VPI&SU Computer Science Dept., April, Blacksburg, VA. Submitted for publication.
- Fox, E. (1989). "The Coming Revolution in Interactive Digital Multimedia Systems." *Commun. of the ACM*, 32(7):794-801.
- Fox, E. executive producer, script writer. *Interactive Digital Video*. A videotape supplement to July 1989 Communications of the ACM, produced for ACM Press Database and Electronic Products with assistance from IBM Thomas J. Watson Research Center (with Guillermo Pulido as producer/director).

- Fox, E., and Chen, Q. F. (1990). *ACM's Need and the CODER/LEND Approach*. Paper HT-5 in Hypertext Standardization Workshop, Jan. 16-18, National Inst. of Standards and Tech., Gaithersburg, MD.
- Fox, E., editor and project manager. *Virginia Disc One*. Produced by Nimbus Records, 1990. Blacksburg VA: Virginia Polytechnic Institute and State University Press.
- Fox, E., Heath, L., Chen, Q. F., and Daoud, A. (1990). "Practical Minimal Perfect Hash Functions for Large Databases." TR 90-41, Virginia Polytechnic Institute and State University, Department of Computer Science, Blacksburg, VA, August.
- Fox, E., Rous, B., and Marchionini, G. (1991). "ACM's Hypertext and Hypermedia Publishing Projects." *Hypertext / Hypermedia Handbook*, Emily Berk and Joseph Devlin (Eds.), McGraw-Hill, New York, NY.
- France, R., Fox, E., Nutter, J. T., Chen, Q. F. (1989). "Building A Relational Lexicon for Text Understanding and Retrieval." *Proc. First Int'l Language Acquisition Workshop*, Aug. 21, Detroit, MI.
- France, R. (1989). *Proposed Specification for an Associative Network Database*. Draft report, Computer Science Department, Virginia Polytechnic Institute and State University, Blacksburg, VA.
- Furuta, R., and Stotts, P. D. (1989). "Programmable Browsing Semantics in Trellis." *Hypertext '89 Proceedings*, Nov. 5-8, Pittsburgh, Special Issue—ACM SIGCHI Bulletin, 27-42.
- Furuta, R., and Stotts, P. D. (1990). "The Trellis Hypertext Reference Model." Paper HT-6 in Hypertext Standardization Workshop, Jan. 16-18, National Inst. of Standards and Tech., Gaithersburg, MD.
- Guling, R. H., Zicari, R., and Choy, D. (1989). "An Algebra for Structured Office Documents." ACM Transactions on Office Information Systems, Vol. 7, No. 4, 123-157 April.
- Halasz, F., and Schwartz, M. (1990). "The Dexter Hypertext Reference Model." Paper HT-9 in Hypertext Standardization Workshop, Jan. 16-18, National Inst. of Standards and Tech., Gaithersburg, MD.
- Kim, W., and Lochovsky, F. H., editors (1989). *Object-Oriented Concepts, Databases and Applications*. ACM Press, New York, NY.
- Lange, D. (1990). "A Formal Model of Hypertext." Paper HT-13 in Hypertext Standardization Workshop, Jan. 16-18, National Inst. of Standards and Technology, Gaithersburg, MD.
- Lindeman, M. J., Crabb, J. C., Borneau, J. R., and Wehrli, V. F. (1990). "Designing a Scholars Electronic Library: The Interaction of Human Factors and Computer Science Tasks." *Advances in Human-Computer Interaction*, Vol. III.
- Littleford, Alan (1991). "Artificial Intelligence and Hypermedia." *Hypertext / Hypermedia handbook*, Emily Berk and Joseph Devlin (Eds.), McGraw-Hill, New York, NY.
- Lynch, C. A. (1987). "Extending Relational Database Management Systems for Information Retrieval Applications." Dissertation. University of California, Berkeley.
- Marshall, C., and Irish, P. (1989). "Guided Tours and On-Line Presentations: How Authors Make Existing Hypertext Intelligible for Readers." Hypertext '89 Proceedings, Nov. 5-8, 1989, Pittsburgh, Special Issue—ACM SIGCHI Bulletin, 15-26.
- Morgado, Emesio J. M. (1985). "Semantic Networks as Abstract Data Types." Dissertation. Department of Computer Science, University of New York at Buffalo.
- Parunak, H. Van Dyke (1989). "Hypermedia Topologies and User Navigation." Hypertext '89 Proceedings, Nov. 5-8, 1989, Pittsburgh, Special Issue—ACM SIGCHI Bulletin, 43-50.
- Pearl, A. (1989). "Sun's Link Service: A protocol for Open Linking." Hypertext '89 Proceedings, Nov. 5-8, 1989, Pittsburgh, Special Issue—ACM SIGCHI Bulletin, 137-146.
- Robertson, S. E., and Jones, K. S. (1976). "Relevance Weighting of Search Terms." *Journal of Documentation*, May-June, 129-146.
- Salton, G., Fox, E., and Wu, H. (1983). "Extended Boolean Information Retrieval." *Communications of the ACM*, 26(11):1022-1036.
- Salton, G., Fox, E., and Voorhees, E. (1985). "Advanced Feedback Methods in Information Retrieval." *Journal of the American Society for Information Science*, 36(3):200-210.
- Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley, Reading MA.
- Shapiro, C. (1979). "The SNePS Semantic Network Processing System." Nicholas Findler. *Associative Networks: Representation and Use of Knowledge by Computers*. Academic Press, Orlando, FL, 179-203.
- Shapiro, C., et al. (1989). *SNePS User's Manual*. Department of Computer Science, State University of New York at Buffalo.
- Sneiderman, B., (Ed.). (1988). *Hypertext on Hypertext, HyperTies edition*. ACM Press Database and Electronic Products, New York, NY.
- Stotts, P. David, and Furuta, Richard (1989). "Petri-Net-Based Hypertext: Document Structure with Browsing Semantics." ACM Transactions on Information Systems, Vol. 7, No. 1, January, 3-29.
- Tanenbaum, Andrew S. (1981). *Computer Networks*. Prentice Hall, Englewood Cliffs, NJ.
- Teskey, F. N. (1989). "User Models and World models for Data, Information, and Knowledge." *Information Processing and Management*. Vol. 25, No. 1.
- Tompa, F. W. (1989). "A Data Model for Flexible Hypertext Database Systems." ACM Transactions on Information Systems, Vol. 7, No. 1, January, 85-100.
- Trigg, R., and Weiser, M. (1986). "TEXTNET: a network-based approach to text handling." ACM Trans. on Office Information Systems 4(1):1-23, Jan.
- Weaver, M., France, R., Chen, Q. F., and Fox, E. (1989). "Using a Frame-Based Language for Information Retrieval." *Int. Journal of Intelligent Systems*, 4(3):223-257.
- Zellweger, P. T. (1989). "Scripted Documents: A Hypermedia Path Mechanism." Hypertext '89 Proceedings, Nov. 5-8, 1989, Pittsburgh, Special Issue—ACM SIGCHI Bulletin, 1-14.
- Zimmerman, H. (1980). "OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection." *IEEE Trans. Commun.*, Vol. COM-28, 425-432, April.