Hyperstructure-Based Search Methods for the World Wide Web

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)

von

Zhanzi Qiu
(Master of Science)

geboren in Fujian, China

Referent: Prof. Dr. Erich J. Neuhold
Koreferent: Prof. Dr. James Geller
Tag der Einreichung: 19.01.2004
Tag der mündlichen Prüfung: 22.03.2004

Darmstadt 2004
D17
Darmstädter Dissertation

Abstract

Keywords: hyperstructure, search methods, World Wide Web, XML/RDF

This thesis presents several hyperstructure-based Web search methods and a prototype system that is designed to implement the methods. Given the context of hyperlink structural and semantic information that is representable with new Web standards, this thesis is an effort to answer the open question of how to efficiently make use of such information for searching the Web and filtering and retrieving relevant information.

The hyperstructure-based approach taken in this thesis is an extension to the traditional structure-based search method, which mainly handles hierarchical structures (composed by non-linking mechanisms) in structured documents (e.g., XML). In addition to such hierarchical structures, this approach can also handle both hierarchical and non-hierarchical structures composed by linking mechanisms. Compared to other link-based approaches that largely take into account the quantity of links in their search methods, this approach also makes use of the semantic information in links and link-based structures. It is in line with the trend of Web development with regard to capturing rich structural and semantic information and thereby capitalizing on the potential of new search methods.

The hyperstructure-based search methods presented in this thesis can be applied to improve the search quality on the Web as the Web evolves from a poorly structured to a more structured, semantic-rich network. More concretely, by making use of hypertext composites and contexts, the search results can be more specific with respect to users' information needs, and additionally, the users' efforts to interpret the search results can be reduced. Presenting structured search results based on hypertext composites as inter-linked nodes/pages rather than separate nodes/pages helps users

understand the retrieved information better. By making use of semantic information in hyperstructures (e.g., types of links and nodes), better filters can be developed for selecting and ranking the Web pages retrieved by search systems. These pages can be either intermediate information for further processing or final search results presented to users. By making use of domain models, domain-specific structure-based search methods can be developed, which may generate better results than general search methods that do not understand the domain-specific information.

Kurzfassung

Schlüsselwörter: Hyperstruktur, Suchmethoden, World Wide Web, XML/RDF

Verschiedene neue Internet-Standards, vor allem XML und RDF, versprechen zwar eine Verbesserung im Zugang zu den Informationen im Internet. Bisher ist es jedoch unklar, wie die neuen Strukturen und semantischen Informationen, die durch diese Standards ausgedruckt werden können, für Informationssuche am besten eingesetzt werden können. Diese Arbeit hat hierauf eine Antwort gegeben. Sie präsentiert vier verschiedene Hyperstruktur-basierte Suchmethoden und ein prototypisches Suchsystem. Einige Experimente wurden auch durchgefuehrt. Die Ergebnisse zeigen, daß mit den neuen Suchmethoden folgendes erreicht werden kann:

- Neuartige formular-basierte Queries können gestellt werden.
- Suchergebnisse können in ihrem ursprünglichen Kontext gesichtet werden (d.h. innerhalb eines Dokuments oder einer Gruppe von Dokumenten). Dadurch können Benutzer die Relevanz besser beurteilen.
- Bessere Filter für die Auswahl und Sortierung nach Relevanz können entwickelt werden, bevor die gefundenen Informationen bearbeitet und dem Benutzer präsentiert werden.
- Domänenspezifische Suchmethoden können entwickelt werden, die bessere Ergebnisse als allgemeine Suchmethoden liefern, da letztere domänenspezifische Information nicht "verstehen".

Acknowledgements

I would like to express my deep gratitude to my supervisor, Prof. Dr. Erich J. Neuhold, for his advice, support, and encouragement throughout my doctoral programme. I would like to thank my second advisor, Prof. Dr. James Geller, for his comments and suggestions on the thesis.

A number of other people deserve special acknowledgements. I thank Dr. Matthias Hemmje for his advice and guidance all through the way in completing this thesis. I thank Dr. Weigang Wang, Dr. Ulrich Thiel and Dr. Reginald Ferber for their valuable advice that helped me make my thesis topic clear. Also, I thank Prof. Dr. Klaus Mätzel for his advice and support during the period I worked in his division.

This thesis was done with the financial and technical support of GMD-IPSI, now FhG-IPSI. All the kind help of the colleagues in the institute, especially in the divisions DELITE, TOPAS and the institute office, are appreciated. The support by the Computer Science Department, Darmstadt University of Technology deserves special thanks.

Finally, I would like to express my sincere appreciation to my family for their support and understanding in helping me to complete this thesis. This thesis is especially dedicated to my dad, Ruizheng Qiu, who is always with me in my heart.

Table of Contents

1 Introduction

The widespread use of the Web makes hypertext a common concept for ordinary users. The ability to browse is generally regarded as one of the most important reasons for using hypertext, however, searching facilities should also be supported in modern hypertext environments [Halasz 1988]. The Web, because of its huge scale and arbitrary structure, creates many challenges for the development of its searching capabilities.

1.1 The Problem

In the traditional Web, links are not typed, and there is no link-based composition mechanism. Thus the Web lacks structure [Trigg 1996]. When a document is split into multiple Web pages (files), it is hard for users and computer systems to tell whether the target of a link is a logical part of the current document or referential material related to the document. Often users want to get a copy of an entire document they find on the Web, but unless the document is contained in a single Web page, they have to save it manually, page by page. This is not convenient. Bill Gates in his Comdex'96 keynote lecture said the following about the WWW:

We also need to take links and give them types, so that if I go to a site and say I want to use this site offline, it can understand based on the link types what pages to bring down. And so we need new standards in that area. The standards committees are working very well here - W3C, IETF - and so I think this richer structure will come quite rapidly [Gates 1996].

To improve the ability of expressing structures and semantics on it, several new Web standards, e.g., XML [XML], XLink [XLink], XPointer [XPointer], RDF [RDF], have been developed or are under development. They open new opportunities to improve the information access on the Web by taking advantage of the structural and semantic information in hyperstructures. However, it is still an open question how to efficiently make use of this kind of information for searching

the Web and filtering and retrieving relevant information. This thesis aims to answer this question. The focus of the thesis is to develop search methods that fully exploit the ever-enriched structural and semantic information in the hyperspace of the World Wide Web.

1.2 Existing Search Methods

Corresponding to the evolution of the Web from a poorly structured to a more structured, semantic-rich network, search methods that apply to it also evolve from using little structural and semantic information to using more such information that is available.

The first generation search engines on the Web are typically **keyword-based**. This kind of search may retrieve irrelevant information as a word may have different meanings, but may also miss some information that should be retrieved when different words of the same meaning are used. Furthermore, keyword-based search fails to integrate information spread over different sources.

As a solution to the problems, **metadata-based search** methods are used in many large search engines. These engines, such as AltaVista [AltaVista] and InfoSeek [InfoSeek], use metadata about Web pages to develop the meaning or ranking of the pages or provide searches in metadata fields individually or with Boolean combinations. Currently, the metadata used is normally stored in the head of HTML pages in the form of attribute-value pairs.

Structure-based search is one of the **metadata-based search** techniques. It uses structural information embedded in documents to retrieve parts of the documents and/or enhance the **keyword-based search** results. Traditionally this approach focuses on handling hierarchical structures in structured documents (e.g. XML). As for a hyperlinked space like the Web, it cannot meet our demand of improving Web search by making use of the link-based structural information.

1.3 My Approach

This thesis takes a hyperstructure-based approach. It focuses on making use of the rich structural and semantic information that is made available by the newly developed hypermedia related Web standards (e.g., XML [XML], XLink [XLink], and XPointer [XPointer]).

To make this approach applicable, three basic questions have to be answered:

- What structural and semantic information can be found in hyperstructures?
- How to represent this information in Web standards?
- How to efficiently apply this information for searching the Web and filtering and retrieving relevant information?

It is clear that the third is the central research question in this thesis. The answers to the first two are the preliminary steps to address the third.

1.4 Research Methods

To answer the research questions, three research methods have been used in close interaction:

- Theoretical exploitation
  - to identify structural and semantic information in hyperstructures, and,
  - to explore search methods that make use of the information,
- System construction
  - to develop a prototype system, i.e., a Web search engine, to apply the methods explored, and
- Empirical studies
  - to perform some evaluation studies to test the methods and tools.

1.5 Innovations and Limitations

The hyperstructure-based approach is an extension to the traditional structure-based search method, which mainly handles hierarchical structures (composed by non-linking mechanisms) in structured documents (e.g., XML). In addition to such hierarchical structures, the hyperstructure-based approach can also handle both hierarchical and non-hierarchical structures composed by linking mechanisms.

Compared to other link-based approaches that largely take into account the quantity of links in their search methods, this approach also makes use of the semantic information in links and link-based structures. It fits the trend of Web development with regard to capturing rich structural and semantic information and thereby capitalizing on the potential of new search methods.

The hyperstructure-based approach has a number of innovative aspects. By making use of structural and semantic information, new kinds of form-based queries can be formulated. By making use of links, anchors, and the structures of documents, the search results can be presented in their original contexts (within a document or a group of related documents). This can help users understand the retrieved information better. By making use of semantic information in hyperstructure (e.g., types of links and nodes), better filters can be developed for information selection and ranking. By making use of domain models, domain-specific search methods can be developed, which can generate better results than general search methods that do not understand the domain-specific information.

However, this approach asks document providers to adopt the new Web standards and to include additional information in their documents. The search methods based on this approach may not be applicable to many existing data on the traditional Web.

1.6 Definitions of Terms

Hypertext refers to the kind of text in which data is stored in *nodes* interconnected by *links*. *Nodes* are containers of *information chunks*, which form the basic content

of hypertext; *links* connect nodes in a nonlinear organization - network. *Hypermedia* is multimedia hypertext whose nodes may contain *information chunks* in multiple media types, such as plain text, graphics, video, and sound. In the context of this thesis, the term *hypertext* is used generally to refer to both text-only and multimedia hypertext. A *hypertext system* is a tool that can be used to manage the nodes and links in a collection of hypertext.

Hyperstructure refers to an information structure that is not necessarily linear. *Structural and Semantic information in hyperstructures* refers to machine understandable information about hypertext components and the overall structural characteristics of hypertext. In the context of this thesis, it includes basic hypertext components and their types, high-level hypertext structures formed by basic hypertext components, document models and domain models.

A *document* in this thesis refers to an information product that focuses on a certain topic and often has at least one overall logical structure. The logical structure defines the boundary of the document. A *document component* is a logical part of a document.

When the information in a document is organized in a network form, the document is called *hypertext document*, or simply *hyperdocument*. In other words, a *hypertext document or hyperdocument* is the combined collection of links and nodes that makes up a hypertext network. A *hyperspace* refers to the information space that contains many hyperdocuments.

1.7 Structure of the Document

The rest of this document is organized as follows. Chapter 2 presents the background and related work of the thesis. Chapter 3 describes structural and semantic information in hyperstructures and their standard representation. Chapter 4 explores search methods that make use of the above-identified information. Chapter 5 presents a prototype system named HyperSM, which implements the search methods explored in the thesis. Chapter 6 describes the evaluation studies to test the

search methods and tools explored. Finally, Chapter 7 concludes the work in the thesis.

2 Background and Related Work

This thesis relates to many basic research fields, among which the World Wide Web, hypertext and information retrieval are the main ones. The methods explored in this thesis can be applied in the areas of digital libraries and knowledge management. Some publications in these research fields and application areas have proposed approaches taking advantage of link structures, which are the main object studied in this thesis.

2.1 Application Areas

2.1.1 Digital Libraries

Throughout recorded history, libraries have played a significant societal role in the preservation and dissemination of human knowledge. Nowadays, people anywhere anytime can use any Internet-connected digital device to search a large percentage of human knowledge. Via the Internet, they can access knowledge in digital collections created by traditional libraries, museums, archives, universities, government agencies, specialized organizations, and even individuals around the world. In a broad sense, any such a digital collection is a digital library. It can provide documents that are of many media types including text, video, sound or image. When a computer user browses the Web and stops at a specific site, he/she is visiting a digital library. *(View from "Digital Libraries: Universal Access to Human Knowledge", President's Information Technology Advisory Committee (PITAC), Panel on Digital Libraries, U.S.A., February 2001)* [PITAC 2001].

The more specific, professional meaning of the term digital library is a digital collection that is assumed to be of continuing interest for human users. Such collections mainly come from libraries, museums or archives. In this way, we distinguish digital libraries from other kinds of information collections that are of current interest but are of little value for long-term availability and preservation.

(View From ECDL 2001 panel: Digital Library Programs: Current Status and Future Plans) [ECDL 2001 Panel].

Digital libraries promise new societal benefits and will have many features not possible in traditional libraries. They also challenge us with respect to technical, operational, and legal issues [PITAC 2001]. Related to this thesis, with the entensive use of hypertext links to interconnect information, digital libraries enable users to find related digital material on a particular topic. On the traditional Web, this is supported mainly by browsing. In order to improve the search and retrieval of information and knowledge for stored digital content, new technical capabilities are needed. For instance, metadata standarization efforts (e.g. Dublin Core [Dublin Core]) for digital libraries must be advanced and new approaches and methods that apply the metadata for search purposes must be explored. The work presented in this thesis takes a hyperstructure-based approach. The search methods proposed in the thesis are a special kind of metadata-based methods and can be applied to various digital libraries available on the Internet.

2.1.2 Knowledge Management

Knowledge management (KM) is a major issue for human resource management, enterprise organization, and enterprise culture [Staab et al. 2001]. The motivation underlying traditional KM is to store information from the past so that lessons will not be forgotten. The goal of modern KM is to enable innovative practices at an organizational (community) level by supporting collaboration and communication among knowledge workers in the same domain and across domains [Fischer&Ostwald 2001]. From the reuse point of view, KM facilitates the acquisition, capturing, deployment, access, and reuse of information and knowledge – typically using contemporary technology [Leary 2001]. The KM system is now the most important system to achieve competitive advantages in a modern corporation.

Throughout the whole cyclic process involving three related activities - creation, integration and dissemination – of KM [Fischer&Ostwald 2001], being able to access stored information in existing knowledge repositories is mostly required by knowledge workers. That is, search tools are a must in a KM system. For example, in

a conventional, closed KM system, users usually search for stored information using database queries.

Hypermedia is one of the major techniques applied in KM. Hypertext links are popularly used for connecting knowledge documents as well as other information items in a knowledge management system. This is to say that a knowledge repository or library is often a collection of digitized knowledge documents that are organized in a hyperstructure and can thus be seen as a special kind of digital libraries. Any hyperstructure-based query and search method can be applied to access such collections. If a knowledge repository is accessed through the Web and externalized knowledge documents stored are represented with the new Web standards, the hyperstructure-based search methods proposed in this thesis can be applied to improve knowledge search results in this global information space.

In addition, as most knowledge in the world is available in the form of national language documents and more and more such documents are now digitized and distributed in the Internet, the Web is often viewed as a global knowledge repository or a global digital library (but not a well organized one). Searching and filtering the knowledge contained in this repository/library becomes crucial but time-consuming, and is not efficient for knowledge workers in constructing a domain-specific knowledge repository. At this point, it is expected that hyperstructure-based search methods will help a lot.

2.2 Related Basic Research Fields

2.2.1 Information Retrieval Issues

Information Retrieval (IR) is an established technology that has been delivering solutions to users for more than five decades, and yet it is still an active research area [Smeaton 1996]. The problem of information storage and retrieval is simply stated: we have vast amounts of information to which accurate and speedy access is becoming ever more difficult. One effect of this is that relevant information gets

ignored since it is never uncovered, which in turn leads to much duplication of work and effort [Rijsbergen 1979].

2.2.1.1 An Information Retrieval System

An *information retrieval system (IRS)* deals with the representation, storage and access to documents or representatives of documents (document surrogates) [Minker 1977]. The purpose of such a system is to satisfy a variety of information needs [Marchionini 1995] of its users. An *information need* is formulated by a query or request and the IRS will answer with a list (maybe empty) of information items in a database. The information items are mostly in the form of pure text (called documents), though the importance of other kinds of information items (multimedia) is increasing.

The key functions performed by an information retrieval system are (as shown in Figure 2.1):

- Indexing
- Query formulation
- Matching and similarity analysis
- Evaluation of retrieved items and feedback for query refinement.

The process of *indexing* is to convert the information items to a special form (internal representation) using an indexing language, which is either pre-specified (controlled) or taken freely from the text of the information items and information requests (uncontrolled). It can be performed either manually (labor intensive and thus very expensive) or automatically by extracting index terms from the text of the information item using a statistical or linguistic procedure. In some systems, each index term may be assigned a weight to reflect its relative importance on the subject matter of the information item.

The *query formulation* represents the query negotiation process. It is to convert the users' requests to a representation consisting of elements from the indexing language, through a similar procedure to the one applied in the indexing.

The *matching* process determines which information items should be retrieved in response to a query based on the internal representations of the requests and information items consisting of elements from the indexing language. Most IRSs also perform a *similarity analysis* of the matching items. For each such item a similarity value is computed to estimate how well the item satisfies the information need. The similarity values are often used to rank the items.

The *evaluation* process is to judge how well a retrieved information item meets the users' information need, i.e., the relevance of the item to the query. Two kinds of relevance assessments can be made. One is by the system, i.e., following the retrieval process the system determines the similarity value of matching items. The other is by the users, i.e., the users evaluate which items are *relevant* or *irrelevant* to their information needs.

R*elevance feedback* [Robertson&Jones 1976] is an automatic query refinement based on users' assessments regarding the relevance of individual retrieved items. For instance, terms present in previously retrieved items that have been identified as

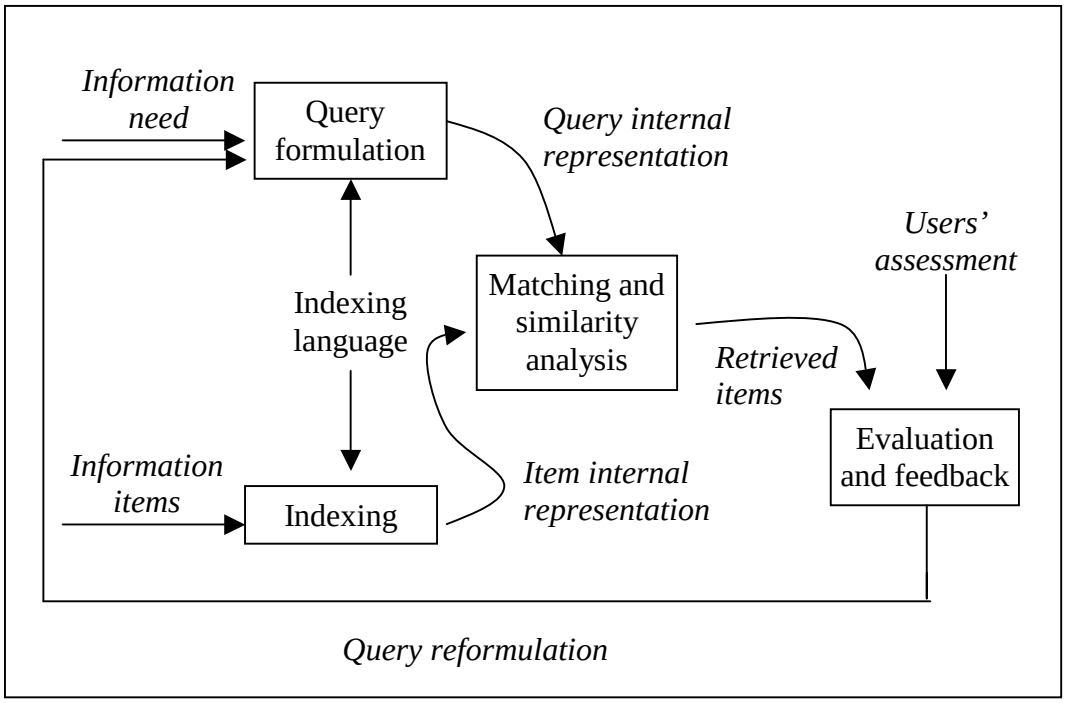


Figure 2.1 A typical information retrieval system

relevant to the user's query are added to the original query. This is useful for improving retrieval effectiveness.

2.2.1.2 Measures of Retrieval Effectiveness

The effectiveness of an IR system expresses how well the retrieved result satisfies an information need. The usual measures of retrieval effectiveness of IR systems are *recall* and *precision* [Salton&Mill 1983; Salton 1989]. Both measures are based on the users' assessments following the retrieval process. The *recall* is the proportion of relevant material retrieved. The *precision* is the proportion of retrieved material that is relevant:

$$\text{Recall} = \frac{\text{Number of relevant information items retrieved}}{\text{Total number of relevant items in the collection}}$$

$$\text{Precision} = \frac{\text{Number of relevant information items retrieved}}{\text{Total number of items retrieved}}$$

One common way to express retrieval effectiveness is the recall-precision graph [Salton 1971] (as shown in Figure 2.2), where the precision, $\Pi(\rho)$, is given for different values of recall, ρ.

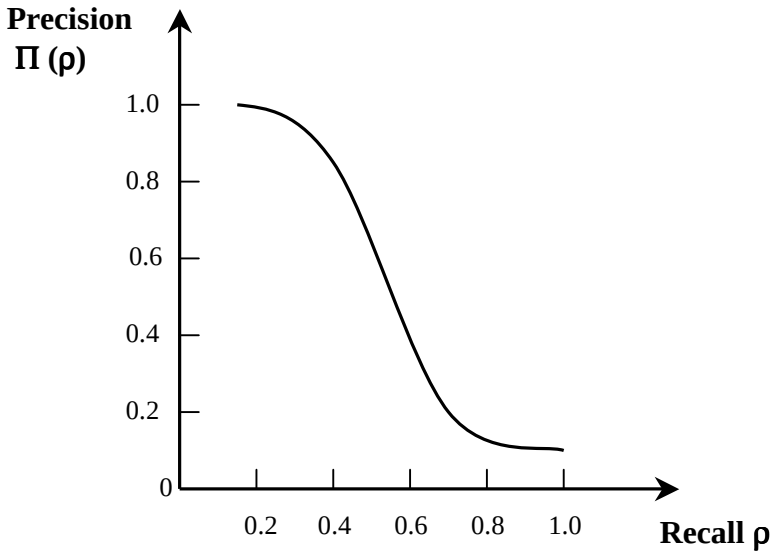


Figure 2.2 Typical average recall-precision graph

In typical keyword-based retrieval systems, *recall* can be increased as the number of retrieved items increases, while the *precision* is likely to decrease at the same time. For this reason, mechanisms improving recall affect precision and vice versa. A trade-off is often necessary between those mechanisms to control retrieval effectiveness.

2.2.2 Hypertext Issues

The concept of hypertext has been around for a long time. This section reviews the general features of hypertext and introduces the best-known hypertext model - the so-called Dexter Hypertext Reference Model [Halasz&Schwartz 1990; 1994]**.** The information retrieval issue in hypertext will be discussed in Sections 2.3 and 2.4.

2.2.2.1 Hypertext

The original idea of hypertext was first brought forward by Vannevar Bush in 1945 [Bush 1945]. He described a device called “memex”, in which an “individual stores his books, records and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.” He described the essential feature of memex as its ability to tie two items together.

In 1965, Ted Nelson coined the word “hypertext” (non-liner text) and defined it as “a body of written or pictorial material interconnected in a complex way that it could not be conveniently represented on paper, i.e., in a linear form. It may contain summaries or maps of its contents and their interrelations; it may contain annotations, additions and footnotes from scholars who have examined it.” [Nelson 1965].

Hypertext has been defined as “an approach to information management in which data is stored in a network of nodes connected by links. Nodes can contain text, graphics, audio, video as well as source code or other forms of data.” [Smith&Weiss 1988]. *Hypertext* with multimedia is called “*hypermedia*”. However, the term *hypertext* is often used indifferently for both *hypertext* and *hypermedia*, as most of the present hypertext systems are able to manage different kinds (medias) of digitized documents.

Essentially, *hypertext* is richly linked, document-like information. It provides a novel way of directly accessing and managing data. Readers of a hypertext document can access the information stored in it from many perceived points of view, in any order or through different paths.

2.2.2.2 Hypertext Challenges

The promise of hypertext lies in its ability to produce complex, richly connected and cross-referenced chunks of information. However, the essential nature of hypertext also carries many challenging problems, among which the two most well known are [Conklin 1987]:

- disorientation
- cognitive overhead

The problem of disorientation or “getting lost in space” arises from the need to know where users are located in the network while reading, where they came from, and how they can get to another place in the network. Cognitive overhead is the additional overhead on authors to create, name, and keep track of nodes and links. For readers, it is the overhead due to making decisions as to which links to follow and which to abandon, given a large number of choices. According to Conklin, these two problems “may ultimately limit the usefulness of hypertext.”

In order to solve or minimize these two problems, considerable amount of research efforts have been done or are underway. Methods to tackle the first problem include the provision of various overview diagrams, paths, guided tours, history lists, and so on. These methods are also effective to reduce the second problem, as the two problems are closely related. Besides, it is commonly agreed that the simplicity and consistency of the hypertext structure as well as providing a set of node and link types to be selected help in reducing the cognitive overhead. Related to this research, the structural and contextual information embedded in the hypertext can be utilized to improve the query and search in a system.

2.2.2.3 Dexter Hypertext Reference Model

There exist several different hypertext models that address different aspects of hypertext. Amongst these, the Dexter Hypertext Reference Model is widely accepted. This model captures the important abstractions found in a wide range of existing and future hypertext systems [Halasz&Schwartz 1990; 1994], and has influenced the design of the new Web standards, mainly XML [XML 1.0] and RDF [RDF Spec].

The Dexter model focuses on the description of the network of nodes and links. In the model, hypertext systems have three layers [see Figure 2.3]:

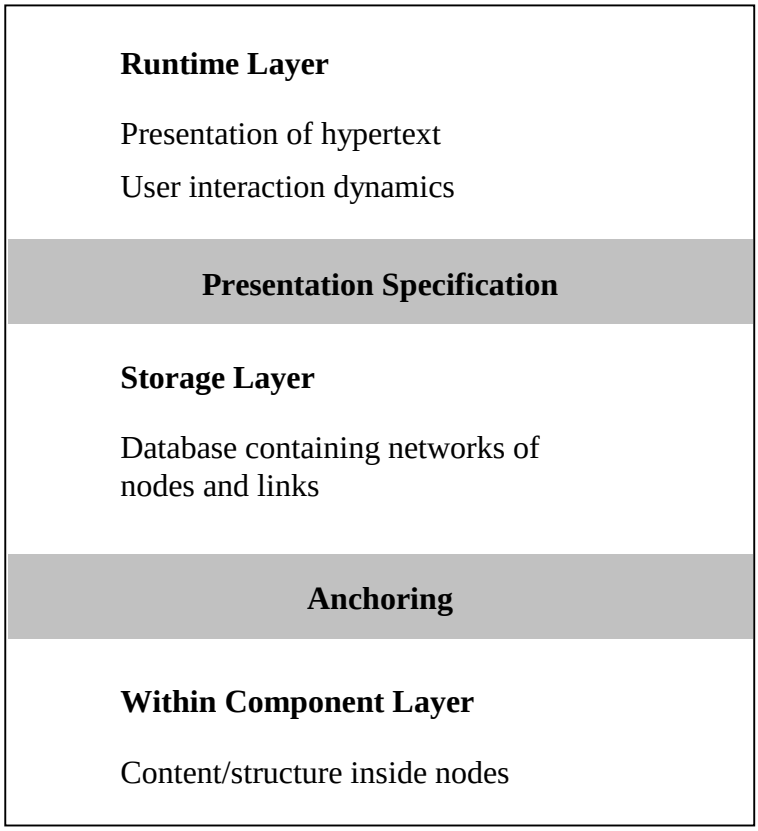


Figure 2.3 Dexter model layers [Halasz&Schwartz 1990; 1994]

- ***Runtime Layer***

This layer deals with the presentation of hypertext and the dynamics of user interaction. Since it is too broad and diverse to be developed into a generic model, the Dexter model does not go into the details of the presentation mechanism. However,

the presentation mechanism can be specified containing information about how a component/network is to be presented to the user. These presentation specifications provide an interface between the runtime layer and the storage layer.

- ***Storage Layer***

This layer is the main focus of the Dexter model. It models a database that is composed of a hierarchy of nodes (called components in the model) that are interconnected by relational links. Nodes may be composites of other nodes. Links (also seen as a kind of *components* in the model) may connect any number of nodes. Each node (either atomic or composite) and link may have arbitrary many attribute-value pairs, which may be used to attach any arbitrary property (and its value) (e.g. keywords) to the node or link and thus enable the implementation of e.g. a node or link type system. The description of each node includes pointers to the exact locations to which its links connect.

- ***Within Component Layer***

This layer covers the contents and structures within hypermedia nodes. Since the range of possible content / structure that can be included in a node is open-ended, the Dexter model treats this layer as being outside its scope. The assumption is that document structure models such as ODA [ISO 8613], SGML [ISO 8879:1986], etc., will be used in conjunction with this model to capture content, structure and semantics. However, a critical interface between the storage layer and the within-component layer called ***Anchoring*** discusses the mechanism of addressing locations or items within the content of an individual node. Anchors can be identified by an unique anchor identifier.

The goal of the Dexter model is to provide a systematic basis for comparing systems and to develop interchange and interoperability standards. The model has been used in developing the Dexter Interchange Format, a hypertext interchange standard. Efforts to use the Dexter model to augment the WWW have also been seen [Gronbak et al. 1997].

2.2.2.4 Semantic Net and Hypertext

A semantic net or semantic network [Quillian 1968] is a knowledge representation schema consisting of a directed graph in which concepts are represented as nodes, and relations between concepts are represented as links. As there is a striking similarity of hypertext and semantic net due to the common elements of nodes and links, the analogy of semantic net and hypertext is straightforward and has long been recognized [Conklin 1987; Rada 1990].

Related to hypertext, semantic nets can be divided into two categories*: independent semantic nets* and *embedded semantic nets*. In the latter case, links are embedded in a document chunk. In traversing such an embedded semantic net hypertext, users have to visit document chunks. One typical example is the traditional World Wide Web. Because of embedding links within documents, a number of serious long-term drawbacks exist [Andrews 1996]. Such drawbacks are:

- Only unidirectional links (forward) are possible; graphical link overviews are difficult.
- There is no automatic link maintenance, which leads to dangling links when documents are moved or deleted.
- Linking from read-only documents (e.g. documents on CD-ROM, or to which one has no write access) is not generally possible.
- It is difficult to link from arbitrary multimedia documents, since one cannot easily store links within the file format of, say, JPEG images or MPEG video clips.
- It is difficult to implement additional features such as link access rights, multiple webs, and automatic link generation.

In contrast, in an *independent semantic net*, links are kept external to documents. Though this may involve some more work to maintain them, all the drawbacks as described above are removed. Links can be bi-directional, from read-only documents and from arbitrary document types. Automatic link maintenance and integrity are possible. Also, it is easy to implement additional features such as link access rights and so on.

In practice, some hypertext systems, e.g. the Intermedia system at Brown University [Haan et al. 1992], Microcosm [Davis et al. 1992; Fountain et al. 1990] and Hyper-G [Andrews et al. 1995], have used an external link database to keep links, although the researchers may have never mentioned semantic networks in their work. As for the Web, new standards, e.g. XML Linking Language (XLink) [XLink], allow the representation of the hypertexts with an independent semantic net and further enable the use of new search methods applying the information contained in the net. Any documents, either text or multimedia, can be retrieved based on link structures. This thesis is an effort to explore several such methods.

2.2.3 WWW Issues

2.2.3.1 Limitations of the Traditional Web

The World Wide Web (WWW) has spread the hypertext paradigm to a mass audience for the first time. The key idea behind the Web is the hypertext navigational paradigm set in a distributed environment. From a hypertext researcher's view, the traditional Web is only a very limited basic hypertext application.

According to [Andrews 1996], two most serious limitations existing in the traditional Web are:

- The embedding of hyperlinks within documents
- The lack of any structuring mechanisms beyond the hyperlinks

The problems caused by the first limitation can be seen in the subsection 2.2.2.4 "Semantic Net and Hypertext". The second limitation leads inevitably to user disorientation or "lost in hyperspace" syndrome, one of the two traditional hypertext challenges. Furthermore, the search engines on the Web can only be keyword-based. Little structural information can be applied in their search algorithms.

2.2.3.2 Document Representation – HTML and/or XML

HTML

HTML (Hypertext Markup Language) (RFC 1866) [HTML] is the most popular application of SGML, which is an ISO standard [ISO 8879:1986] that uses tagging (markup) to describe the logical structure of a document. It is a simple markup language used to create hypertext documents that are platform independent and has been used in the Web since 1990. Together with HTTP, the transport mechanism between Web clients and servers, it has led directly to the success of the Web, because of their initial simplicity.

On the other hand, while simple, HTML is limited in its expressiveness and thus cannot suffice for all publishers' needs. This leads to "Balkanization of HTML", namely, many dialects of HTML have been emerging. Publishers have to face the difficult no-win task of deciding which version of HTML to support. This is against the basic information exchange rules, and also leads directly to advocate using XML [XML].

However, it is popularly thought that XML itself does not replace HTML. HTML is expected to remain in common use for some time to come.

XML

XML (Extensible Markup Language) [XML 1.0] is an application profile or restricted form of SGML. The goal of it is to enable generic SGML to be saved, received, and processed on the Web in the way that is possible with HTML. It has been designed for ease of implementation and for interoperability with both SGML and HTML.

Like SGML, XML also uses a so-called Document Type Definition (DTD) to define the tags needed to describe a specific document structure and the order in which these tags should appear in a document instance. It thus offers:

- *Extensibility* -- can define new elements, containers, attribute names
- *Structure* -- a DTD can constrain the information model of a document
- *Validation* -- every document can be validated; also, *well-formedness* can establish conformance to the structure mandated by the DTD.

In addition, it includes extensible linking and style formatting. This refers to XLink [XLink], XPointer [XPointer] and XSL [XSL]. Among them, XSL is primarily

targeted for highly structured XML data that e.g. needs element reordering before presentation and has little to do with this thesis.

The relationship among XML, DTDs and document instances is illustrated in Figure 2.4.

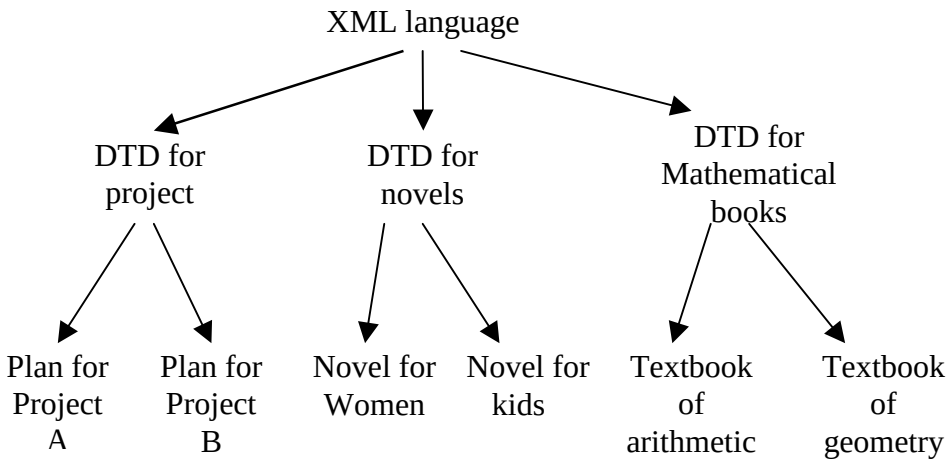


Figure 2.4 XML, DTDs, and document instances

2.2.3.3 Expressing Semantics – RDF & RDF Schemas

RDF (Resource Description Framework) [RDF Spec] is a foundation for processing metadata. It provides interoperability between applications that exchange machine-understandable information on the Web.

The foundation of RDF is a model for representing named properties and property values. The RDF data model consists of three object types: *resources*, *properties* and *statements*. A *statement* specifies for a resource a value (literal or another resource) for a *property*. A *resource* is any object that is uniquely identifiable by a Uniform Resource Identifier (URI) [URI1 1994; URI2 1995] and has properties (attributes or characteristics). It may be e.g. an entire Web page, a part of a page, or a whole collection of pages. It may also be an object that is not directly accessible via the Web, e.g. a printed book, an abstract concept, and so on.

RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent

relationships between resources and an RDF model can therefore resemble an entity-relationship diagram.

However, the RDF data model provides no mechanism for declaring the properties, nor does it provide any mechanism for defining the relationships between these properties and other resources. That is the void that **RDF Schema** [RDF Schema] seeks to fill.

RDF Schema [RDF Schema] provides for particular domain community a mechanism to declare *vocabularies*, the sets of semantic property-types used in application-specific RDF models. More succinctly, the RDF Schema mechanism provides a basic type system for use in RDF models. It defines resources and properties such as rdfs:Class and rdfs:subClassOf that are used in specifying application-specific schemas.

The RDF Schema mechanism is itself specified as a set of RDF resources (including properties), and constraints on their relationships. Thus, RDF schemas are themselves instances of RDF data models and are entity-relationship (ER) diagrams. An RDF schema can be seen as a description of a domain model.

2.2.3.4 Semantic Web

The term “Semantic Web” was coined by Tim Berners-Lee in 1998 [Berners-Lee 1998]. It is to denote the efforts dealing with the conceptual structuring of the Web in an explicit and machine-readable way. The primary goal of the Semantic Web is to facilitate resource discovery, information filtering and “intelligent browsing”. XML (Extensible Markup Language) [XML] and RDF (Resource Description Framework) [RDF] are two base technologies created and promoted by W3C [W3C] for enabling the Semantic Web.

Accompanying W3C’s effort, much work has been done with respect to the models, architectures, and management aspects for the future semantic Web [Semantic Web], and a number of commercial vendors are preparing XML and RDF software tools [RDF; XML]. Famous projects in this area include

- Ontobroker [Ontobroker] / Ontoserver [Ontoserver],

- SHOE [SHOE],
- Desire [DESIRE],
- OIL – Ontology Interchange Language [OIL],
- OntoAgents [OntoAgents],
- etc.

These projects cover mainly the research fields of

- ontology construction tools (e.g. Ontolingua server [Farquhar et al. 1997], KAON [Bozsak et al. 2002], OilEd [Bechhofer et al. 2001]),
- Web page annotation tools (e.g. CREAM [Handschuh&Staab 2002], Annotea [Kahan et al. 2001]),
- metadata storage and inference engines (e.g. Ontobroker's [Decker et al. 1999] underlying inference engine SilRL [Decker et al. 1998], DAML+OIL FACT reasoner [Horrocks 1998; Broekstra et al. 2001], or a fact serving peer [Nejdl et al. 2002]).

Here ontology is a knowledge representation that has been defined differently by different authors. The most popularly accepted definition is given by Gruber [Gruber 1993], who defines ontology as a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. From this definition, we see ontologies as typical examples of domain models that are defined in this thesis. The work listed above is mainly related to the representation of domain models, their distributions and general storage and processing operations.

In spite of so much ongoing work, the vision of the future Semantic Web is not yet clear. In the keynote session at the XML 2000 conference, Tim Berners-Lee outlined his vision for the Semantic Web. He explained the layered architecture (as shown in Figure 2.5) that he foresaw being developed in the next ten years.

According to Berners-Lee, the word "semantic" in the context of the Semantic Web means "machine processable". The sense of natural language semantics is explicitly ruled out. For data, the semantics convey what a machine can do with that data. In the future, it is anticipated that they will enable a machine to figure out how to convert

that data too. The Semantic Web is, like XML, a declarative environment, where one says what one means by some data, and not what one wants done with it.

Having outlined the scope of the Semantic Web, Berners-Lee explains the importance of RDF/RDF Schema as a language for the description of "things" (resource) and their types. Above this, he describes the layer of ontology. From his point of view, an ontology is capable of describing relationships between types of things, such as "this is a transitive property", but does not convey any information about how to use those relationships computationally. On top of the ontology layer sits the logic layer, which is the point at which assertions from around the Web can be used to derive new knowledge. Rather than designing one overarching reasoning system, Berners-Lee instead suggests a universal language for representing proofs. Systems can then digitally sign and export these proofs for other systems to use and

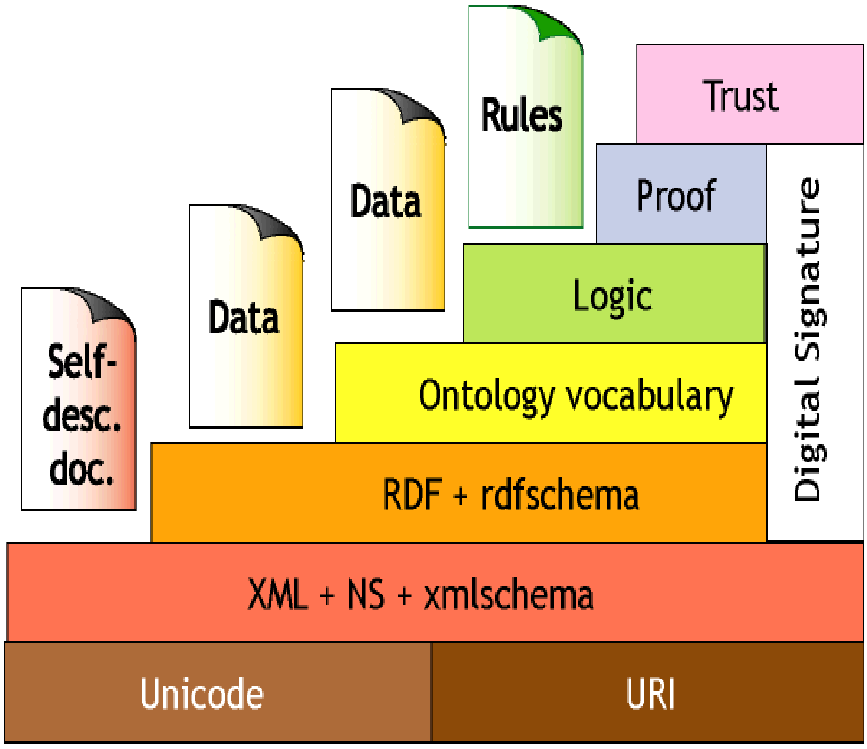


Figure 2.5 The Semantic Web "layer cake" presented by Tim Berners-Lee at the XML 2000 conference ***(****http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide1-0.html****).***

possibly incorporate into the Semantic Web.

Berners-Lee also observes that the higher layers of his architecture are likely to take around ten years to come to fruition – most of the new work today is happening on ontologies. Practical deployment includes RDF extraction & report generation using XSLT [XSLT], RDF & topic maps convergence, many general-purpose RDF databases and engines, and generic and specific GUIs for RDF data. He also notes that a rearrangement of the metadata activity within the W3C will have a bearing on Semantic Web work.

This thesis aims at developing methods that fully exploit the ever-enriched structural and semantic information that is representable with the new Web standards, mainly XML and RDF, for searching the Web and filtering and retrieving relevant information. It is among the basic research efforts towards the Semantic Web. The search methods it proposes will be able to be applied to any domain applications in the future Web hyperspace.

2.3 Hypertext Information Retrieval and Web Search Engines

2.3.1 Hypertext and Information Retrieval

Information Retrieval (IR) has concentrated on the development of information management systems to support information search in large collections of homogeneous textual material. Hypertext systems, on the other hand, provide a retrieval paradigm based on browsing through a structured information space, following pre-defined connections between information fragments until an information need is satisfied, or appears to be [Agosti&Smeaton 1996]. Much work has been done to apply techniques from one to the other between these two areas.

First, even from their beginning, hypertext systems have been a topic of great interest for the information retrieval research community as potential tools to be used to implement new information retrieval ideas and capabilities [Agosti 1996]. Link information has been used to inform retrieval algorithms. Early work in this direction is presented in e.g. [Croft&Turtle 1989], [Frisse&Cousins 1989], [Guinan&Smeaton

1992], [Frei&Stieger 1992], [Savoy 1993], [Savoy 1996]). More recent work is done mostly in the context of the Web, as described later in Section 2.4.

Second, information retrieval methods have been applied to hypertext systems in several ways. One important purpose of doing this is for the automatic or semi-automatic hypertext construction or link generation. For instance, IR techniques have been used to segment long articles into shorter, more focused nodes (e.g., [Salton&Allan 1993], [Salton et al. 1996], [Hearst 1994]). Term co–location information has been used to suggest links to human authors (e.g., [Bernstein 1990], [Chignell&Nordhausen 1991], [Robertson et al. 1994]). Similarity measurement has been used to create links between specific nodes (e.g. [Salton et al. 1994], [Green 1998]), or for link type inference (e.g., [Allan 1996; 1997]), or for representation and comparison of hypertext structures using graphics (e.g., [Furner et al. 1996]). Queries have been used to retrieve hypertext nodes, or used as navigational aids to identify relevant neighborhoods in the hypertext (e.g., [Crouch et al. 1989], [Egan et al. 1989], [Clitherow et al. 1989], [Christophides&Rizk 1994]). Relevance feedback has been used to guide retrieval and to infer links among documents (e.g., [Boy 1991]).

To satisfy different types of information needs, IR approaches and hypertext approaches need to be integrated [Wilkinson&Fuller 1996]. A so-called hypertext information retrieval (HIR) system is a tool that combines both the search facilities of an IR system together with the navigation and browsing facilities of a hypertext system [Agosti 1996]. End-users of such a system will be given the possibility to satisfy their information needs by using different retrieval techniques based upon value (a technique present in a highly specialized way in database management and IR systems), content (a feature mainly available in IR systems) and direct association (a possibility for the direct presentation through information connection as in hypertext systems). Aspects to be considered in the design and construction of efficient HIR systems are discussed by Agosti [Agosti 1996].

2.3.2 Query and Search Mechanisms for Hypertext

About the importance of information retrieval through queries for large hypertext databases, Halasz has claimed: “navigational access itself is not sufficient. Effective

access to information stored in a hypermedia network requires query-based access to complement navigation ……. search and query needs to be elevated to a primary access mechanism on par with navigation." [Halasz 1988]. Conklin suggested that search and query mechanisms for hypertext could present information at a manageable level of complexity and detail [Conklin 1987].

According to Halasz [Halasz 1988], query and search mechanisms for hypertext can be classified into *content search* and *structure search*. *Content search* is standard IR technique extended to hypertext systems. That is, all nodes and links are treated independently and examined for a match to the given query. *Structure search* will yield the hypertext sub-network that matches a given pattern. Query facilities that combine aspects of both content search and structure search will be capable of acting as filters. Based on the users' query, the interface will display only those nodes and links that match the query, while filtering out other parts of the network. Such filtered browsers have been implemented both for NoteCards [Halasz 1988] and Tektronix's Neptune [Delisle&Schwartz 1986]. In NoteCards, a user can filter out information based on the node or link types. In Neptune, the query can be content-based; if the query is broad enough, a global view of the entire network is displayed; if the query is well refined, the viewing size will be manageable.

In the hypertext literature, approaches to structure search and structural query languages have been proposed in [Consens&Mendelzon 1989], [Amann&Scholl 1992], [Lucarella et al. 1993], [Beeri&Kornatzky 1990], [Afrati&Koutras 1990], and [Christophides&Rizk 1994], etc. All these approaches assume that links are first-class objects and are thus not applicable to the traditional Web. But whether or to what degree they can be applied to the future Semantic Web is to be studied.

As far as we know, though the query and search issue for hypertext has been recognized as very important even from the beginning of hypertext research, this issue is far from sufficiently studied by the research community. However, as the Web grows, this issue is becoming more interesting for both the industrial and the research community. We see this in the large amount of completed or ongoing R&D work on Web search engines.

2.3.3 Web Search Engines

The World Wide Web is a global hypertext environment. Searching in this environment inherits almost all the issues related to hypertext information retrieval. However, because of its huge scale and arbitrary structure, it creates many new challenges for the development of its searching capabilities. Even some techniques achieved in the HIR field cannot be easily applied in Web search systems.

The traditional Web lacks explicit structural meta information [Trigg 1996]. The search engines on it are thus typically keyword-based. With such engines, people usually get a large number of pages that they cannot process, or even worse, many of the pages are totally irrelevant to their information needs, especially when they search for information on specific topics.

As a solution to the problem, several kinds of search limitation mechanisms have been supported in some of the most popular systems. One can use Yahoo [Yahoo] to search only one category at a time. InfoSeek [InfoSeek] and Lycos [Lycos] support a "search within results" mechanism, as well as a so-called "find similar pages" function. Some support fielded search mechanisms, such as searching only in links, only in titles and so on. More work can be seen in less popular systems like WebGlimpse [WebGlimpse; Manber et al. 1997], which allows the search to be limited to the neighborhood (defined as all pages within a certain distance following outgoing links) of a current page.

Another approach is to make use of additional structural information to improve Web searching. Structural information has so far been used for enhancing relevance judgments, ranking WWW pages, or other purposes. Among the work in this area the achievements of Google [Brin&Page 1998; Page et al. 1998; Google] and Clever [Kleinberg 1998; Clever] are most attractive. Both systems use weighted link popularity as primary criteria in their ranking mechanism.

There are also discussions about search agents that will traverse the Web looking for specific information in real time [Sumner et al. 1998] or allow users to set up their own local searches [Miller&Bharat 1998]. For instance, the IRISWeb [Sumner et al. 1998] system enables users who have a valid access code to submit web addresses

(URLs) of seed pages and specify a depth to which these are traversed. The system then collects and indexes all resources linked from the seed page. After the process has been completed, the user is notified by e-mail and can search in the collection.

The World Wide Web Consortium (W3C) [W3C] has been working hard to create and promote base technologies for enabling the "Semantic Web" [Berners-Lee 1998] for several years. The new standards, especially XML (Extensible Markup Language) [XML] and RDF (Resource Description Framework) [RDF] improve the ability of expressing structures and semantics on the Web. It is anticipated that search methods applying more structural and semantic information that is representable with these new Web standards will help to improve the quality of searching the Web and filtering and retrieving relevant information. This thesis is also an effort to explore several such methods.

Being related to this thesis, some advanced approaches that take advantage of link structures for query and search purposes are reviewed in the next section.

2.4 Advanced Approaches Taking Advantage of Link Structures for Query and Search Purpose

2.4.1 Study in Hypertext Research

Approaches taking advantage of link structures have been studied in hypertext research that predates or is in parallel with the Web. For instance, Botafogo, Rivlin et al. [Botafogo et al. 1992; Rivlin et al. 1994] use basic graph-theoretic notions such as connectivity, as well as "compactness" measures based on link density and node-to-node distance, to identify clusters in the graph of a hypertext environment. Frisse [Frisse 1988] describes a method that is applicable in a tree-structured environment: the relevance of a page with respect to a query is also based on the relevance of its descendants in the tree. Frei and Stieger [Frei&Stieger 1992, 1995] have discussed how knowledge of the adjacency of nodes via hyperlinks can be used to help a user navigate or find the answer to a query. They annotated links with frequent terms from the source and target documents to enhance retrieval in UNIX manuals.

2.4.2 Study for the Web

There is a trend to integrate textual content and link information for the purpose of searching (especially for ranking and enhancing relevance judgments), organizing, visualizing, etc. on the Web. Most of the presented work on Web search does not take into account the semantic information in hyperstructures. The reason might be that they just focus on the traditional Web, in which very little semantic information in hyperstructures can be found, and new standards have not yet been widely adopted.

At first, Boyan et al. [Boyan et al. 1996] have observed that Web pages differ from general text in that they possess external and internal structure. They use this information to propagate rewards from interesting pages to those that point to them (also done by LaMacchia [LaMacchia 1996]). Iwazume et al. [Iwazume et al. 1996] have preferentially expanded hyperlinks containing keywords relevant to the user's query, although Leary [Leary 1996] observes that anchor text information can be unreliable.

Weiss et al. [Weiss et al. 1996] define similarity measures among pages in a hypertext environment based on the link structure. These measures are used in clustering hypertext documents (including link information) to structure a given information space for browsing and search activities.

Kaindl et al. [Kaindl et al. 1998] present a rudimentary form of structure search that is based upon content search to improve the precision of Web searches. Arocena et al. [Arocena et al. 1997] describe frameworks for constructing WWW queries from a combination of term-matching and link-based predicates. Both of these approaches should be able to benefit from the Connectivity Server [Bharat et al. 1998], which provides linkage information for all pages indexed by the AltaVista search engine.

Some approaches take into account link information and connectivity in their evaluation function used for ranking search results. Underlying most of the link-based ranking algorithms, there is the following *recommendation assumption* [Henzinger 2000]: a hyperlink in page *X* pointed to page *Y* stands for the recommendation of page *Y* by the author of page *X*. In more general speaking, the whole Web is a citation graph and each hyperlink represents a citation or recommendation relationship.

Carriere and Kazman [Carriere&Kazman 1997] propose a link-based method for visualizing and ranking the results of queries returned by WWW search engines. In their WebQuery system, the connectivity of a node is defined as the node's total number of incoming and outgoing links, and the highest rank is given to the most highly connected nodes (without regard to the directions of links).

Marchiori [Marchiori 1997] presents a method to improve the quality of search engines by considering the "hyper information" of Web objects. He defines the "hyper information" of a Web object to be the dynamic informative content that is provided by hyperlinks. If in the "hyper information" the same content occurs, a Web object is more relevant regarding the queried content.

Kleinberg [Kleinberg 1998] presents a model of the Web as hubs and authorities, based on an eigenvector calculation on the co-citation matrix of the Web. In his view, hubs and authorities exhibit a so-called mutually reinforcing relationship: a good hub is a page that points to many good authorities; a good authority is a page that is pointed to by many good hubs. His algorithm HITS first uses a standard text search engine to gather a "root set" of pages matching the query subject. Next, it adds to the pool all pages pointing to or pointed to by the root set. Thereafter, it uses only the links between these pages to distil the best authorities and hubs. Based on this algorithm, the Clever system [Clever] was developed. This system ranks pages primarily by measuring links between them and additionally using the content of the Web pages. It thus exploits not only the link structure but also the text and other properties of the web pages being distilled. Compared to the work of Carriere and Kazman [Carriere&Kazman 1997], it makes crucial use of the directionality of hyperlinks.

Several researchers have extended or modified the original HITS algorithms. For instance, the ARC algorithm of Chakrabarti et al. [Chakrabarti et al. 1998] has enhanced the HITS algorithm with textual analysis. They present a method for the automated compilation of resource lists, based on a combination of anchor text and link analysis for distilling authoritative Web resources. Bharat et al. [Bharat et al. 1998] introduce additional heuristics to HITS algorithm by giving a document an authority weight and a hub weight according to the number of links from or to the document. Dean and Henzinger [Dean&Henzinger 1999] propose an improved HITS

algorithm and apply the algorithm to find relevant pages in the Web. Li et al. [Li et al. 2002] propose a new weighed HITS-based method that assigns appropriate weights to in-links of root Web pages and combine content analysis with the HITS-based algorithm.

Page et al. [Brin&Page 1998; Page et al. 1998] present a static ranking schema PageRank for assigning a universal "rank" to each page on the WWW, so that subsequent user searches can be focused on highly ranked pages. The rank of a page is based on the number of other pages linking to it and the importance of those pages. Importance, as with Clever [Clever], is derived from an overall link count. Based on their method, they have also developed a search engine Google [Google], which employs a number of techniques to improve search quality including page rank, anchor text, and proximity information. An important difference from Clever is that Google actually crawls the web itself, rather than analyzing a core set of pages from another search engine. Thus, its results should be more comprehensive.

Some work goes in the direction to improving PageRank or adapting PageRank to some specific scenarios. For instance, Rafiei et al. [Rafiei et al. 2000] propose using the set of Web pages that contain some terms as a bias set for influencing the PageRank computation, with the goal of returning terms for which a given page has a high reputation. Richardson and Domingos [Richardson&Domingos 2002] introduce a model that probabilistically combine page content and link structure in the form of an intelligent random surfer. Haveliwala [Haveliwala 2002] has proposed computing a set of PageRank vectors to capture more accurately the notion of importance with respect to a particular topic. Xue et al. [Xue et al. 2003] propose an approach to constructing implicit links by mining users' access patterns, and then apply a modified PageRank algorithm to re-rank Web pages for small Web search. They claim that the *recommendation assumption* [Henzinger 2000] is generally correct for the global Web but invalid in the case of a small web, in which most links are used to organize the content of a site into a hierarchical or linear structure.

Finally, there exists work that proposes approaches to automatically constructing domain-specific thesauri from the Web using link structure information and applying the constructed thesauri on query expansion to improve search precision [Chen et al. 2003]. Some work in the field deals with frequently occurring structures in the Web

for search purpose or for classifying / categorizing Web pages or sites (e.g. [Pirolli et al. 1996; Spertus 1997; Mizuuchi&Tajima 1999; Tajima et al. 1999; Chakrabarti et al. 2002; Li-W et al. 2002; Amitay et al. 2003; Eiron&McCurley 2003]). Some of these structures or Web page classifications / categories can serve as an initial set of node or link types for the Web, and/or as the basis for computing high-level hypermedia structures as defined in this thesis.

2.4.3 Other Related Studies

The use of links for ranking documents is similar to work on citation analysis in the field of bibliometrics, which studies the patterns of citation – an implicit type of "linkage" – among scientific papers (see [White&McGain 1989] for a review). There has been work in bibliometrics on using citation counts to assess the "impact" of scientific journals. The classic work in the area is that of Garfield [Garfield 1972] (see also [Garfield 1994; Rousseau&Hooydonk 1996]).

The meaning of bibliometrics in the context of the WWW is studied in [Larson 1996], in which the author performs a co-citation analysis of a set of Web pages relevant to a sample query and generates "topical" clusters of WWW sites revealed by the analysis. Rousseau [Rousseau 1997] investigates the distribution of domain names and the distribution of links between web sites and introduces a notion of "sitation".

More recently, there exists also some work studying document links for ranking or other purposes from the IR research community. Especially, Justin Picard has shown a technique derived from logic and probability for integrating the evidence provided by document links [Picard 1998]. Norbert Fuhr has presented a probabilistic version of Datalog (pD) and shown its suitability for IR [Fuhr 2000]. Rölleke & Blömer have shown an application of the pD. They use a probabilistic link type space and uncertain links that yield a ranking of retrieved documents according to their links to documents that have been retrieved with respect to the content criteria of the query [Rölleke&Blömer 1997]. The CACM (Communications of ACM) collection has been used in all their experiments and the link types mainly considered were "citing" and "cited" relationship between documents. To what degree these models can be applied

directly to the Web is not clear, as in the Web space there exists a large number of pages and links of various types, and there are important differences between a scientific citation and a Web link [Efe et al. 2000].

3 Structural and Semantic Information in Hyperstructures & Its Standard Representation

This chapter describes structural and semantic information in hyperstructures and their standard representation. First, such information is introduced using terms from the hypertext research field. Then, its corresponding terms and the representations in the Web standards are presented.

3.1 Structural and Semantic Information in Hyperstructures

The structural and semantic information in hyperstructures can be classified into three different categories. The first category is basic hypermedia components and the types of these components, the second category is high-level hypermedia structures formed by these basic hypermedia components, and the third category includes various document models and domain models.

3.1.1 Hypermedia Components and their Types

The basic hypermedia components are nodes, links and anchors. The main semantic information about them is their types.

3.1.1.1 Nodes and Node Types

A node is a unit of information [W3C Terms]. It may contain any kind of data, such as a fragment of text, a graph, a picture, sound, or motion video sequence. A node type specifies a concept. In other words, it is the specification of the nature of the kind of information contained in the node.

Hypertext systems vary widely in their support for typed nodes. Node types may be pre-defined or user-defined. Some systems support special organization-oriented

node types to perform unusual tasks, such as to help manage the structure of a hypermedia graph. For example in NoteCards [Halasz 1988] there are special cards (alternative term for nodes) known as fileboxes (organizing nodes) that will present graphical representations of a selection of other cards. In this way, readers can understand immediately the context of one node among many.

In most cases, node typing is content-oriented and domain-specific or even application-specific or task-specific. For instance, a large part of the nodes in a Web site that sells books can be of the types "Author", "Book", "Comment" and "eBookshop". These node types exist in the domain model shown in Figure 3.3 as domain model concepts.

3.1.1.2 Links and Link Types

A link is a relationship between nodes. The meaning of a link is often indicated by the semantic type of the link. A link with a specified semantic type is a semantic link.

Semantic Links

Formally, a semantic link $\lambda \in \Lambda$ consists of the following components: [Frei&Stieger 1995]

$$\lambda = <t_\lambda, sn_\lambda, dn_\lambda, A_\lambda, c_\lambda>$$

where

$t_\lambda \in T_\Lambda$ is the link type

$sn_\lambda \in N$ denotes the source node of the link λ

$dn_\lambda \in N$ denotes the destination node of the link λ

$A_\lambda = \{a_{1\lambda}, a_{2\lambda}, \ldots., a_{k\lambda}\} \subseteq A_\Lambda$ is a set of structured link attributes

$c_\lambda \in A^*$ is a free text annotation to the link.

The link type t_λ serves to discriminate between several types of semantic links. It can be used to restrict a domain community to a few link types so that their semantics can be understood fully by both information providers and users in the domain. If we assume a closed hyperstructure, the source node sn_λ and the

destination dn_λ are members of the set of nodes N that constitutes this structure. The link information is composed of a structured part (the link attributes A_λ) and of an unstructured part c_λ.

In reality, for a link λ, it is imperative that the components sn_λ and dn_λ exist, while t_λ, A_λ and c_λ are not required. A link with no specification of t_λ, A_λ and c_λ is a non-semantic link, or can be seen as a special kind of semantic links.

As the value of A_λ and c_λ in ranking and filtering Web pages is not explored in this thesis, in the rest of the thesis a semantic link is simply denoted as $\lambda = <t_\lambda, sn_\lambda, dn_\lambda>$.

Link Types

Starting from early research on semantic networks [Woods 1985], typed links have long been a topic of interest. Many people have advocated typed links [Shum 1996; Mohageg 1992; Halasz 1991; Conklin 1987; DeRose 1989]. In 1983, Trigg [Trigg 1983] even listed a set of 80 classes of link types. Link types can help users and computers to distinguish various kinds of links, e.g., whether a link is a traversal connection, a structural means, or an argument representation (in general, to connect the premises of an argument to its conclusions) [Trigg 1996].

In the context of the Web, the typical nodes are Web pages or fragments in the pages (thus we use the terms *node* and *page* alternatively in the thesis). Links may exist between any of these nodes. Most of the links on the traditional Web are not typed. However, HTML 3.2 [HTML 3.2] and its later versions [HTML 4.0; HTML 4.01] define a set of generic link types to describe relationships of a given node with other nodes. The Dublin Core [Dublin Core] can also be seen as an ongoing effort to define a relation type system for this global hypertext system. It defines an element "RELATION" in its element set to provide a means to express relationships among resources (metaphor of nodes in RDF [RDF Spec]) that have formal relationships to others, but exist as discrete resources themselves, e.g. images in a document, chapters in a book, or items in a collection.

In most cases, link types are domain-specific and usually exist in certain domain models. It is anticipated that in addition to the above mentioned generic link type systems various domain-specific link type systems will also come into the Web as new Web standards become more common and some browsing and search systems appear to take into account this kind of information. This anticipation motivates our work in exploring the ranking and filtering mechanism that makes use of link types as described in Section 4.1.

3.1.1.3 Anchors

An anchor is an area within the content of a node that is the source or destination of a link. The anchor may be the whole of the node content [W3C Terms]. Anchors as defined in XPointer [XPointer] can specify different mechanisms to locate where a link starts and where a link ends. They can be substrings in character data, elements, and to whole tree fragments.

3.1.2 High-Level Hypermedia Structures

High-level hypermedia structures are structures formed by basic hypermedia components. This concept has been in common use since Halasz suggested that a composition mechanism should be added in the basic hypermedia model [Halasz 1988].

Two of the major high-level hypermedia structures are hypertext contexts and hypertext composites. They are all groups of nodes and links, however, the first without structural constraints, and the second with structural constraints (e.g. a tree or a directed acyclic graph).

3.1.2.1 Hypertext Contexts

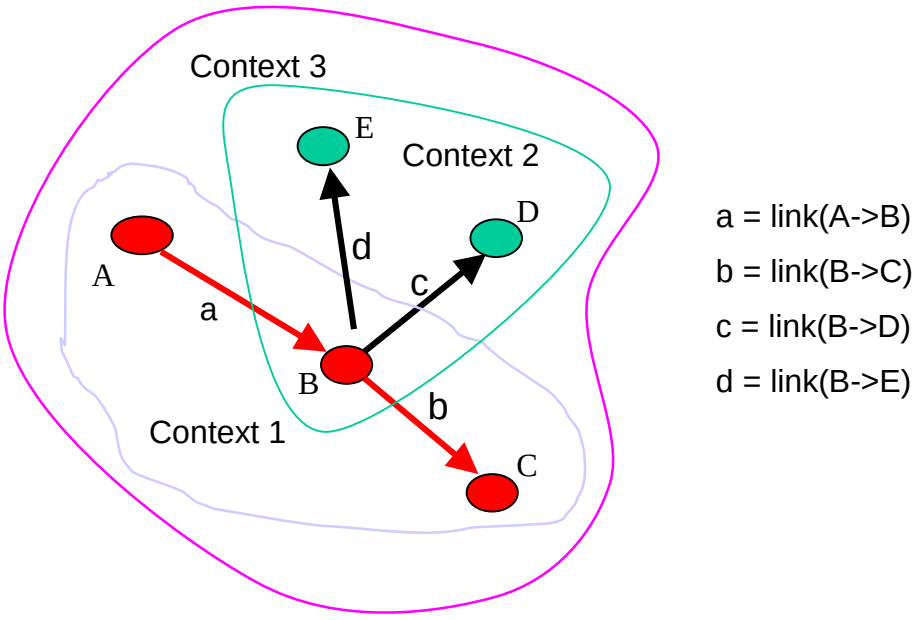
Concept

A *hypertext context* is a generic high-level hypermedia structure that groups together a set of nodes and links into a logical whole. The idea of *hypertext contexts* was first introduced by Schwartz and Delisle [Schwartz&Delisle 1987]. *Contexts*

partition the data within a hypertext graph. A hypertext graph contains one or more *contexts*; each *context* has one *parent context* and zero or more *child contexts* [Campbell&Goodman 1988].

Precisely, if C is a *hypertext context*, then its contents must define a pair (N, L), where N is a set of nodes in a hypertext graph and L is a set of links whose end nodes belong to N. We say that C contains a node *M* if *M* is in N and that C contains a link *l* if *l* is in L. *M* is a *node component* of C, while *l* is a *link component* of C.

If C1, C2 are two *hypertext contexts*, we say that C1 contains C2 only when all nodes and links in C2 are contained in C1. That is, if C1 defines a pair (N1, L1), C2 defines a pair (N2, L2), and C2 ⊆ C1, then N2⊆ N1 and L2⊆ L1. Contrary, if N2⊆ N1 and L2⊆ L1, then C2 ⊆ C1. We say C1 is the *parent context* of C2, and C2 is



Context 1 = ({<A>,<B>,<C>}, {<a>,<b>})

Context 2 = ({<B>,<D>,<E>}, {<c>,<d>})

Parent context: Context 3 ⊃ Context1, Context2

Context 3 = ({<A>,<B>,<C>,<D>,<E>}, {<a>, <b>,<c>,<d>})

Figure 3.1 Examples of hypertext contexts

one of the *child contexts* of C1.

This parent-child relationship between hypertext contexts is well illustrated with the examples shown in Figure 3.1.

Usually, a *hypertext context* is said to be a container for a group of nodes, while the links between the nodes are thought to be included implicitly in the context. We adopt this meaning of hypertext contexts later in this thesis. That is to say that unless clearly specified, the *components* of a *hypertext context* refer to the nodes contained in it.

In practice, *hypertext contexts* can be used to support configuration, private workspaces, and version history trees [Schwartz&Delisle 1987]. They can be used as a mechanism to describe different context views of the same hyperdocuments, tuned to different applications or classes of users of the documents [Casanova&Tucherman 1991]. In this sense, a (group of) hyperdocument(s) may contain any number of *hypertext contexts*. Such *hypertext contexts* can exist statically in hypertext document collections or can be created dynamically by hypertext based information systems.

Examples

Typical examples of hypertext contexts that exist statically in hypertext document collections are various maps, paths, guided tours, and focused node lists related to a particular topic or subject domain. These hypertext contexts are usually encoded in concrete nodes, maybe one context in one node, or several contexts in one node. Many such nodes (pages) exist on the traditional Web but cannot be recognized automatically. So the hypertext contexts described in the pages are.

For instance, the page “DELITE publications” (http://www.darmstadt.gmd.de/delite/Publications/) contains a complete list of all publications from the division DELITE of FHG-IPSI (http://www.ipsi.fhg.de/), and each item in the list points to a DELITE publication. What the list describes is actually a hypertext context that is composed of all nodes (pages) about DELITE publications. Moreover, it can be said that the items listed under each year constitute a child context of the above large one.

As for the hypertext contexts that are created dynamically by hypertext based information systems, the most typical examples are various search results, which are attained by computation against certain query criteria. Like static hypertext contexts, they cannot be recognized automatically on the current traditional Web. In addition to these typical examples, dynamic hypertext contexts can apparently be attained from Boolean operations performed on hypertext contexts existing in systems.

3.1.2.2 Hypertext Composites

In a hypertext system, the logical structures in hypertexts are usually supported by means of composites (groups of nodes and links) [Halasz&Schwartz 1994].

There are two kinds of hypertext composites: composite nodes that are composed by non-linking mechanisms (such as hierarchical structures in the structured documents defined by a DTD); and link-based composites that are composed by computation based on link types that represent containment (or part-of) relations. In most cases, especially in a single system, non-linking composition mechanisms are more efficient than link-based composition mechanisms [Carr et al. 1996]. However, there are also many cases where link-based constructs are desired [Gronbak&Trigg 1996].

This thesis intends to explore the value of link-based hypertext composites in Web searching. Thus, unless clearly specified, composites later in this thesis refer to link-based ones. A more formal description about such composites is first given below.

Formal Description of Link-based Hypertext Composites

A *link-based hypertext composite* is a special kind of node that is constructed out of other nodes and composites [Halasz&Schwartz 1994]. These nodes and composites are *components* of the composite. They are linked from the composite or the other components in the composite with containment (or part-of) relations. They may also link to each other with other types of relations. No link-based hypertext composite may contain itself either directly or indirectly.

To distinguish, the *components* that are atomic nodes are referred to as *atomic components* of the composite. The *components* that are composites are referred to as *compound components* of the composite. However, as a composite is a special kind of node, unless explicitly specified, the *node components* of a composite refer to both its *atomic components* and its *compound components*.

More precisely, suppose the *link (x->y)* denotes a link from a node *x* to a node *y*, and *link (*x-> *y).type* denotes the type of the link from *x* to *y*. Then, if C is a *link-based hypertext composite*, its contents must contain a pair (N_c, L_c), where N_c is a set of nodes in a hypertext graph and L_c is a set of semantic links whose endpoints belong to N_c, and C $\notin$ N_c. For any $n_1 \in N_c$, there exists a *link (*C-> n_1*)* and *link (*C-> n_1*).type* represents containment relations, or, there exists an $n_2 \in N_c$ so that

> *(1) link (*n_1-> n_2*)* $\in$ L_c *AND link (*n_1-> n_2*).type* represents containment relations,
>
> OR
>
> *(2) link (*n_2-> n_1*)* $\in$ L_c AND *link (*n_2-> n_1*).type* represents containment relations.

(1) represents the case when n_1 is a *compound component* of C, n_2 is a component of n_1 and thus also contained in C. (2) represents the inverse case. Either n_1 or n_2 can be reached from C by following links of the types representing containment relations.

We say that C contains a node *m* if *m* is in N_c and that C contains a link *l* if *l* is in L_c. Here *m* is a *node component* of C, while *l* is a *link component* of C.

Usually the *components* of a *hypertext composite* refer to its *node components*, as the meaning of the link components is mostly reflected in the building-up process of the composite. This usage of *components* is adopted later in this thesis.

Structural Levels of Link-based Hypertext Composites

One main use of a link-based hypertext composite is to organize a document in a hierarchical structure. By means of embedding compound components, the

composite organizes its components in different levels of the structure. The composite itself is in the first level, while its components are in the levels corresponding to their distances from the composite regarding the containment relations. A composite with *n* structural levels contains at least one *compound component* in the structural levels 2, … , n-2.

For instance, if C_1 is a composite with 3 structural levels, it should contain directly at least one compound component, e.g. a composite C_2. That is, C1 is at level 1, C2 is at level 2, and the components of C_2 are at level 3. The value of such structural levels can be explored for improving Web search, as proposed in this thesis (Section 4.3).

Hypertext Composites and Hypertext Contexts

Based on the above definition of hypertext composites, a hypertext composite can be seen as a special kind of hypertext context. A hypertext context can be used to combine nodes from one or more hypertext composites to describe a specific view of the nodes in a hypertext system.

A Simple Example

A typical kind of documents that can be organized by making use of a link-based hypertext composite is an online user manual, which has a hierarchical structure as its backbone, but also has other hypertext links within or across the document boundary. Such a document in a Web site usually has at least one URL that is

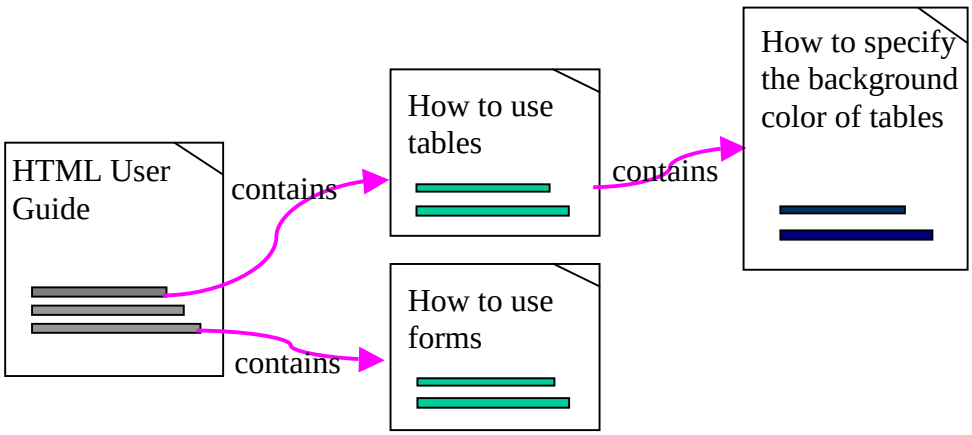


Figure 3.2 A simple composite "HTML User Guide"

distinguished as an entry point or leader. For documents to be read linearly, this entry point or leader is often a table of contents or title page [Mizuuchi&Tajima 1999; Eiron&McCurley 2003].

Figure 3.2 shows a simple hypertext composite “HTML User Guide” with 3 structural levels. It contains the nodes “How to use tables”, “How to use forms” and so on as its components. These nodes can be grouped into various hypertext contexts that represent the different views of the document that is organized hierarchically with the help of the composite.

3.1.3 Domain Models and Document Models

Domain models and document models are two kinds of hyperstructures that have domain specific semantics. From the structural point of view, they can be either hypertext composites or hypertext contexts as described above.

3.1.3.1 Domain Models

Three Level Domain Models

A domain is an area of specific activities, information, or knowledge containing applications that share a set of common functional capabilities and data. For information organization purposes, as in adaptive hypermedia systems, such a domain is usually modeled along a set of concepts, which represent elementary pieces of knowledge about the given domain and its applications. This set of domain concepts is usually referred to as the *domain model*. An independent set of concepts is the simplest form of a domain model [Brusilovsky 1996]. We call it a *level one model*.

A more advanced form of domain model is a hierarchical structure, in which domain concepts are organized hierarchically. They form parent-child and sibling relationships. We call such a model a *level two model*.

A *level three model* is a conceptual (semantic) network, in which the domain concepts are linked to each other (links represent the relationships between

concepts). In more complex domain models, the concepts and links can be typed [Brusilovsky 1996] and hence a semantic relationship between concepts is part of the domain model [Brusilovsky&Schwarz 1997].

In this thesis we refer to *level three domain models* as *link-based* (or *relationship-based*) *domain models.*

Link-Based Domain Models

Formally, a link-based domain model is a network with nodes corresponding to domain concepts and with links reflecting the relationships between concepts. Each concept may still have internal structure, which can be represented as a set of attributes where different kinds of topics usually have different sets of values.

Precisely, a link-based domain model D is defined as

$$D = (N, L)$$

where N is a set of nodes, L is a set of links. A_n is a set of attributes of a node $n \in N$.

If $n_1 \in N$, $n_2 \in N$, and $l_{1,2} \in L$ is a link from n_1 to n_2, we say n_2 is an *outgoing-neighborhood concept* of n_1, and n_1 is an *incoming-neighborhood concept* of n_2. Either an *outgoing-neighborhood concept* or an *incoming-neighborhood concept* is a *neighborhood concept*.

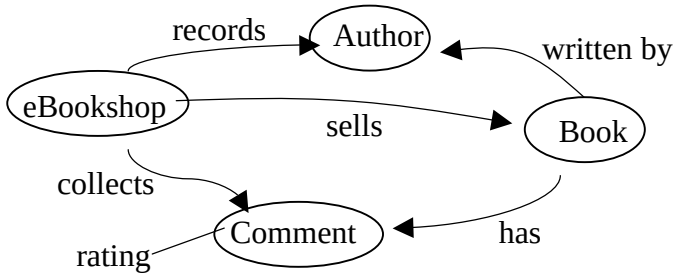


Figure 3.3 A link-based domain model

It is apparent that a *level one domain model* can be seen as a link-based domain model with a null set *L*, and a *level two domain model* is a link-based domain model with a set *L* containing only links representing hierarchical relationships.

A Simple Example

Figure 3.3 provides an example of a simplified link-based domain model used for organizing electronic commerce sites that sell books on the Web. The main concepts of the domain model are 'Book', 'Author', 'eBookshop', and 'Comment'. The relationship between the concepts indicates that "Book is written by Author", "eBookshop sells Book", "eBookshop records Author", "eBookshop collects Comment", and "Book has Comment". Each concept may also have some attributes. For instance, 'Comment' may have an attribute 'rating'.

3.1.3.2 Document Models

A document model specifies the allowed elements and the relationship between the elements in a kind of documents. Traditionally, a document model refers to a hierarchical structure specified by an SGML DTD (or an XML DTD or an XML Schema, see the subsection 3.2.2.7) for a structured document. That is to say that document models are usually specified by non-linking based composition mechanisms.

For link-based hyperdocuments, their document models can be seen as a special kind of domain model.

3.2 Standard Representation

3.2.1 Representation in Existing Hypermedia Systems and Models

Some of the above mentioned structural and semantic information has already been expressed in existing hypermedia systems and models, such as the Dexter model [Halasz&Schwartz 1990, 1994], Hyper-G [Andrews et al. 1995], and Microcosm [Davis et al. 1992; Fountain et al. 1990]. These models and systems

have influenced the design of the new Web standards, such as XML [XML 1.0] and RDF [RDF Spec], which provide the Web with new abilities of expressing structures and semantics of Web resources.

Of all the existing hypertext models, the Dexter model is best known. It captures the important abstractions found in a wide range of hypertext systems. It supports composites of nodes (referred to as composite components in the model). In it, each node (either atomic or composite) and link may have arbitrarily many attribute-value pairs, which may be used to attach any arbitrary property (and its value) (e.g. keywords) to the node or link and thus enable the implementation of a node or link type system. Note that here a node may be of type composite node, which is one of the main high-level hypermedia structures mentioned above.

Hyper-G and Microcosm are both systems that use an external link database to manage links. In Microcosm, both links and nodes (referred to as documents) may have user-defined attributes. These attributes enable authors to attach keywords and descriptions to the nodes and links, and thus make node typing and link typing possible. In Hyper-G, which has once been considered to be a model for the next generation WWW system, each user has the rights to read, create, modify and annotate links. This enables the implementation of a link type system.

3.2.2 Representation with New Web Standards

3.2.2.1 Representation of Nodes and Node Types

On the Web, any resources that can be specified by an URI can be seen as nodes. The type of a node can be represented with the *role* attribute of a locator in an XLink. A simple example is given as follows:

```
<mylink  xml:link="extended" inline="false">
   <locator href="doc1.xml" role="definition"/>
</mylink>
```

The elements in XML documents can also be seen as nodes. The node types are described as element types, which are encoded in XML tags. For instance, the

following example of an XML document encodes 4 node types – *document*, *title*, *author*, and *content*:

<document><title>database</title><author>Smith</author><content>
database is ...</content></document>

Note that the node of the *document* type is in fact a hypertext composite, as its content is composed of other nodes.

3.2.2.2 Representation of Links and Link Types

Links and link types can be represented in HTML, XLink, and RDF.

HTML

In HTML 2.0 and its later versions [HTML 3.2; HTML 4.0; HTML 4.01], links can be encoded in two kinds of elements. The first kind is the *A* element, which defines a link that can be traversed and may only appear in the body of a document. The second kind is the *LINK* element, which defines a relationship between the current document and another resource and may only appear in the head of a document. The types of the links are represented in their *rel* and *rev* attributes.

The *Rel* attribute specifies a forward link from the current document to the anchor specified by the href attribute, while the *Rev* attribute specifies a reverse link from the anchor specified by the href attribute to the current document. The value of these attributes is a space-separated list of link types, which are defined in HTML itself or by users (Only HTML 4.0 and 4.01 permits user-defined link types).

The following example illustrates how links are encoded in the *A* element and how links and link types are encoded in LINK definitions:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html40/strict.dtd">
<HTML>
<HEAD>
<TITLE>Chapter 2</TITLE>
<LINK rel="Index" href="../index.html">

<LINK rel="Next" href="Chapter3.html">

<LINK rev="Next" href="Chapter1.html">

</HEAD>

<BODY>

<A href="toc.html">Table of Contents</A>

...the rest of the document...

XLink

In XLink, links are encoded in linking elements. The types of links can be encoded via the *role* attributes of linking elements. The *role* attribute identifies to application software the meaning of the link. The values of this attribute are of the kind CDATA. They may be predefined in DTDs (fixed) or specified in documents (no default value is provided in DTDs).

Following is an example of an out-of-line extended xlink:

<mylink xml:link="extended" inline="false" role="commentary">
<locator href="smith2.1" role="Essay"/>
<locator href="jones1.4" role="Rebuttal"/>
<locator href="robin3.2" role="Comparison"/>
</mylink>

RDF

In RDF [RDF Spec], a statement may specify a named relationship between two resources. In a broad sense, such a named relationship can be regarded as a typed link. The link types can be encoded in the *propertyElt* element via its tag (property name) and *resource* attribute. As illustrated in the following example, http:doc2 is a part of http:doc1. Using Dublin Core [Dublin Core] vocabularies, the link type isPartOf is represented by the tag *<dcterms:isPartOf>*:

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc ="http://purl.org/dc/elements/1.1/"

```
xmlns:dcterms =“http://purl.org/dc/terms/”>
<rdf:Description about=“http:doc2”>
   <dc:title>doc2</dc:title>
   <dcterms:isPartOf rdf:resource=“http:doc1”/>
</rdf:Description>
</rdf:RDF>
```

As shown in the example, property names must always be associated with a schema, which represents a domain model or ontology, such as Dublin Core. This is consistent to what we have mentioned before, i.e., link types are often domain specific.

3.2.2.3 Representation of Anchors

Anchors can be encoded in both HTML and XML.

In HTML, there is only one kind of anchor, which is typically represented by explicit markup, i.e. <A NAME = “ ”>. For instance, here is a URI pointing to an anchor named section_2: *http://somesite.com/html/top.html#section_2.*

In XML, anchors can be described with various location terms provided by XPointer [XPointer]. The terms are classified into absolute terms, relative terms, span terms, attribute terms, and string terms. They provide for specific reference to elements, character strings, and other parts of XML documents, whether or not they bear an explicit ID attributes.

In the following example, the first XPointer identifies the 29th paragraph of the 3rd sub-division of the 5th major division of a location source, which can be an element or location in an XML document, or simply the entire document. The second selects the first child of the location source for which the attribute TARGET has a value. And the third selects the second through the fourth child of the element with ID a5:

Child(5, DIV1).child(3, DIV2).child(29, P)
*Child(1, #element, TARGET, *)*
Id(a5).span(child(2), child(4))

3.2.2.4 Representation of Hypertext Contexts

This thesis suggests applying, e.g., XML extended links and RDF containers as defined within the XML and RDF standards for specifying hypertext contexts. By this means, sharing and reuse of hypertext contexts is generally supported.

Representing Hypertext Contexts with XML Extended Links

In the XML model, the linking mechanisms supported are specified in the XML Linking Language (XLink) [XLink]. An extended link is a link that associates an arbitrary number of resources (an example is shown in Figure 3.4). A hypertext context that contains Web resources as its components can be described with a linking element for an extended link. Each component of the hypertext context is given in a locator element, which is a child element of the linking element.

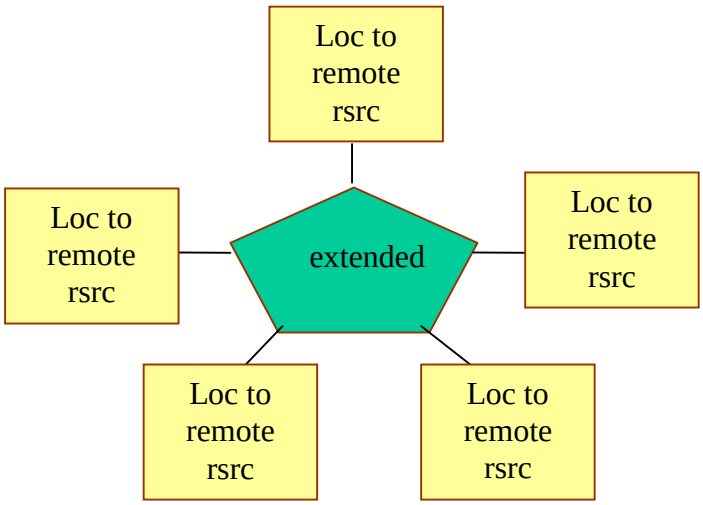


Figure 3.4 XML extended links (an example)

For instance, the following encoding (DTD and an out-of-line extended link) describes a hypertext context *mycontext*:

<!ELEMENT mycontext (component+)>

```
<!ATTLIST mycontext  xmlns:xlink CDATA #FIXED
    http://www.w3.org/1999/xlink
                xlink:type   (extended)   #FIXED "extended"
                xlink:title   CDATA       #REQUIRED
                xlink:role    CDATA       #IMPLIED>
<!ELEMENT component EMPTY>
    <!ATTLIST component xlink:type (locator)  #FIXED "locator"
                xlink:href    CDATA        #REQUIRED
                xlink:title    CDATA       #REQUIRED
                xlink:role     CDATA       #IMPLIED>

    <mycontext xlink:title="People working in the project DELITE-online">
        <component xlink:href="http://www.darmstadt.gmd.de/~qiu"
            xlink:title="Zhanzi Qiu"/>
        <component xlink:href="http://www.darmstadt.gmd.de/~moelle"
            xlink:title="Karin Moelle"/>
        ...
    </mycontext>
```

A locator may indicate a resource, which itself contains an extended link, i.e., also describes a hypertext context. In this way, the *parent-child* relationship between hypertext contexts can be represented.

Especially, hypertext contexts and their parent-child relationship may be described with extended link group elements (a special kind of extended links) and extended link document elements (a special kind of locator elements). The Steps attribute of the extended link group elements can be given a numeric value that serves as a hint from the author to any system as to how many levels hypertext contexts exist.

That is, an extended link group element may be used to store a list of links to other resources that together constitute an interlinked group. Each such resource is identified by means of an extended link document element and may itself contain an extended link. In this case, the group element describes a hypertext context that is

the parent of the hypertext contexts described in the resources indicated by the document elements.

To give an example, suppose the above descriptions about *mycontext* are stored in *mycontext.xml*, a possible parent context of *mycontext* can be described as follows:

```
<group xml:link="group" steps=2>
<document xml:link="document" href="mycontext.xml"
role="recommend"/>
<document xml:link="document" href="..." role="..."/>
...
</group>
```

These descriptions are contained in a document other than *mycontext.xml*.

Representing Hypertext Contexts with RDF Containers

In the RDF model, a hypertext context can be represented in a container. Each component (node) of the context is referred to with a resource. (In RDF, the term *resource* is in most cases a metaphor of *node*).

RDF defines three types of container objects: bag, sequence, and alternative. The first two are used to declare the multiple values of a property, and the third is to declare alternatives for the (single) value of a property. For representing hypertext contexts, the first two types of containers fit better. Besides, the difference between them, i.e., one declares unordered lists and another declares ordered lists, does not make much sense, as the sequence of components in a hypertext context is usually unimportant.

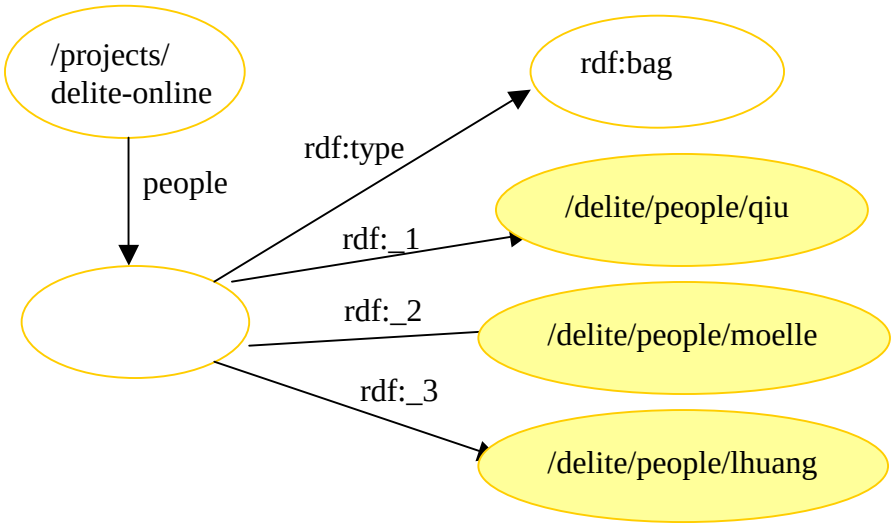
For example, a hypertext context that consists of resources about people working in the project *DELITE -Online* can be described as follows:

```
<rdf:RDF><rdf:Description
about="http://www.darmstadt.gmd.de/delite/projects/delite-online.html">
<s:people><rdf:Bag>
<rdf:li resource="http://www.darmstadt.gmd.de/~qiu/">
```

```
    <rdf:li resource="http://www.darmstadt.gmd.de/~lhuang/">
    <rdf:li resource="http://www.darmstadt.gmd.de/~moelle/">
  </rdf:Bag></s:people>
</rdf:Description></rdf:RDF>
```

Figure 3.5 illustrated the RDF bag encoded above. Such RDF descriptions can exist in a separate RDF document, or be contained in the head of an HTML document.

It is clear that by embedding an RDF container in another RDF container, the parent-child relationship between hypertext contexts can be represented.



Resources in such a container can be regarded as components of a hypertext context

Figure 3.5 A simple RDF bag container describing a hypertext context

3.2.2.5 Representation of Hypertext Composites

The traditional Web has only a single node type called the page. All pages are equally accessible in a "flat" pool [Gronbak&Trigg 1996]. Something like the effect of composites can be obtained using pages full of URLs, but true structuring composites are not supported.

This situation has been changed due to some new Web standards. As we have discussed in the subsection 3.2.2.2, links and link types can now be encoded in HTML, XLink, and RDF. If the containment relations between Web documents are represented with special link types, application systems can then make use of these link types to compute link-based composites.

As for HTML, in addition to defining the LINK element, HTML 4.01 [HTML 4.01] has also defined a set of link types permitted in the documents. Among the predefined set the link types such as *contents*, *chapter*, *section*, and *subsection* express the containment relation between Web documents and can be used to compute hypertext composites. Further, HTML 4.01 also allows users to define additional link types by using a metadata profile to cite the conventions used to define the link types.

For instance, in Figure 3.2, suppose the document for "How to use tables" is "table.html" and the document for "How to use forms" is "form.html". They both are chapters of a document collection for "HTML User Guide", whose table of contents is in "HTML.html". Then "HTML.html" may contain the following example encoding for describing the containment relation:

```
<HEAD> … other head information...
    <LINK rel="chapter" href="table.html">
    <LINK rel="chapter" href="form.html">
    <LINK rev="contents" href="table.html">
    <LINK rev="contents" href="form.html">
</HEAD>
```

Or, "table.html" and "form.html" may contain:

```
<HEAD> … other head information...
    <LINK rel="contents" href="HTML.html">
    <LINK rev="chapter" href="HTML.html">
</HEAD>
```

With respect to XML documents, by taking advantage of the XLink [XLink] mechanism, the containment relations between the documents can be expressed in

special link types that are encoded via the *role* attribute of linking elements. For instance, suppose in Figure 3.2 the XML document for "HTML User Guide" is "HTML.xml", the document for "How to use tables" is "table.xml" and the document for "How to uses forms" is "form.xml". The following out-of-link extended xlink can be contained in "HTML.xml" to describe the simple composite shown in the figure:

```
<content xml:link="extended" inline="false">
 <locator href="table.xml" role="contains">
 <locator href="form.xml" role="contains">
</content>
```

Finally, RDF [RDF Spec] provides a more systematic way to describe composites, whose components may be either HTML documents or XML documents or any other kinds of Web resources. The containment relations between the documents can be expressed in RDF descriptions via the link types such as "has Part" or "is Part of". The following sample RDF encoding shows how the simple composite displayed in Figure 3.2 can be described with Dublin Core [Dublin Core] vocabularies (suppose the documents are in HTML format):

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dc ="http://purl.org/dc/elements/1.1/"
      xmlns:dcterms="http://purl.org/dc/terms/">
  <rdf:Description about="HTML.html">
    <dc:title>HTML</dc:title>
      <dcterms:hasPart rdf: resource="table.html"/>
      <dcterms:hasPart rdf:resource="form.html"/>
  </rdf:Description>
</rdf:RDF>
```

3.2.2.6 Representation of Domain Models

Hierarchical structured domain models can be represented with either DTDs or RDF schemas. There are many such DTDs on the Web, for instance, the

Mathematical Markup Language (MathML) can be found at (http://www.w3.org/Math/).

Link-based (or relationship based) domain models can be represented with RDF schemas, which are defined with RDF Schema language [RDF Schema]. In fact many RDF schemas already exist on the Web. Especially, the SchemaWeb site (*http://www.schemaweb.info/*) provides a comprehensive directory of RDF schemas to be browsed and searched by human agents and also an extensive set of web services to be used by RDF agents and reasoning software applications that wish to obtain real-time schema information whilst processing RDF data

With RDF Schema language, the concepts in a domain model can be described with *rdfs:Class*. The subset/superset relationship between the concepts can be represented through the *rdfs:subClassOf* property. Other kinds of relationships can be represented in the *rdf:Property* class via the *rdfs:range* and the *rdfs:domain* property. The value of a *range* property is an RDF class describing a concept that is the end node of a relationship in the model. The value of a *domain* property is an RDF class describing a concept that is the start node of a relationship in the model.

This means, for representing domain models with only hierarchical structure such as libraries' classifications, *rdfs:Class* and *rdfs:subClassOf* are sufficient. For describing link-based, i.e. non-hierarchical domain models, in addition to *rdfs:Class* and *rdfs:subClassOf*, the *rdf:Property* class with its properties needs to be used.

For example, the following schema describes such a simple link-based domain model: "A book is written by an author, a book has comments":

```
<rdf:RDF xml:lang="en"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

    <rdfs:Class rdf:ID="book">
    <rdfs:subClassOf
rdf:resource="http://www.classtypes.org/useful_classes#Resource"/>
        </rdfs:Class>
```

```
<rdfs:Class rdf:ID="author">
<rdfs:subClassOf  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdfs:Class>

<rdfs:Class rdf:ID="comment">
<rdfs:subClassOf  rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdfs:Class>

<rdf:Property ID="writtenBy">
<rdfs:domain rdf:resource="#book"/>
<rdfs:range rdf:resource="#author"/>
</rdf:Property>

<rdf:Property ID="hasComment">
<rdfs:domain rdf:resource="#book"/>
<rdfs:range rdf:resource="#comment"/>
</rdf:Property>
</rdf:RDF>
```

3.2.2.7 Representation of Document Models

A document model specifies a document type, which can be represented in a DTD or an XML schema.

A DTD contains a formal definition of a particular type of document. It specifies what names can be used for element types, where they may occur, and how they all fit together. It also specifies attributes that can be associated with a given element type, as well as the data type and default value (if any) of each attribute.

An XML schema is a mechanism analogous to a DTD but more powerful than a DTD [XML Schemas: Structures] [XML Schemas: Datatypes]. This refers mostly to its ability of integration with namespaces, definition of constraints on the content of

an element type, explicit descriptions of relations between element types, and integration of structural schemas with primitive data types.

The following is a simple DTD that describes a type of documents that consist of a title and a content part, which further consists of zero or more paragraphs:

```
<!ELEMENT doc (title, contents)>
  <!ATTLIST doc version CDATA #IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT contents (paragraph*)>
<!ELEMENT paragraph (#PCDATA)>
```

A schema corresponding to this DTD can be:

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="doc">
      <complexType>
        <sequence>
          <element name="title" type="string"/>
          <element name="contents">
            <complexType>
              <sequence>
                <element name="paragraph" type="string"
minOccurs="0" maxOccurs="unbounded"/>
              </sequence>
            </complexType>
          </element>
        </sequence>
        <attribute name="version" type="string"/>
      </complexType>
    </element>
</schema>
```

This example shows us that the elements in a DTD or schema are organized in a tree. Each element is a node of the tree. The name of the element identifies the type of the node. In this sense, we can say that a DTD or schema describes a set of node types that can be used in a class of XML documents. Non-hierarchical, i.e., link-based document models can be defined using RDF schemas as link-based domain models (see Subsection 3.2.2.6).

3.3 Summary

This chapter has answered the question of what structural and semantic information can be found in hyperstructures and how it can be represented with the new Web standards. It provides the preliminary information for answering the central research question of this thesis, that is, how to efficiently apply this information for searching the Web and filtering and retrieving relevant information.

The structural and semantic information in hyperstructures has been classified into three different categories. The first category is basic hypermedia components and their types. The second category is high-level hypermedia structures formed by these basic hypermedia components. Two such major structures are hypertext composites and hypertext contexts. The third category contains various document models and domain models, which are two kinds of hyperstructures that have domain specific semantics. From the structural point of view, they can be either hypertext composites or hypertext contexts.

In standard representations of the above information, the encoding of the basic hypermedia components and their types in new Web standards is fundamental to the encoding of the high-level hypermedia structures and various document / domain models. With either HTML or XML or RDF and their friends, the encoding of the high-level structures is done by the use of some special kinds of elements as well as their attributes, e.g. XML extended links or RDF containers. In other words, high-level hypermedia structures can be computed based on the encoding of the basic hypermedia components and their types. Such computation can also be done for deriving document models and domain models, as these models are two kinds of

hyperstructures that exhibit domain semantics. However, these models can best be represented explicitly with RDF schemas and XML DTDs / schemas.

The next chapter of this thesis will present several search methods that explore the value of the structural and semantic information present in hyperstructures.

4 Hyperstructure-based Search Methods

The main effort of this thesis is to develop search methods that exploit the structural and semantic information in hyperstructures. This chapter presents four methods to show how the information can be applied in hyperstructure-based search. The first method applies link types in page ranking and filtering. The second makes use of hypertext contexts to draw a boundary for search efficiency. The third uses link-based composite structures in query and search. The fourth uses link-based domain models in search. The implementation issues for these methods are described in Chapter 5.

4.1 Method 1: Using Link Types in Page Ranking and Filtering

This section presents a general ranking and filtering mechanism that makes use of not only the quantity of links but also the link types that can be explicitly represented on the Web. The system interface designed for enabling the users to edit and specify various profiles for ranking/filtering tasks (described in Section 5.3) makes the mechanism to be easily applied to various domains on the Web.

4.1.1 Using Link Types in Ranking

There are some systems that make use of links in their ranking mechanisms. Most of them, however, only count the number of links associated with a search hit [Carriere&Kazman 1997] [Kleinberg 1998] [Brin&Page 1998].

In this thesis, we agree with Brin et al. [Brin&Page 1998] that page ranking can be propagational through links, i.e. the rank of a page is given by the rank of those pages which link to it, and their rank again is given by the rank of pages which link to them. Furthermore, we take into account not only the number of links associated with search hits but also the types of the links. In other words, we assume that the rank of a page depends on the sum of the ranks of the pages linking to it and the types of those links.

4.1.1.1 A Simplified Consideration

Intuitively, a simplified ranking mechanism that uses link types can be described like this: Links that are not relevant to the current search are filtered out and not counted in the ranking formulas.

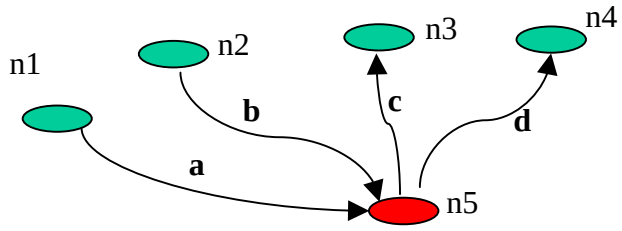


Figure 4.1 Typed links

For example, in Figure 4.1, the node at the lowest position (n5) has 2 incoming links, one is of type *a*, and the other is of type *b*. If link types are not considered, the number of incoming links that will be counted in the ranking is 2, while if link types are considered, say, to select only *a* type links, the number of incoming links counted in the ranking is 1.

It is clear that with respect to some users' specific interests, by taking into account the link types may improve the ranking of search hits.

4.1.1.2 A More Formal Consideration

Considering more carefully, the meaning of link types in ranking is not simply as "take it" or "ignore it". It seems necessary to introduce a factor called *ranking propagational rate* (RPR) into the mechanism. This factor will blur the absolute distinction between different link types, and may help to produce more reasonable ranking results with respect to users' specific interests.

Formally, the page ranking mechanism using link types is described as:

Let u be a page, $V = \{v_1, v_2, ..., v_k\}$ be a set of pages linking to u, t_i be a type of links, $f(t_i,v_i,u)$ be a value describing the *ranking propagational rate* from v_i to u, then the rank of u $R(u)$ can be defined as:

$$R(u) = \sum_{v_i \in V} f(t_i,v_i,u) * R(v_i)$$

$f(t_i,v_i,u)$ reflects a kind of view or interest on the page u from the page v_i (in which this link type is very important or unimportant). It also depends on a user's specific set of interests.

For instance, in Figure 4.1, the rank of the node *n5 is*:

$$R(n5) = f(a,n1,n5) * R(n1) + f(b,n2,n5) * R(n2)$$

This indicates, the rank of the node *n5* will be decided by the number of the links associated to it (*n1* and *n2)* as well as the types of those links (*a* and *b*).

4.1.1.3 Ranking Propagational Rate

Where can ranking propagational rates for a link come from? This is a critical issue to be addressed in order to make the mechanism really work.

There should be a few factors that may affect the value of a ranking propagational rate for a link. In our work up to now, we take into account the following information relevant to links:

- the type of the link
- the type of the source node of the link
- the type of the destination node of the link
- the content similarity of the two nodes of the link

Stated in a formal way, an RPR specification rule α taking into account the above factors is defined as:

$$\alpha = < RPR\_exp_\alpha , T\text{-}links_\alpha, T\text{-}sns_\alpha, T\text{-}dns_\alpha, Sims_\alpha >$$

where

$T\text{-}links_\alpha = \{t\text{-}link_i\}$ is a set of types of links

$T\text{-}sns_\alpha = \{t\text{-}sn_j\}$ is a set of types of the source nodes of links

$T\text{-}dns_\alpha = \{t\text{-}dn_k\}$ is a set of types of the destination nodes of links

$Sims_\alpha = \{sim_l\}$ is a set of values that represents the content similarity between source nodes and destination nodes of links

RPR_exp_α is an expression used to compute the RPR values corresponding to the combined condition of any $t\text{-}link_i$, $t\text{-}sn_j$, $t\text{-}dn_k$, sim_l.

Note not all the factors must be reflected in each RPR specification rule. That is, either $T\text{-}sns_a$ or $T\text{-}dns_a$, or $Sims_a$ can be left null in a rule.

For a link $\lambda = <t_\lambda, sn_\lambda, dn_\lambda>$, suppose $TypeofNode\ (sn_\lambda)$ denotes the type of the node sn_λ, $TypeofNode\ (dn_\lambda)$ denotes the type of the node dn_λ, $Similarity\ (sn_\lambda, dn_\lambda)$ denotes the content similarity value between sn_λ and dn_λ,

IF $(t_\lambda \in T\text{-}links_\alpha)$ *AND*

$(TypeofNode\ (sn_\lambda) \in T\text{-}sns_\alpha)$ *AND*

$(TypeofNode\ (dn_\lambda) \in T\text{-}dns_\alpha)$ *AND*

$(Similarity\ (sn_\lambda, dn_\lambda) \in Sims_\alpha)$

THEN its ranking propagational rate

$$f(t_\lambda, sn_\lambda, dn_\lambda) = RPR\_exp_\alpha$$

A ranking profile specified for performing a ranking task may consist of a set of RPR specification rules. Such a profile can be organized in a table as:

Types of Links	Types of Source Nodes	Types of Destination Nodes	Similarity of Source Nodes and Destination Nodes	RPR Expression
$T\text{-}links_\alpha$	$T\text{-}sns_\alpha$	$T\text{-}dns_\alpha$	$Sims_\alpha$	RPR_exp_α

Each row in the table represents the expression for computing a ranking propagational rate (the last column) and its conditions (the other columns). The RPR expression can be a constant, or built based on some symbolic expressions, e.g., $1/n$, where n is supposed to be the number of the links of a certain type in the source node

of the links. During performing the ranking task on a hypertext document collection, a system just needs to go through the rows, get corresponding ranking propagational rate expressions and compute the rate for each link in the collection.

The profiles for performing ranking tasks in a certain domain can be specified by users before they ask for the ranking, or predefined and contained as parts of the domain model, which defines node types and link types used in the domain. In the latter case, the profiles transfer the view or interest of the authors or document providers about the nodes and links in their documents to the users, and such profiles should be helpful suggestions for users to get a high quality ranking.

4.1.1.4 Computing Page Ranks

To enable computing page ranks in a document collection, a set of pages to be used as the sources of ranking need to be specified and assigned with certain page ranks. These pages usually reflect users' specific information interests and personalized views on them and even their neighborhood pages. With the page ranks of these source pages and a certain ranking profile, the ranks for all the other pages in the collection can be computed fairly straightforward if the issues of scale are ignored. Section 5.3 later in this thesis presents how our prototype system HyperSM supports this kind of computing.

4.1.2 Using Link Types in Filtering

As for ranking, link types can also be used in page filtering to improve the efficiency of browsing and searching on the Web. A simple example to support this argument is that a browsing system may hide the links of certain types from a page so that the users cannot go to pages that are irrelevant to their interests. In other words, the system filters out those pages for users by taking advantage of link types.

A filtering mechanism making use of link types may be more efficient when it takes into account some other information relevant to links. Whether a node (page) will be filtered out (discarded) in a browsing or searching activity can be decided based on one or more kinds of the following information:

- the type of the node to be filtered out

- the information related to the incoming links of the node, including
  - the types of those links
  - the types of the ancestor nodes, .i.e., the nodes which are the source nodes of the links (the node is the destination node of the links)
  - the content similarity of the node and the ancestor nodes
- the information related to the outgoing links of the node, including
  - the types of those links
  - the types of the descendant nodes, i.e., the nodes which are the destination nodes of the links (the node is the source node of the links)
  - the content similarity of the node and the descendant nodes

Formally, a filtering rule considering all the above kinds of information can be denoted as:

toFilterOut (T-nodes, INC, OUT), **in which**

$INC = <INC\_Y, INC\_N>$

$INC\_Y = \{inc_1, inc_2, inc_3, \ldots, inc_{niy}\}$

$INC\_N = \{inc_{niy+1}, inc_{niy+2}, inc_{niy+3}, \ldots, inc_{ni}\}$

$inc_i = < T\text{-}incLink_i, T\text{-}incNode_i, Sim\text{-}inc_i>$ $(1 <= i <= ni)$

$OUT = <OUT\_Y, OUT\_N>$

$OUT\_Y = \{out_1, out_2, out_3, \ldots, out_{noy}\}$

$OUT\_N = \{out_{noy+1}, out_{noy+2}, out_{noy+3}, \ldots, out_{no}\}$

$out_j = < T\text{-}outLink_j, T\text{-}outNode_j, Sim\text{-}out_j>$ $(1 <= j <= no)$

where

T-nodes is a set of types of the nodes to be filtered out

INC_Y / *OUT_Y* denotes a set of prerequisites for filtering out the nodes (i.e. the nodes that satisfy *INC_Y* / *OUT_Y* will be filtered out)

INC_N / *OUT_N* denotes a set of prerequisites for not filtering out a part of the nodes (i.e. the nodes that satisfy *INC_N* / *OUT_N* will be kept in the result)

T-incLink$_i$ / *T-outLink*$_j$ is a type of incoming/outgoing links of the nodes

T-incNode$_i$ / *T-outNode*$_j$ is a type of ancestors/descendants of the nodes

Sim-inc$_i$ / *Sim-out*$_j$ is a value representing the content similarity between the nodes and their ancestors/descendents.

In a filtering process following this rule, given a node *u*, suppose

$B_u = \{bu_k\}$ $(1 <= k <= N_{bu})$ is a set of ancestor nodes of *u* (N_{bu} is the number of ancestors of *u*)

$F_u = \{fu_l\}$ $(1 <= l <= N_{fu})$ is a set of descendant nodes of *u* (N_{fu} is the number of descendants of *u*)

TypeofLink$(bu_k \rightarrow u)$ is the type of the link from bu_k to *u*

TypeofLink$(u \rightarrow fu_l)$ is the type of the link from *u* to fu_l

TypeofNode(bu_k) / *TypeofNode*(fu_l) is the type of the node bu_k / fu_l

Similarity(bu_k, u) / *Similarity*(u, fu_l) is the content similarity value between the node *u* and bu_k / fu_l

$INC_u = \{<TypeofLink(bu_k \rightarrow u), TypeofNode(bu_k), Similarity(bu_k, u)>\}$

$OUT_u = \{<TypeofLink(u \rightarrow fu_l), TypeofNode(fu_l), Similarity(u, fu_l)>\}$,

THEN *u* will be filtered out

IF

(TypeofNode (u) $\in$ *T-nodes)* ***AND***

$INC\_Y \subseteq INC_u$ ***AND***

$INC\_N \cap INC_u = \varnothing$ ***AND***

$OUT\_Y \subseteq OUT_u$ ***AND***

$OUT\_N \cap OUT_u = \varnothing$.

Same as in ranking, not all the factors considered must appear in each rule. Either *T-nodes* or *T-incNode*$_i$ / *T-outNode*$_j$ or *Sim-inc*$_i$ / *Sim-out*$_j$ or even the complete *INC* or *OUT* can be left null. When a factor is left null, the values corresponding to the factor will not be taken into account in the filtering process and thus are not necessary to be computed. An example filtering rule and its application are shown in Figure 4.2. In this rule, the similarity factor in *inc1* and *out1* is left null. This filtering rule states that: to filter out the "person" typed pages that

- have "hasAssistant" typed incoming links from the "division" typed pages, and
- have "isStudentof" typed outgoing links to the "university" typed pages, but
- have no "loves" typed outgoing links to the "book" typed pages, and the similarity value between the "book" typed pages and the "person" typed pages is 1.

As for the application shown in the figure, in a filtering process following the rule,

The rule:

toFilterOut({person}, INC, OUT)

INC=<INC_Y, >

INC_Y= {inc$_1$}

inc$_1$=<"hasAssistant", "division", >

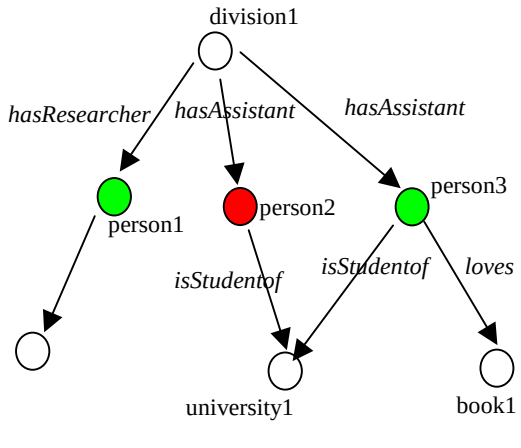
OUT=<OUT_Y, OUT_N>

OUT_Y= {out$_1$}

out$_1$=<"isStudentof", "university", >

OUT_N= {out$_2$}

out$_2$=<"loves", "book", 1 >



Similarity ("book1", "person1") = 1 *(both pages contain a set of keywords – a simplified case of content similarity)*

In a filtering process following the rule, the page *person2* will be filtered out while *person1* and *person3* will be kept in the result.

Figure 4.2 An example filtering rule and its application

the page *person2* will be filtered out while *person1* and *person 3* will be kept in the result.

In practice, a filtering profile used for performing a filtering task may need to contain a set of filtering rules. It can be represented in a table as:

Types of Nodes	**Incoming Links**	**Outgoing Links**
T-nodes	*(INC)*	*(OUT)*

Each row in the table represents a filtering rule. In fulfilling a filtering task with such a profile, all nodes that meet one or more rules in the profile will be filtered out.

Like ranking profiles, filtering profiles may also be specified by either document users or providers. In the latter case, the profiles may be provided as parts of certain domain models, and they are helpful suggestions from the document providers to the users for filtering.

How to implement the ranking and filtering mechanism in a search system is described in Section 5.3 in this thesis.

4.2 Method 2: Using Hypertext Contexts as Web Search Boundaries

The idea of drawing a boundary for the information space to be examined in search activities has been reflected in some popular search systems which provide category search, fielded search, "search within results", “find similar pages", and so on, such as Yahoo [Yahoo], InfoSeek [InfoSeek] and Lycos [Lycos]. It can also be seen in search agents that traverse the Web looking for specific information in real time [Sumner et al. 1998] or allow users to set up their own local searches [Miller&Bharat 1998].

This thesis proposes a general schema for searching in the Web space by making use of hypertext contexts. This schema covers the schemas proposed by other systems and meets the trend of Web development in its ability of expressing structures and semantics. It can be applied wherever hypertext context types are explicitly

represented in a collection or can be computed automatically by application systems based on users' hypertext context views. In the first scenario, representing hypertext contexts in a standard way (as described in Chapter 3 before) is the prerequisite for applying the contexts. In the second scenario, how to describe users' hypertext context views and the view construction process is essential before concrete instantiations of hypertext contexts based on the views can be constructed and further applied in search activities. For this reason, a model to describe such a kind of views and the view construction process is first introduced in the subsections below, and then the mechanism for searching in the Web space by making use of hypertext contexts is described.

4.2.1 Pure or Non-Pure Hypertext Context Nodes on the Web

The *hypertext contexts* that exist statically in hypertext document collections are usually described by means of specific node types. At least on the traditional Web,

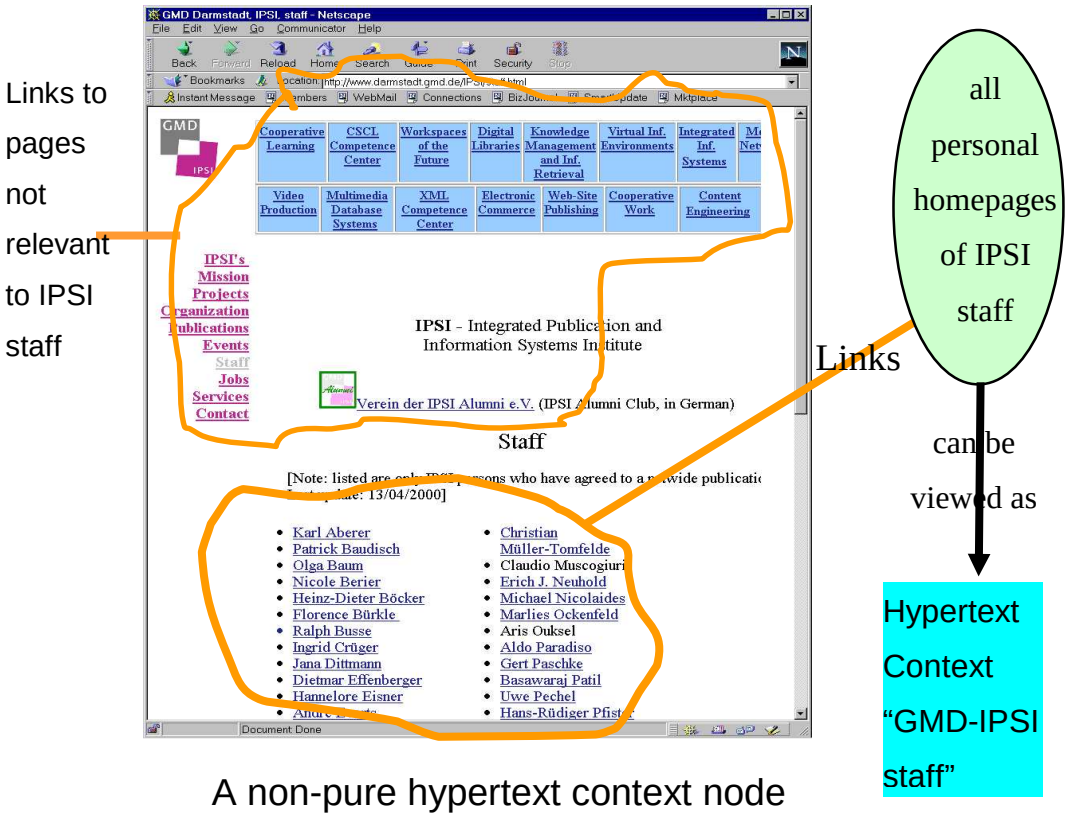


Figure 4.3 From hypertext context nodes to users' mental context views

there exist no hypertext context types that are explicitly represented and can be recognized automatically. However, many Web pages can be regarded as the nodes that "*contain*" hypertext context information, especially those pages with a number of outgoing links. Examples of those pages are various indexing pages, pages that contain site maps, tables of contents, guided tours, and focused WWW resource lists related to a particular topic or subject domain. However, they are usually not "pure" and cannot be recognized by computers.

A "pure" hypertext context node contains only outgoing links to pages that can be collected in a logical whole (a hypertext context) for a certain topic or subject domain. A "non-pure" hypertext context node contains outgoing links to pages that belong to a logical whole (a hypertext context), but also pages that are out of the context. On the current Web, it is clear that "non-pure" hypertext context nodes are more popular, as people may link any pages together for any reasons.

Figure 4.3 shows a "non-pure" hypertext context node which includes a complete list of links to all personal homepages of IPSI staff and also some other contents, such as links to "DELITE Homepage" and "DELITE homepages:Services". It is clear that the all personal homepages of IPSI staff can be collected in a logical whole (a hypertext context) for "GMD-IPSI staff", while the other pages should be kept out of this context.

4.2.2 Hypertext Context Views in Web Users' Minds

Corresponding to the theory about *personal information infrastructures* [Marchionini 1995], the contents of pure or non-pure hypertext context nodes reflect their authors' views on the related Web resources. Before people construct a specific pure or non-pure context node, they usually already have a mental context model and a corresponding mental view of their context application in their mind.

However, during browsing and searching, it is not usual that users intend to explicitly construct specific context nodes to record their mental context views. This is because it is in fact not easy to list all resource addresses (in the case of the Web, all urls) related to a particular topic or subject domain. Usually, users develop their mental context view about a topic or subject domain when they browse and search the

Web, and the view is often at the beginning somewhat vague and becomes more and more clear during the browsing and searching process. Especially, in most cases, users do not pay much attention to the urls of the related resources in the process. Users just recognize the context and may want to find something in the context.

Web users' mental context views about a topic or subject domain may come from a non-pure hypertext context node by recognizing a logical whole and other irrelevant parts (as shown in Figure 4.3, difficult for a machine, but easy for a human). Users may view all pages linking to a set of selected pages and all pages linked from the selected pages (regarded as seed pages for constructing the context) together as a context. Furthermore, users often find that some more or less isolated, i.e., "single" pages are included in their context view. This is to say that theoretically any Web pages are potential components of a context view in a Web users' mind.

4.2.3 A Model to Describe Users' Hypertext Context Views and the View Construction Process

4.2.3.1 A Simple Example

A simple example can be that one wants to know which people in GMD-IPSI (http://www.darmstadt.gmd.de/IPSI/) are doing work related to "data mining" and that one has access to the page "GMD Darmstadt IPSI, staff" (http://www.darmstadt.gmd.de/IPSI/staff.html) as shown in Figure 4.3. This page includes links to all personal homepages of IPSI staff, but also some other links (and is thus seen as a non-pure context node) as well. It would be great if one could tell the system to view all personal homepages of IPSI staff and their following pages in a context and the system could search only in that context. To do this easily, one hopes to select this page "GMD Darmstadt IPSI, staff" as a seed page, specify that all the pages linked from it are included in the context, while excluding the irrelevant pages from the context. Or, if a kind of "contain" link type has been used to annotate the links from this page to all personal homepages of IPSI staff, one can simply specify that the pages linked from the seed page with the type "contain" are included in the context. (Note: The seed page itself is not included in the context.)

In most cases one may want to or need to select more than one seed page for his/her intended context, and then include both the descendants and ancestors of the seed pages in the context, or include only one of them in the context. At the same time, one may also want to add single pages in the context. In one word, the operations needed by users depend on their choice of seed pages, the relationships between the seed pages and the components of their intended contexts, and their purposes as well.

4.2.3.2 The Model

To describe Web users' hypertext context views as mentioned in the above example, the following terms are used:

url - url of a page

url.ancestors(distance, relation) – all pages linking to the *url* page within a certain *distance* (specified by users) and the relationship between those pages and the *url* page is *relation* (specified by users, can be left null)

url.descendants(distance, relation) – all pages linked from the *url* page within a certain *distance* (specified by users) and the relationship between the *url* page and those pages is *relation* (specified by users, can be left null)

The following objects and methods are used to describe the interactive process in which users form their context views:

Context.id = Context.Create(label) – create a new empty context labeled with *label*

Context.id = Context.Create(label, url) – create a new empty context labeled with *label* and with *url* as the first seed page

Context.AddSeedPage(url) - add *url* as a new seed page to the context (the context may have several seed pages, which are themselves not included in the context as components)

Context.AddComponent(url) – add *url* to the context as a new component

Context.Include(url, neighbors, distance, relation) – use *url* as a seed page, add its descendants/ancestors *(neighbors="descendants/ancestors")* within distance *distance* and linked from or has link to *url* with the relationship *relation* to the context as new components

Context.Exclude(urls) – exclude the set of *urls* from the context

Context.Exclude(url, neighbors, distance, relation) – use *url* as a seed page, exclude its descendants/ancestors *(neighbors="descendants/ancestors")* within distance *distance* and linked from or has link to *url* with the relationship *relation* from the context

Given this, the above informal example could be described more formally in this way:

url1 = http://www.darmstadt.gmd.de/IPSI/staff.html

url2=http://www.darmstadt.gmd.de/IPSI/index.html

url3=http://www.darmstadt.gmd.de/IPSI/mission.html,

..., ... (some excluded pages)

exclude_urls = {url12, url3, ...}

The context view is:

url1.descendants(1) – exclude_urls

An example interactive context view construction process can, e.g., be described as:

1. *contextStaff.id = contextStaff.Create(“GMD-IPSI staff”);*
2. *contextStaff.AddSeedPage(url1);*
3. *contextStaff.Include(url1, "descendants", 1, “contains”);*
4. *contextStaff.Exclude(exclude_urls);*

Clearly, for more complex cases, step 2 to 4 can be iterated, and the operation *Context.AddComponent(url)* might also be included in the process.

If the link type information is available, the context view can be simply stated as *url1.descendants(1, "contains")*. No pages need to be further excluded from the resulting context.

4.2.4 Computing Instantiations of Hypertext Contexts from Users' Context Views

With clearly described users' context views, instantiations of hypertext contexts from users' context views can be computed automatically by systems and stored for later reuse. A standardized representation of these hypertext contexts (as described in Chapter 3 in this thesis) is required for sharing purposes.

As any Web pages may be linked to each other for any reasons and may be selected as seed pages or components in a context view, when computing a hypertext context from the view, all duplicated pages are removed. That is, the above mentioned functions *Context.AddComponent(url)* and *Context.Include(url, neighbors, distance, relation)* need to union rather than simply add the potential components into the context.

Thus, the components in the resulting hypertext context can be stated as: selected single pages *UNION* descendants/ancestors of selected seed pages and *NOT* excluded pages. The seed pages themselves are not included in the context unless they have been added explicitly in other ways, e.g., with *Context.AddComponent(url)*.

How our prototype systems help users to form their hypertext context views and then compute instantiations of hypertext contexts based on the views will be described in Section 5.4 in this thesis.

4.2.5 Limiting a Search in a Hypertext Context

No matter how a hypertext context is computed based on users' mental context views or directly encoded with the new Web standards, it is a logical container for a

set of nodes and links. It effectively cuts a boundary between its containing nodes and links from other nodes and links that are out of it (as shown in Figure 4.4), and thus our thinking about applying hypertext contexts as a mechanism to specify the information space to be examined in a search activity is straightforward. This mechanism should prove to be useful for improving the quality of the search and of the filtering process with respect to a specific topic or subject domain, since a large amount of non-relevant or undesirable pages are filtered out before any further search activities or pattern matching processes take place.

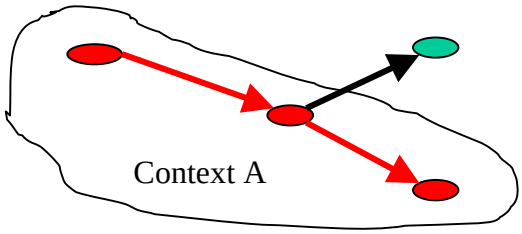


Figure 4.4 A simple hypertext context

For instance, in our experiments described later in this thesis, for the above simple example of searching for data mining experts in GMD-IPSI, if the search is limited in the hypertext context “GMD-IPSI staff”, one receives only 3 hits (at the time of our experiment). Comparatively, if the search is done in the whole collection, one will receive 375 hits and one has to browse through them to find out which are really relevant to one's information need. This example demonstrates that limiting a search concerning a specific topic or subject domain in a hypertext context is useful.

One can also mention other useful application scenarios. A scenario might be that users have been navigating to a page containing the keyword "information retrieval" and want to get all pages “surrounding” it and talking about "information retrieval" too. This helps users to define the content context of the page. In this case, one can simply define a hypertext context using the current page as a seed page and all pages surrounding it within a certain distance as components, and then submit a query with the keyword "information retrieval".

How our prototype system implements the schema is described in Section 5.4 later.

4.3 Method 3: Using Hypertext Composites in Structured Query and Search

The existence of compound documents [Eiron&McCurley 2003], or say logical information units [Tajima et al. 1999], or logical Web documents [Li-W et al. 2002] in the traditional Web space has been identified with devised algorithms. Such documents or information units contain implicit hypertext composite information. It is suggested that the structure of such documents or information units (which may consist of multiple pages) should be taken into consideration for information retrieval, as conjunctive queries with multiple keywords may fail to retrieve an appropriate document if those keywords appear in different pages within that document. That is, for such queries, an information unit as a set of connected nodes that reference the query terms should be provided to users as one atomic retrieval unit [Tajima et al. 1999; Li-W et al. 2002].

In this thesis, it is the possibility of explicitly representing hypertext composite information with the new Web standards that has motivated us to explore new search methods making use of the information. We propose that based on composite structures, hyperstructure-based query and search facilities can be implemented to enable users to query different levels of the structures with the same or different keywords (see the left-hand side part of Figure 4.5) and get search hits that are sets of inter-linked nodes (see the right-hand side parts of Figure 4.5) rather than separate

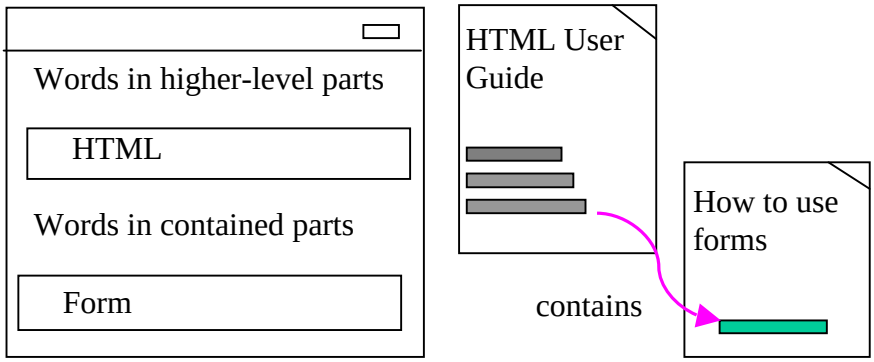


Figure 4.5 Structured query and search using link-based composite structure

nodes. Compared to the search results that are single nodes, the structured search results may be more precise and relevant to users' specific information needs. Furthermore, users will be provided with more contextual information about the nodes contained in the results.

4.3.1 Structured Queries Based on Hypertext Composites

Formally, the general structured queries based on hypertext composites can be described as follows:

Definition 1: *Structured Query Levels*

> *Structured query levels* STQLs is an integer value that indicates how many structural levels are to be contained in a structured query.

Definition 2: *Query Term*

> A *query term* QT_i is a content descriptor used to search for nodes indexed in a document collection. It may be a keyword or a phrase.
>
> QT = $\{QT_i\}$, where $1 \le i \le N_{qt}$, N_{qt} is the number of query terms

Example: QT = {"HTML user guides", "digital library", "information retrieval"}

Definition 3: *Qualifier*

> A *qualifier* Q_i is an additional restriction placed on the query terms that users input. It is used to fine-tune the query terms.
>
> $Q = \{Q_i\}$, where $1 \le i \le N_q$, N_q is the number of qualifiers

Example: Q={<exactly like>, <case sensitive>, <misspellings allowed>, <using stemming expressions>}

Definition 4: *Logical Operator*

> A *logical operator* L_i is used to combine logically two query terms or avoid (negation) a query term.
>
> $L = \{L_i\}$, where $1 \le i \le N_l$, N_l is the number of logical operators

Example: L={$\wedge$, $\vee$, $\neg$} where $\wedge$ means AND; $\vee$ means OR; $\neg$ means AND NOT.

Definition 5: *Level Query Expression*

> A *level query expression* $\overline{LQ_l}$ is a content descriptor used to search for nodes in the structural level *l*. It is constructed by one or more query terms QT (may with certain qualifiers Q) combined by logical operators L.
>
> $\overline{LQ_l} = ([(Q_1)]\ QT_1)\ \ L_1\ ([(Q_2)]\ QT_2) \ldots L_{n-1}\ ([(Q_n)]\ QT_n])$,
>
> where $n \ge 1$; $QT_i \in$ QT, $1 \le i \le n$; $Q_j \in$ Q, $1 \le j \le n$; $L_K \in$ L, $1 \le k \le n-1$

Example: $\overline{LQ_1}$ = ((<case sensitive>)"HTML") $\wedge$ ("user guide")

Definition 6: *Structured Query Expression*

A *structured query expression* $\overline{SQ}$ is a conjunction of some level query expressions.

$$\overline{SQ} = \wedge (\overline{LQ_1}, \overline{LQ_2}, \overline{LQ_3}, \ldots , \overline{LQ_n}),$$

where n = STQLs, i.e. the structured query levels, "$\wedge$" is logical operator AND

Example: $\overline{SQ} = \wedge$("User guide", "HTML", "form"), which describes a 3-level structured query with the query terms "User guide" in the first level, "HTML" in the second level, and "form" in the third level.

4.3.2 Structured Search Results Based on Hypertext Composites

As mentioned, the structured search hits resulted from the structured queries are not separate nodes but sets of inter-linked nodes. That is, in each search hit, links representing containment relations exist between the nodes.

Formally, the search results for a structured query $\overline{SQ}$ can be described with the following definitions:

Definition 7: *Level Query Result*

The *level query result* LQR for a level query expression $\overline{LQ_l}$ is a set of nodes $N = \{N_j\}$ that satisfy the query criteria specified in $\overline{LQ_l}$. Suppose $lqr(x, \overline{LQ_l}) = 1$ denotes that a node x satisfies $\overline{LQ_l}$ and $lqr(x, \overline{LQ_l}) = 0$ if x does not satisfy $\overline{LQ_l}$, then

$$LQR\ (\overline{LQ_l}) = \{N_i\}$$

where for every $N_j \in N$, $lqr(N_j, \overline{LQ_l}) = 1$, $1 \le j \le n$, n is the number of the result nodes.

If no nodes satisfy $\overline{LQ_l}$, $LQR\ (\overline{LQ_l}) = \varnothing$.

Definition 8: *Inter-containment-linked node chain*

An *inter-containment-linked node chain* $ICLNC_i$ is a series of nodes, each of which (except the first one) is linked from the node before it with a type that represents a containment relation and (except the last one) also links to the node after it with a type that represents a containment relation.

$$ICLNC=\{ICLNC_i\}$$

$$ICLNC_j = \{N_1(\text{contains})\text{->}N_2(\text{contains})\text{->}\ \ldots\ \text{->}N_n\},$$

where n = the number of the nodes in the chain

Example: $ICLNC_1$ = {“HTML User Guide” (contains) -> “How to user tables” (contains) -> ”How to specify the background color of tables”}, as shown in Figure 3.2.

Definition 9: *Structured Search Result*

The *structured search result* SSR for a structured query expression ($\overline{SQ}$) is a set of inter-containment-linked node chains, which are derived by computing the containment links between the nodes contained in the level query results corresponding to the level query expressions in $\overline{SQ}$.

$$SSR(\overline{SQ}) = SSR(\curlywedge(\overline{LQ}_1, \overline{LQ}_2, \overline{LQ}_3, \ldots, \overline{LQ}_n)) = \{ICLNC_n\}$$

$$ICLNC_j = \{N_1(\text{contains})\text{->}N_2(\text{contains})\text{->} \ \ldots \text{->}N_n\},$$

where $N_l \in \text{LQR}(\overline{LQ}_l)$, n = STQLs, i.e. the required structured query levels

Example: $SSR(\overline{SQ}) = \{ICLNC_1, ICLNC_2, ICLNC_3, ICLNC_4\}$, which describes a structured search result that consists of four inter-containment-linked node chains as shown in Figure 5.19 later in this thesis. In this result,

$ICLNC_1$ = {"The GNU C Library - Table of Contents" (contains) -> "The GNU C Library – Memory Allocation" (contains) -> "The GNU C Library – Memory Concepts"}

In Section 5.5 we will see how our prototype system implements the structured query and search based on hypertext composites.

4.3.3 Some Notes

The structured query and search model proposed in this section is for making use of link-based hypertext composites, while the XML Query [XML Query] with the support of XPath [XML Path] is to provide flexible query facilities to extract data from XML documents, in which the composite nodes are composed by non-linking mechanisms. It is claimed in [XML Query Requirements] that XML Queries must be able to traverse intra- and inter-document references. However, in XML Query Use

Cases [XML Query Use Cases] up to now, we see no use cases that show how the intra- and inter-document references will be used in XML queries.

With respect to the structured query construction parameters and the form of the structured search results, the structured search we propose in this section is different from the general structured search in the domain of database technology, which is the traditional domain that structured search tools fall into. A very good introduction to database technologies is [Chris 1995].

Finally, as hypertext composites can be seen as a special kind of hypertext contexts, the schema for using hypertext contexts as Web search boundaries as described in the last section (4.2) should also apply to hypertext composites.

4.4 Method 4: Using Link-Based Domain Models in Searching

Link-based domain models introduced in the subsection 3.1.3.1 provide a generalized level for describing the knowledge structure used in certain domains. They can be used to organize Web resources (hypermedia pages or fragments in the pages) and presented to users as a graphical navigation guide. They can also be applied in formulating structured queries so that users' information needs can be better satisfied than using no domain knowledge. This idea is presented in the following subsections with more details.

4.4.1 Web Resource Indexing with Domain Model Concepts

At first an indexing technique that takes advantage of domain model concepts in adaptive hypertext systems (AHS) [Brusilovsky 1996] has been adapted to the purpose of our work and a general method for indexing Web resources is defined.

Web resources, either pages or page fragments, are indexed with domain model concepts that are related to their contents. Moreover, as what the ELM-ART system [Schwarz et al. 1996] has done, during the indexing each concept is assigned a *role* in the resource index to signify the type of the relationship between the concept and the Web resource indexed. Still more, a *label* is set in each indexing result to distinguish

the Web resource indexed this time from the other resources indexed with the same concept.

Formally, this general method to index Web resources with domain model concepts can be described as:

Let

U_D signifies the set of Web resources in a given domain D,

L_D signifies the set of labels assigned to Web resources during indexing,

C_D signifies the set of domain model concepts in the domain D,

R_D signifies the types of the relationship between domain model concepts and Web resources in the domain D,

I_D signifies the set of the results of indexing Web resources with domain model concepts in the domain D,

then

An indexing result $\lambda \in I_D$ consists of the following components:

$$\lambda = <role_\lambda, concept_\lambda, label_\lambda, uri_\lambda>,$$

where

$uri_\lambda \in U_D$ signifies the Web resource being indexed,

$label_\lambda \in L_D$ signifies the label assigned to this Web resource in this indexing,

$concept_\lambda \in C_D$ signifies the domain model concept being used in this indexing,

$role_\lambda \in R_D$ signifies the type of the relationship between the $concept_\lambda$ and uri_λ, i.e. the role that $concept_\lambda$ plays to the content of uri_λ.

To give a simple example, suppose:

Page 1: *http://ebookshops/Arizon* is the homepage of an electronic bookstore site “Arizon”.

Page 2: *http://authors/smith* is the homepage of “Smith”, who is an author recorded in the bookstore site.

Page 3: *http://books/math* is the page describing the book "Math", which is sold by the bookstore site and written by the author "Smith".

With the simple domain model as shown in Figure 4.6, the site can produce the following indexing results:

For Page 1: *<"category", "eBookshop", "Arizon", "http://ebookshops/Arizon">*

For Page 2: *<"category", "Author", "Smith", "http://authors/smith">*

For Page 3: *<"category", "Book", "Math", "http://books/math">*

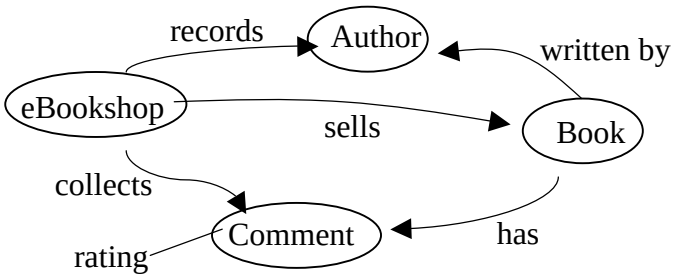


Figure 4.6 A simple domain model

That is, for example, the site indexes the page "*http://ebookshops/Arizon*" with the concept "eBookshop", and sets the relationship between the concept and the page as "category" and the label of the page as "Arizon".

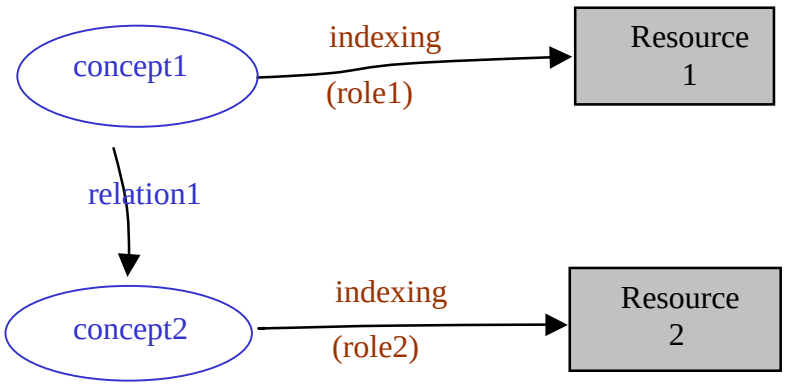


Figure 4.7 The essence of the indexing process with domain model concepts

It is clear that the indexing process is in fact a process of annotating a set of semantic links (relationships) between domain model concepts and Web resources. It is also a process to make the link-based domain model a superstructure of the Web resources indexed (as shown in Figure 4.7). This process is applicable to all level domain models (as described in the subsection 3.1.3.1), but the possible search methods based on the indexing results may be different. Especially, in order to support some of the methods, additional efforts may be needed during or after the indexing process.

As for supporting the structured query model to be presented in the next subsection, annotating the relationships between the Web resources indexed is needed. As described in the subsection 5.6.2 later in this thesis, this annotation process can be concurrent with the indexing process. The result of such an annotation can be described formally as:

$$\Upsilon = <sourceUri_{\Upsilon}, conceptRel_{\Upsilon}, destUri_{\Upsilon}>,$$

where

$sourceUri_{\Upsilon} \in U_D$ signifies the Web resource as the start point in the relationship,

$destUri_{\Upsilon} \in U_D$ signifies the Web resource as the end point in the relationship,

$conceptRel_{\Upsilon} \in REL_D$ signifies the relationship between $sourceUri_{\Upsilon}$ and $destUri_{\Upsilon}$, here REL_D is the set of relationships between domain model concepts in the domain D.

U_D has been defined above, i.e.

U_D signifies the set of Web resources in a given domain D.

For instance, in the example we give above about the 3 pages in the bookstore site "Arizon", the relationships between these pages can be expressed as:

<"http://ebookshops/Arizon">, "sells", "http://books/math">

<"http://ebookshops/Arizon">, "records", "http://authors/Smith">

<"http://books/math">, "written by", "http://authors/Smith">

In the following we will present our general structured query model that takes advantages of link-based domain models.

4.4.2 Structured Queries with Link-Based Domain Models

As stated in the last subsection, when the concepts in a link-based domain model are used to index Web resources in the given domain, the link-based domain model in fact describes a superstructure upon the Web resources. This structure may help users to understand the content and context of the resources better, and may be applied in constructing structured queries so that users may get better and more specific search results that are relevant to some specific information needs.

For instance, in an electronic bookstore site, the comments on the books made by reviewers and readers may help potential buyers to make a decision on which book to buy. In other words, users may like to search for the books, e.g. written by 'Smith' and receiving a good rating in readers' comments. To meet such users' specific information needs, the simple link-based domain model as shown in Figure 4.6 can be used to organize the site. All the books (as well as their descriptions, prices and so on) are indexed under the 'Book' concept, all the authors are indexed under the "Author" concept, and all the comments (rating, etc.) collected are indexed under the "Comment" concept. The search system for the site then enables users to formulate structured queries such as "Book=? AND Author='Smith' AND Comment.rating =5" based on the relationship information in the domain model, and afterwards implements the searches. The search results can surely help users to decide which new books to buy.

Formally, the general model for structured queries with link-based domain models is described as follows:

Definition 1: *Query Term*

> A *query term* QT_i is the content descriptor used to search for Web resources indexed by domain model concepts. It may be a keyword, a phrase, a value, or a logical expression (i.e. keywords / phrases combined with *AND* / *OR* and parentheses).
>
> $QT = \{QT_i\}$, where $1 \le i \le N_{qt}$, N_{qt} is the number of query terms

Example: QT = {"Smith", "Digital Library", "5", "information retrieval *AND* hypertext"}

Definition 2: *Query Field*

> A *query field* QF_i corresponds to a concept or an attribute of a concept in the domain model. It limits the scope of a query term, i.e., it requires that the provided term be contained in the indexing result of Web resources indexed by the domain concept.
>
> $QF = \{QF_i\}$, where $1 \le i \le N_{qf}$, N_{qf} is the number of query fields

Example: For the domain model shown in Figure 4.6, QF = {<Author>, <Book>, <Comment.rating>, <eBookshop>)

Definition 3: *Role Operator*

> A *role operator* RO_i is used to describe the relationship between a domain model concept and a Web resource indexed with the concept.
>
> $RO = \{RO_i\}$, where $1 \le i \le N_{ro}$, N_{ro} is the number of role operators

Example: RO = {<content>, <prerequisite>, <category>}

Definition 4: *Qualifier (same as **Definition 3** in Section 4.3)*

> A *qualifier* Q_i is an additional restriction placed on the query terms that users input. It is used to fine-tune the query terms.
>
> $Q = \{Q_i\}$, where $1 \le i \le N_q$, N_q is the number of qualifiers

Example: Q = {<exactly like>, <case sensitive>, <misspellings allowed>, <using stemming expressions>}

Definition 5: *Field Query Expression*

> A *field query expression* $\overline{FQ_i}$ is constructed by a query field QF_j with a query term QT_k combined by "=". QF_j may have a role constraint RO_m, while QT_k may have a qualifier Q_l. That is:
>
> $\overline{FQ_i} = (QF_j\ [(RO_m)] = [(Q_l)]\ QT_k)$, where $QF_j \in$ QF,
>
> $RO_m \in$ RO, $QT_k \in$ QT, $Q_l \in$ Q

Example: $\overline{FQ_1}$ = (<Author> (<category>) = (<exactly like>) "Smith")

Field ($\overline{FQ_i}$) is used to denote the query field used in the query expression $\overline{FQ_i}$, i.e., *Field* ($\overline{FQ_i}$)= QF_i.

Definition 6: *General Structured Query Expression*

A *general structured query expression* $\overline{GSQ}$ is a conjunction of some field query expressions.

$$\overline{GSQ} = \wedge (\overline{FQ_1}, \overline{FQ_2}, \overline{FQ_3}, \ldots , \overline{FQ_n}),$$

where $n \leq N_{GSQ}$, i.e. the number of concepts and attributes in the domain model, "$\wedge$" is logical operator AND.

It is used for searching the Web resources indexed with the concepts represented by the query fields *Field* $(\overline{FQ_1})$, *Field* $(\overline{FQ_2})$, …, *Field* $(\overline{FQ_n})$, especially those resources that own the relationships defined in the domain model between each other.

Example: $\overline{GSQ}$ = $\wedge$(<Author> (<category>) = "Smith", <Book> (<category>) = "Digital Library", <Comment.rating> = "5"), which searches for authors "Smith", books "Digital Library", and comments giving the rating level 5. Especially, the Web resources about

- the authors "Smith" who *write* (the inverse of *written by*) the books which are named "Digital Library" and *have* comment rating 5,
- those books, and
- those comments

are the most interesting search hits for this query request.

Definition 7: Neighborhood Fields

> Given a query field QF_j representing a concept, the *neighborhood fields* of QF_j, denoted as $NF(QF_j)$, are a set of query fields which correspond to the neighborhood concepts of QF_j or the attributes of these concepts.

Example: For the domain model shown in Figure 4.6, NF (<Book>) = {<Author>, <Comment.rating>, <eBookshop>}

Definition 8: *Neighborhood-oriented Structured Query Expression*

> A *neighborhood–oriented structured query expression* $\overline{NOSQ}\ (QF_j)$ is a conjunction of some field query expressions of the neighborhood fields $NF(QF_j)$.
>
> $$\overline{NOSQ}\ (QF_j) = \wedge\ (\overline{FQ_1}, \overline{FQ_2}, \overline{FQ_3}, \ldots, \overline{FQ_n}),$$
> where $QF_j \in QF$, $Field\ (\overline{FQ_i}) \in NF(QF_j)$
>
> It is used for searching the Web resources indexed with the concept represented by QF_j via the relationships between these resources and the resources indexed with the concepts represented by $NF(QF_j)$.

Example: $\overline{NOSQ}$ (<Book>) = $\wedge$ (<Author> (<category>) = ”Smith”, <Comment.rating> = ”5”), which searches for books written by authors “Smith” and rated as 5.

It is apparent that a neighborhood-oriented structured query can only be formulated dynamically, as it is based on a given field and every field has probably different neighborhood fields.

The implementation issues for the above structured query model making use of link-based domain models will be discussed in Section 5.6.

4.4.3 Discussion

As far as we know, the work presented in this thesis is the first one that focuses on exploring Web search methods that make use of both the structural and semantic information contained in link-based domain models. Though the SHOE search tool [Heflin&Hendler 2000; SHOE] provides a somewhat similar query model, it uses a specific SHOE language and just applies the concepts (called categories) in SHOE ontologies in categorizing documents. Comparatively, our model for indexing and structured queries with link-based domain models is much more general with respect to the use of Web standards in domain representations and the facility of enabling the specification of concept roles in indexing.

4.5 Summary

This chapter presents four methods that exploit the structural and semantic information in hyperstructures for searching the Web and filtering and retrieving relevant information. Being within the framework "Hyperstructure-Based Search Methods for the World Wide Web", these methods are actually related to each other.

For instance, the ranking and filtering mechanism can not only be used in ranking and/or filtering the search results derived by other hyperstructure-based search methods but can also help to construct hypertext contexts or composites based on node and link types. All the Web resources indexed with a same domain concept can be regarded as components of a hypertext context. In deriving search results for a structured query formulated based on a link-based domain model, the idea of using hypertext contexts as search boundaries has in fact been applied.

5 A Prototype System - HyperSM

This chapter presents a prototype system HyperSM designed to implement the hyperstructure-based search methods described in the last chapter.

5.1 Issues for Supporting the Hyperstructure-Based Search Methods in a System

In practice, there exist a few issues to be addressed to support the hyperstructure-based search methods in a system.

The first issue is to represent hyperstructural information in a standard way and make it sharable and reusable throughout the Web. This issue has been discussed in Section 3.2 and can be seen as the prerequisite for addressing other issues. With this prerequisite, a Web search system that intends to support the hyperstructure-based search methods proposed in this thesis should be able to

- gather from the Web available hyperstructural information, do necessary indexing and organize the information in the system efficiently,
- provide a user-friendly interface to enable users to construct queries that are based on hyperstructural information in a comfortable way,
- compute hyperstructure-based search results with acceptable system performance, and
- present sound search results in a way that makes it easy for users to understand and get more contextual information about the results.

In the following sections, we will see how our prototype system HyperSM addresses these issues. We first give a high-level discussion of its architecture and then introduce how it supports each of the search methods. In details, Section 5.3 presents how it supports Method 1. Section 5.4 presents how it supports Method 2. Section 5.5 presents how it supports Method 3. Finally, Section 5.6 presents how it supports Method 4.

5.2 System Architecture Overview

The prototype system HyperSM contains the components that represent different aspects of the system functions (as shown in Figure 5.1):

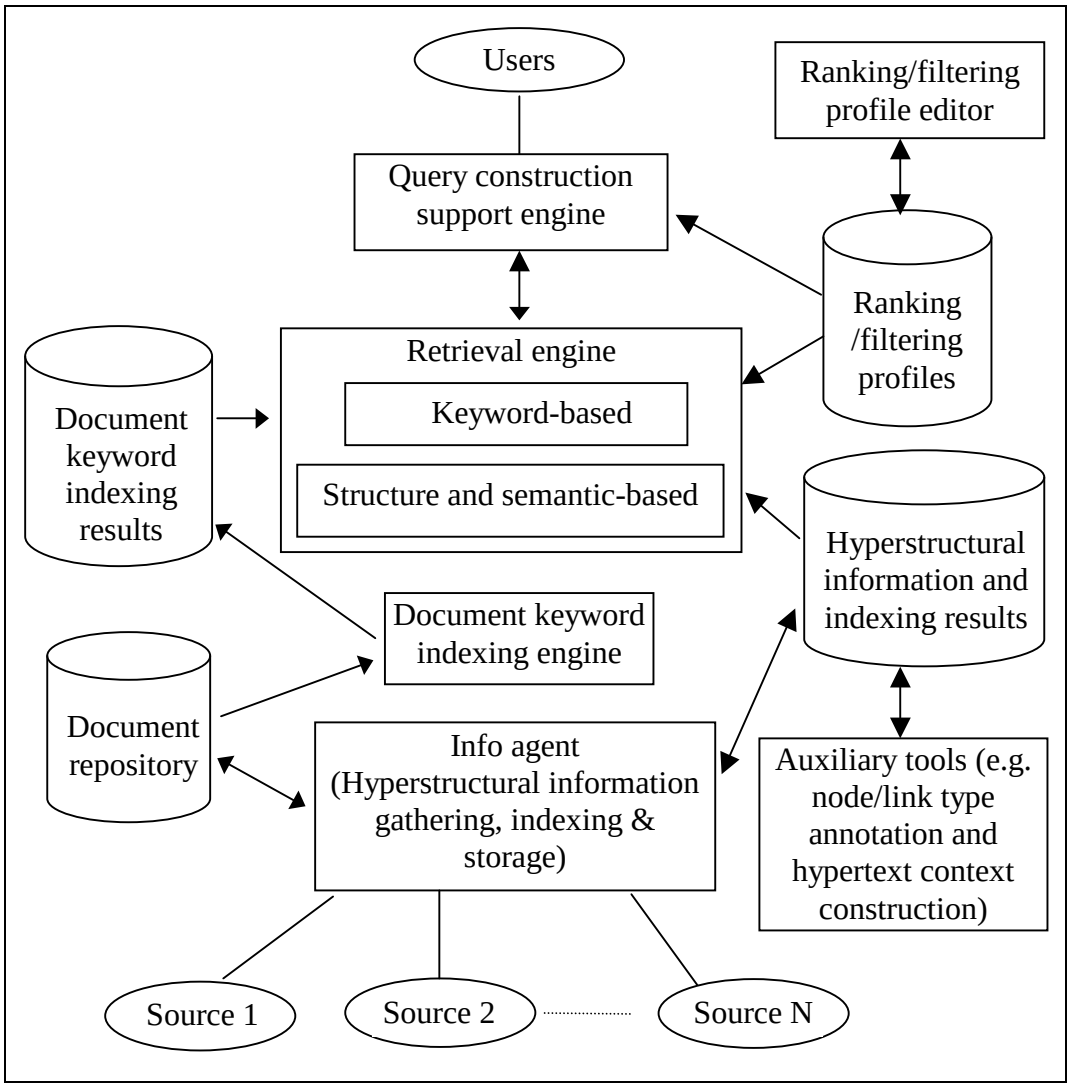


Figure 5.1 System architecture overview

The **info agent** is responsible for handling Web resources (HTML, XML and RDF docs). It performs in principle two functions. The first is to download the Web resources that own URIs specified by users or extracted from the resources during the parsing process and then store them in the document repository. The second is to extract hyperstructural information from the downloaded documents directly or

through indexing and then store the information in the database for hyperstructural information.

The **document keyword indexing engine** is to do keyword indexing of the Web resources (HTML and XML docs) gathered for content search. As keyword indexing is a mature technology, this thesis does not need to give much detail about it. As an example, the current version of HyperSM makes use of Glimpse [Glimpse] as its keyword indexing and search engine.

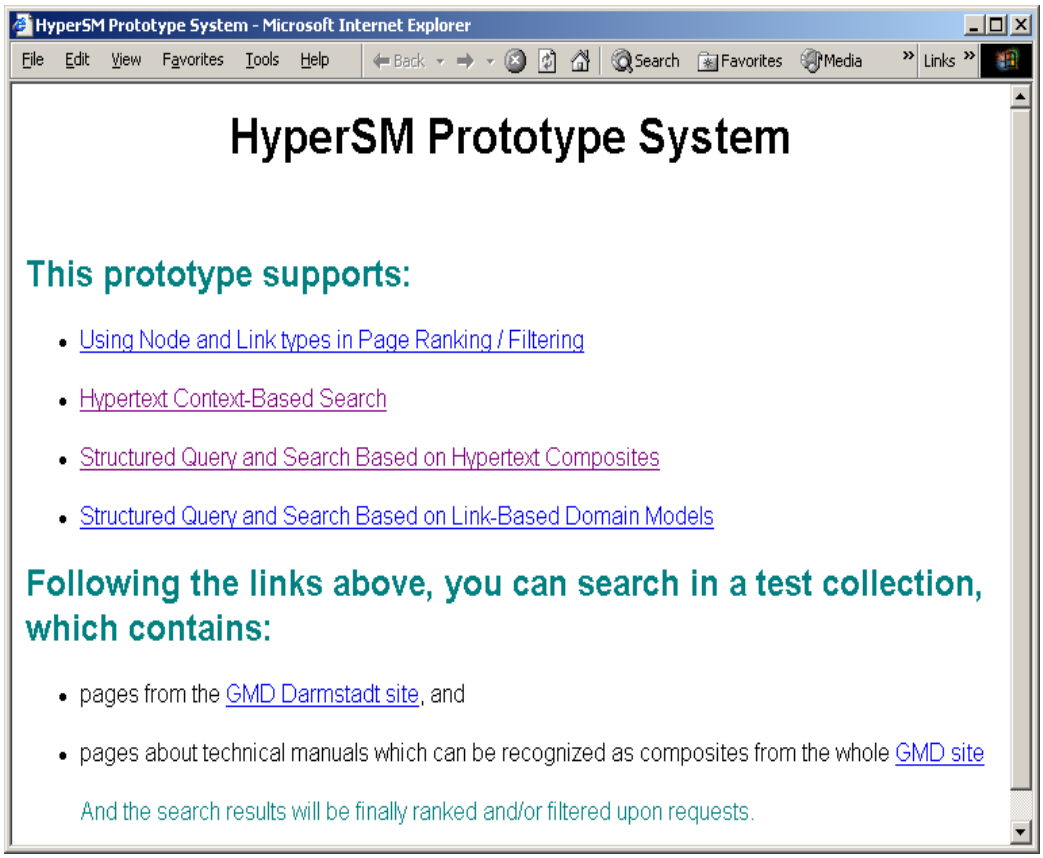


Figure 5.2 HyperSM prototype system

The **query construction support engine** is responsible for enabling users to construct queries applying hyperstructural information, transferring the queries to the **retrieval engine**, and presenting search results derived by the **retrieval engine** to users. This engine is implemented as Web browser clients with form-based user interfaces, each of which supports users to construct queries corresponding to one or several search methods proposed in this thesis. An exemplary integrated interface is as shown in Figure 5.2.

The **retrieval engine** is responsible for using data in the database to compute search results and sending search results to the **query construction support engine**. It includes **keyword-based search engine** (not further exploited here) and **structure and semantic-based search engine**, which is one of the most important engineering parts contributed by this thesis. In this combination, the **structure and semantic-based search engine** is responsible for receiving queries from the **query construction support engine** and decomposing the queries. It sends each query term in the queries to the **keyword-based search engine,** gets the distinct results, and then makes use of the structural and semantic information in hyperstructures as filters to compute the final results, which are then sent back to the **query construction support engine.**

The **ranking/filtering profile editor** is for enabling users to edit the ranking and filtering profiles that correspond to the ranking/filtering mechanism proposed in this thesis for applying link types. These profiles can be stored in the database of the system for later reuse.

The **auxiliary tools** in the system are used for enabling users to define hyperstructural information for browsing and searching the Web. These tools also help to apply the search methods proposed in this thesis to the traditional Web, in which little explicitly represented structural and semantic information exist. For instance, in the current version of the system, an interactive node/link type annotation and a hypertext context construction tool are provided.

The **data storage** in the system includes the **document repository** and databases for **hyperstructural information,** for (corresponding) **indexing results**, and for **ranking/filtering profiles**. Though not illustrated in the Figure 5.1, a **database management system** is the backbone of the entire system. It receives and sends data to the other components of the system. It can be an object relational DBMS, as for example, Informix Universal Server.

The details about how each component works in supporting the search methods presented in this thesis will be described in the remainder of this chapter. As it is not necessary for a system to support all introduced methods in parallel, this thesis discusses the implementation issues of each method separately. For this reason,

redundancies of some descriptions (especially about the database schema) cannot be avoided totally.

5.3 Supporting Method 1: Using Link Types in Page Ranking and Filtering

This section discusses several of the special issues to be addressed in implementing the ranking and filtering mechanisms presented in Section 4.1.

5.3.1 Link Information Gathering and Storing

In the system, the info agent is responsible for gathering and storing link information. The Web pages (nodes) that own URIs specified by users or extracted from the specified pages during the parsing process are downloaded and stored in the document repository. Every page gets an associated ID number called nodeID, which is assigned whenever a new URI is extracted from a page. All the link information represented with Web standards (as described in the subsection 3.2.2) in the collection is extracted and stored in the link base of the system. The database schema for the link base is shown in Figure 5.3.

The *node type table* stores all the node types used in the collection. The *node table* is used for storing the types and URIs of the nodes gathered in the collection. The *link type table* is used for storing the link types existing in the collection. The *link table* contains the links between the nodes in the collection.

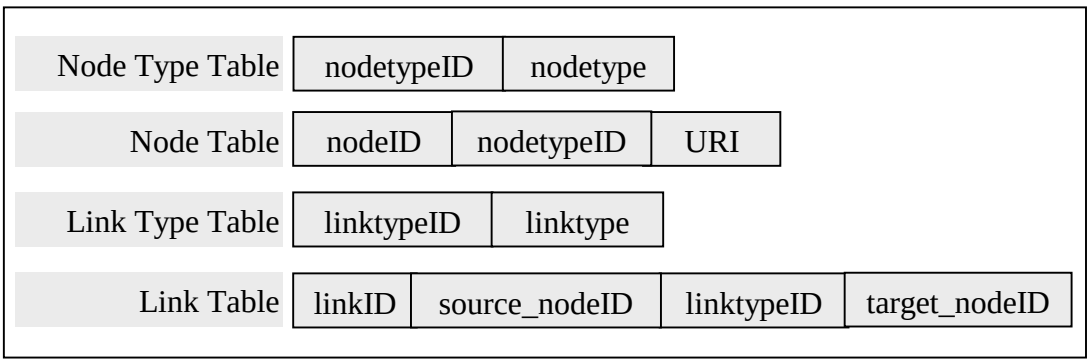


Figure 5.3 Database schema for link information – a link base

For instance, given page 1 "*http://www.darmstadt.gmd.de/~qiu*" is an "*author*" typed page, page 2 "*http://www.darmstadt.gmd.de/~qiu/publ1.html*" is a "*publication*" typed page, the link from page 1 to 2 is of the type "*writes*", then the following items may be stored in the database:

*in the **node type table***: *(nodetypeID | nodetype)*

1 | publication

2 | author

*in the **node table***: *(nodeID | nodetypeID | URI)*

1 | 2 | http://www.darmstadt.gmd.de/~qiu

2 | 1 | http://www.darmstadt.gmd.de/~qiu/publ1.html

*in the **link type table***: *(linktypeID | linktype)*

1 | writes

*in the **link table***: *(linkID | source_nodeID | linktypeID | target_nodeID)*

1 | 1 | 1 | 2

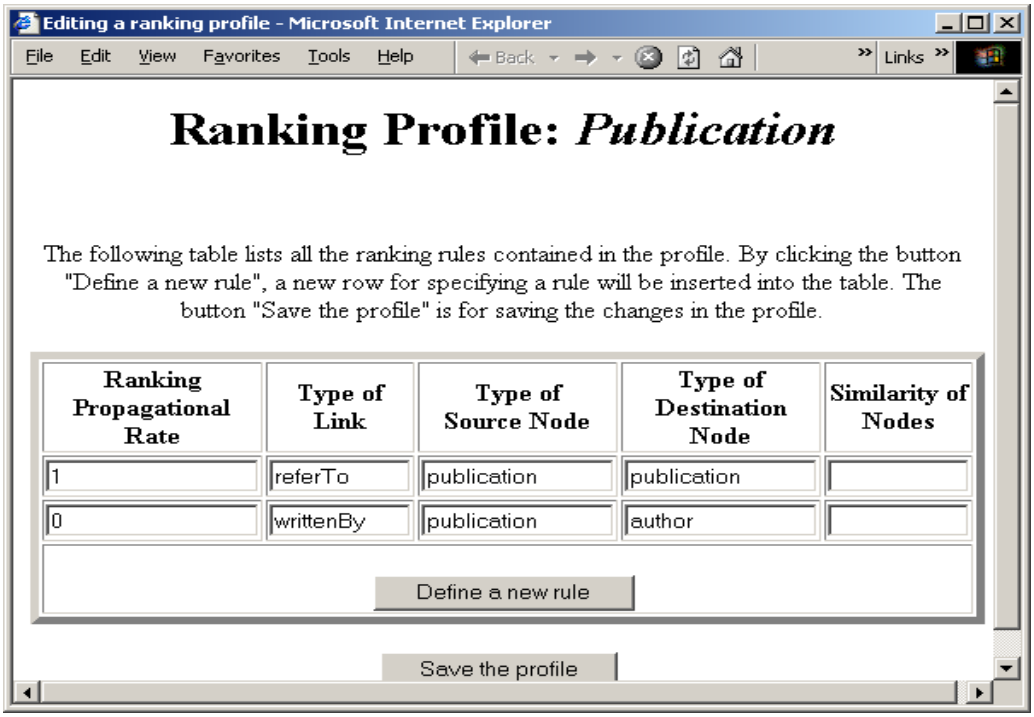


Figure 5.4 Editing ranking profile

5.3.2 Ranking and Filtering Profiles Editing and Storing

As described in Section 4.1, any ranking or filtering task will be performed by the system based on a profile specified by users or document providers. For editing such a profile, a user-friendly interface needs to be provided. Such an interface presents all the ranking or filtering rules contained in a profile and enables users to insert new rules into the profile. For instance, Figure 5.4 shows a ranking profile "Publication", which contains two ranking rules. The first rule states that the ranking propagational rate is 1 for a "referTo" typed link from a "publication" typed page to another "publication" typed page. The second rule states that the ranking propagational rate is 0 for a "writtenBy" typed link from a "publication" typed page to an "author" typed page. Figure 5.5 shows a filtering profile "Publication", which contains only 1 filtering rule. This rule states: to filter out (discard) "publication" typed pages that have no "writes" typed incoming links from "author" typed pages.

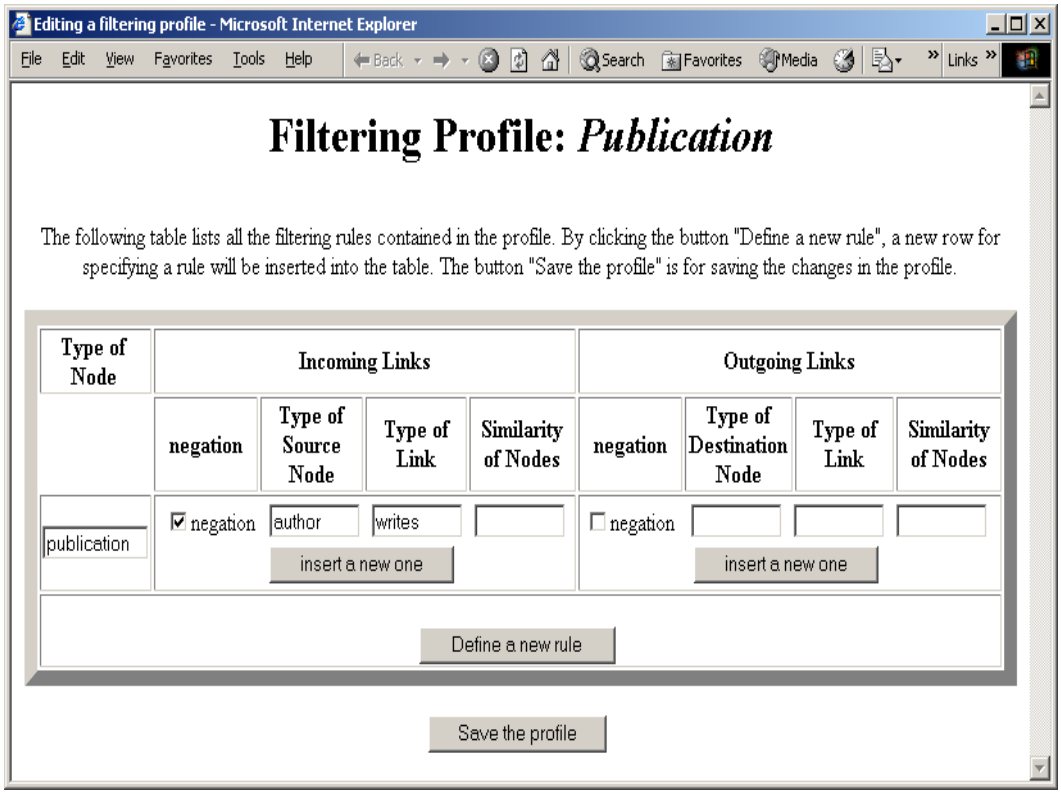


Figure 5.5 Editing filtering profile

To enable the reuse of the defined profiles in the system, the database schema as shown in Figure 5.6 has been designed for storing the profiles. The database tables in

the schema implement the tables for representing ranking or filtering profiles described in Section 4.1. Every profile and rule gets an id value (i.e. *R_profileID*, *R_ruleID*, *F_profileID*, *F_ruleID*), which is then referenced by other tables. In each table, all columns for describing node, node type, link and link type contain certain id values referring to the corresponding column values in the link base described in the subsection 5.3.1. The column "negation" is used to distinguish the elements in *INC_Y*/*OUT_Y* (the value is 0) and the elements in *INC_N*/*OUT_N* (the value is 1) as described in the subsection 4.1.2. Every time when a profile stored in the database is to be modified, the system gets the data of the profile from the database and constructs the interface for editing (as shown in Figure 5.4 and 5.5) dynamically.

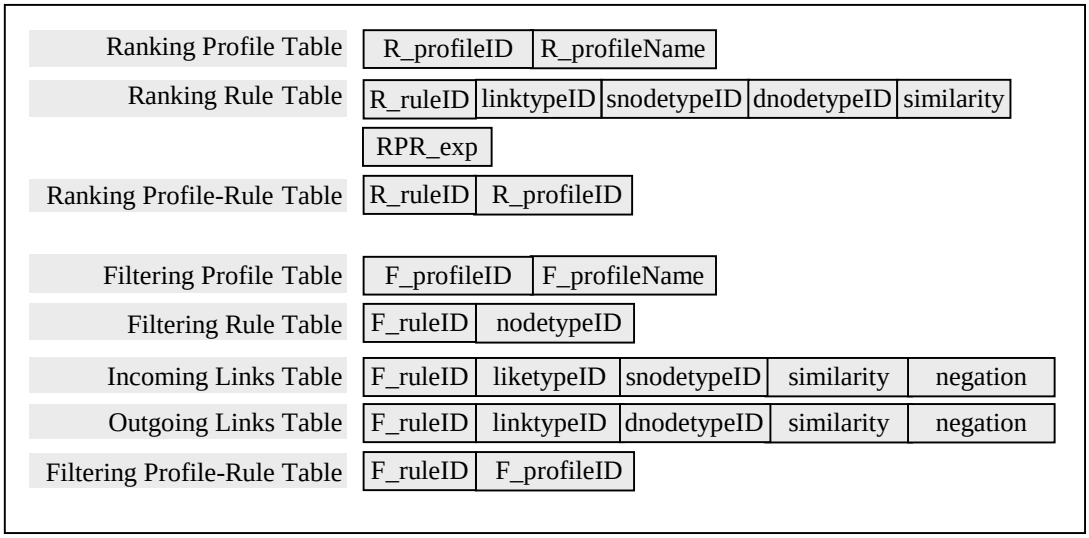


Figure 5.6 Database schema for ranking and filtering profiles

Suppose the node types and link types contained in the ranking / filtering profiles shown in Figure 5.4 and 5.5 are stored in the ***node type table*** and ***link type table*** shown in Figure 5.3 with the following items:

*in the **node type table***: *(nodetypeID | nodetype)*

1 | publication

2 | author

*in the **link type table***: *(linktypeID | linktype)*

1 | writes

2 | writtenBy

3 | referTo

The ranking profile shown in Figure 5.4 may then be stored in the database with the following items:

*in the **ranking profile table***: *(R_profileID | R_profileName)*

1 | Publication

*in the **ranking rule table***: *(R_ruleID | linktypeID | snodetypeID | dnodetypeID | similarity | RPR_exp)*

1 | 3 | 1 | 1 | null | “1”

2 | 2 | 1 | 2 | null | “0”

*in the **ranking profile-rule table***: *(R_ruleID | R_profileID)*

1 | 1

2 | 1

The filtering profile shown in Figure 5.5 may be stored in the database with the following items:

*in the **filtering profile table***: *(F_profileID | F_profileName)*

1 | Publication

*in the **filtering rule table***: *(F_ruleID | nodetypeID)*

1 | 1

*in the **incoming links table***: *(F_ruleID | linktypeID | snodetypeID | similarity | negation)*

1 | 1 | 2 | null | 1

*in the **filtering profile-rule table***: *(F_ruleID | F_profileID)*

1 | 1

5.3.3 Form-Based Interface for Specifying Ranking or Filtering Profiles in Queries

In the system, the interface for specifying a ranking or filtering profile can be as simple as a selection field contained in a query form (varies a lot with respect to

complexity in different systems). The options listed in the selection field are all the ranking or filtering profiles defined in the system.

As a simple example, Figure 5.7 displays a query page that contains only two fields for specifying query criteria other than the selection field for profile names. One of the fields is for specifying query keywords, while the other one is for specifying the search boundary, which can be either the full collection or a hypertext context predefined in the system. Figure 5.8 displays a result page that indicates which ranking or filtering profile has been used in deriving the search results. The result page also provides a link for going back to the query page.

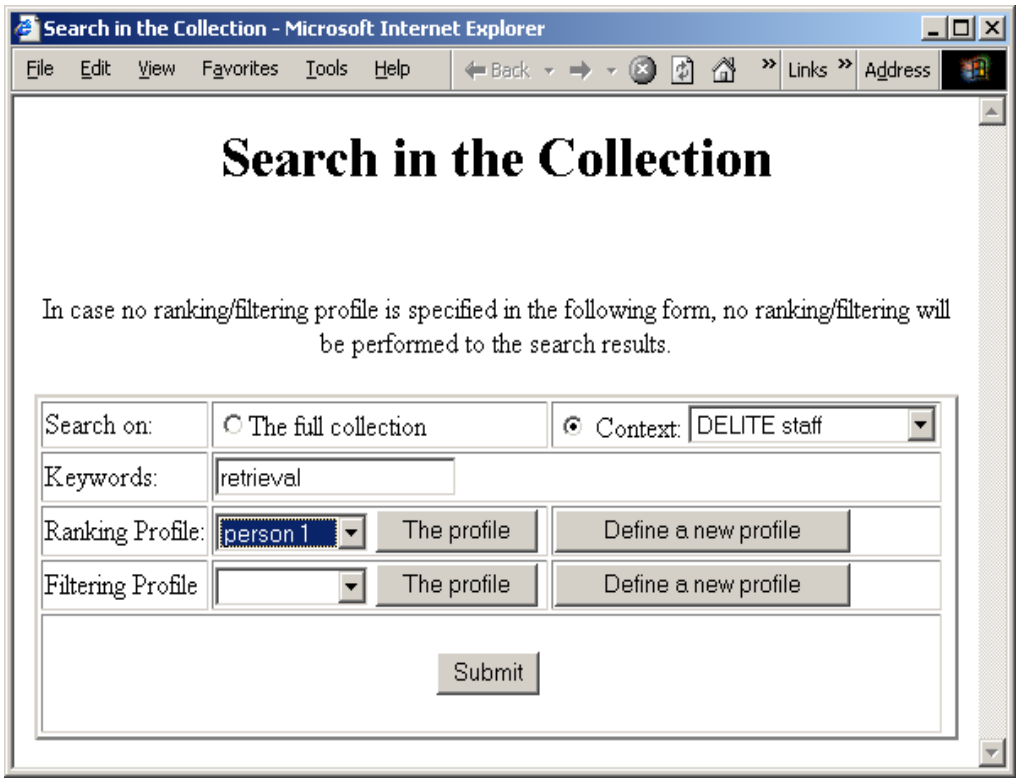


Figure 5.7 An example query page with specifying ranking / filtering profiles

5.3.4 Computing Ranked Search Results

To compute results for queries with a specified ranking and/or filtering profile, the **structure and semantic-based search engine** of the system needs to contain 2

components: a ranking component and a filtering component. Together with the **keyword-based search engine,** each component performs a part of the work for deriving search results, as its name indicates.

Concretely, the process of deriving the search results for a query can be divided into 3 steps:

1. The keyword-based search engine computes keyword-based search results corresponding to the input query terms. *(Let r1 denote the result from this step).*

2. If a certain filtering profile has been selected in the query, the filtering component performs filtering of the result from the first step (i.e. *r1*) according to the selected filtering profile. *(Let r2 denote the result from this step. If no filtering has been performed, r2=r1).*

3. If a certain ranking profile has been selected in the query, the ranking component performs ranking of the result from the second step.

After these 3 steps, the search results corresponding to a query have been derived. If users are not satisfied with the results, they can go from the result page (an example is shown in Figure 5.8) back to the query page, select other profiles and submit the query again.

5.3.5 Discussion

In a search system, page filtering and ranking may be performed not only to search results corresponding to various queries but also in the process of specifying search criteria. What needs to be done is to design appropriate user interfaces for specifying profiles and submitting ranking or filtering tasks.

For instance, in a system that supports hypertext context-based search, the ranking and filtering mechanism can help users to define hypertext contexts when they browse the Web. More clearly, a set of Web pages that meet a certain ranking or filtering rule can be added into or filtered out from a user-defined hypertext context. This may speed up the definition of hypertext contexts that reflect users' specific interests well

and furthermore improve the quality of searches performed within the boundary of the hypertext contexts.

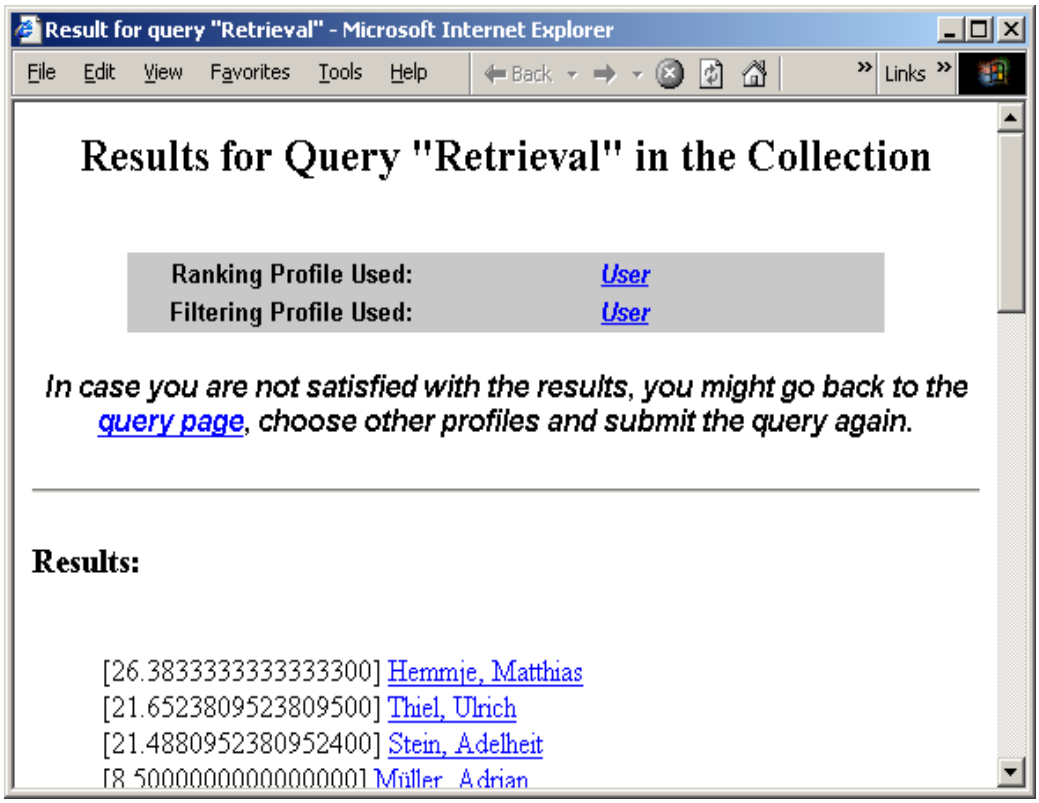


Figure 5.8 An example result page after specified ranking / filtering

5.4 Supporting Method 2: Hypertext Context-Based Search

This section discusses several technologies for the implementation of searching in the Web space by making use of hypertext contexts. It also shows how a search system can help users to form their mental context views when they browse and search the Web. This issue is crucial for having rich hypertext context information available in the Web hyperspace.

5.4.1 Hypertext Context Information Gathering and Indexing

Corresponding to how XML and RDF represent hypertext contexts, the system gathers and indexes the following information (referred to as hypertext context information) for each hypertext context:

- the URI of the Web resource that the context describes about (called aboutURI later),
- the URI of the Web resource that contains the context description (called sourceURI later), and,
- descriptive keywords, which can be extracted from RDF property types, sourceURI, aboutURI, roles of external links, names of the link elements that contain the context, and special descriptive information about the context with regard to certain Web resources.

The JEDI (Java Extraction and Dissemination of Information) [Huck et al. 1998] tool can be used in the system as the extractor for gathering the above information. The tool consists of a wrapper that can collect information by navigating through multiple documents and by explicating their implicit logical structure, and a mediator that maps the collected information to an integrated view.

Since the system may be fed by heterogeneous textual information sources, the translation of the incoming texts to an internal format is helpful for speeding up the indexing process later. The internal system format is provided in form of an XML DTD and aims to cover the demand of describing the hypertext context information exhaustively. Figure 5.9 displays an example hypertext context encoded in the system internal format. The hypertext context is assigned a name. All the descriptive keywords extracted are listed in the content of the *description* element. Moreover, its sourceURI, aboutURI, components are all contained in this document.

Such documents are stored into the document repository of the system. The indexing engine then reads the repository, parses the documents and performs keyword-indexing functions to the content of the element *description*. The indexing results and all URIs related to the hypertext contexts are sent to the database, which owns the schema as described in the following subsection (5.4.2).

```
<context name="DelitePub1999">
<source>http://www.darmstadt.gmd.de/delite/publication/1999.rdf</source
>
    <about>http://www.darmstadt.gmd.de/delite/publication/index.html</a
    bout>
      <description>delite, publication, 1999</description>
      <components>
        <component>
          http://www.darmstadt.gmd.de/delite/publication/1999/pub1.html
        </component>
        <component>
          http://www.darmstadt.gmd.de/delite/publication/1999/pub2.html
        </component>
         ...
      </components>
      </context>
```

Figure 5.9 An example hypertext context encoded in the system internal format

5.4.2 Internal Organization of Hypertext Context Information

In the system, hypertext context information after indexing is stored in a relational database. Every hypertext context has an associated ID number (called contextID) that is assigned whenever a new hypertext context is parsed out of a Web resource. To represent the components of hypertext contexts and the aboutURIs and sourceURIs of

hypertext contexts in a non-redundant way, every Web resource also has an ID number (called docID later) whenever a new URI is parsed out. Similarly, every keyword used to describe hypertext contexts is assigned an ID number (called wordID later) in order not to waste space.

Based on the above basic point of view, the database of hypertext contexts is built with the schema as shown in Figure 5.10. The *URI Table* contains URIs that are parsed and the primary serial numbers assigned to the URIs. The *Word Table* contains keywords that are used to describe the hypertext contexts and their primary serial numbers. The *Context Table* contains the primary serial numbers, names and IDs for aboutURIs and sourceURIs of hypertext contexts. The *Context Component Table* lists the components of hypertext contexts. *The Context-Word Table* builds relationships between hypertext contexts and the keywords used to describe them. Finally, the *Context Parent Table* represents the parent-children relations between hypertext contexts. The *ccID*, *cwID*, and *cpID* are all primary serial numbers in their contained tables.

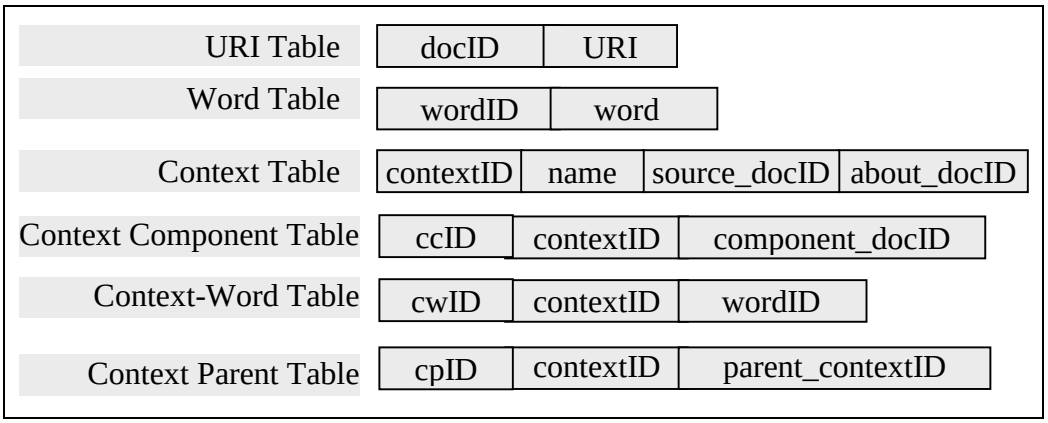


Figure 5.10 Database schema for hypertext contexts

For instance, the example hypertext context encoded as Figure 5.9 may be stored in the database with the following items:

*in the **URI table***: *(docID | uri)*

11 | http://www.darmstadt.gmd.de/delite/publication/1999.rdf

21 | http://www.darmstadt.gmd.de/delite/publication/index.html

31 | http://www.darmstadt.gmd.de/delite/publication/1999/pub1.html

32 | http://www.darmstadt.gmd.de/delite/publication/1999/pub2.html

*in the **word table***: *(wordID | word)*

1 | delite

2 | publication

3 | 1999

*in the **context table***: *(contextID | name | source_docID | about_docID)*

1 | DelitePub1999 | 11 | 21

*in the **context component table***: *(ccID | contextID | component_docID)*

1 | 1 | 31

2 | 1 | 32

*in the **context-word table***: *(cwID | contextID | wordID)*

1 | 1 | 1

2 | 1 | 2

3 | 1 | 3

With this internal organization of hypertext context information, the system is able

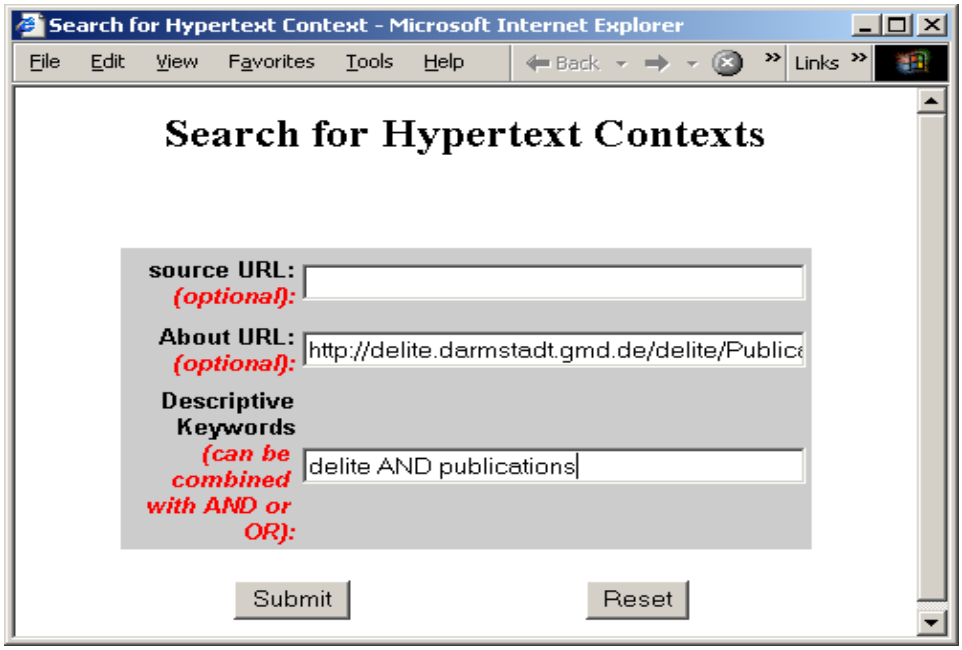


Figure 5.11 Search for hypertext contexts

to support the specification of hypertext contexts as search boundaries, not only single hypertext contexts, but the combination of hypertext contexts as well. Furthermore, searching for hypertext contexts themselves by inputting keywords and/or the aboutURIs and sourceURIs is also possible.

5.4.3 User Interface for Querying and Specifying Hypertext Contexts

The system provides a form-based interface to enable users to query hypertext contexts and to specify hypertext contexts. As shown in Figure 5.11, users can query hypertext contexts by specifying sourceURIs, aboutURIs, and keywords. After users confirm the specification, the system will display all the candidate contexts (in some case, maybe not only one context is found) by showing their sourceURIs, aboutURIs, and all their descriptive keywords (as shown in Figure 5.12). Then users can adjust their queries or go to take a look at the components of the candidate contexts (as shown in Figure 5.13) and/or make their choice of the contexts to be used as the search boundaries for their queries. As Figure 5.12 displays, users can also ask the system to perform a Boolean combination of the hypertext contexts found and use the

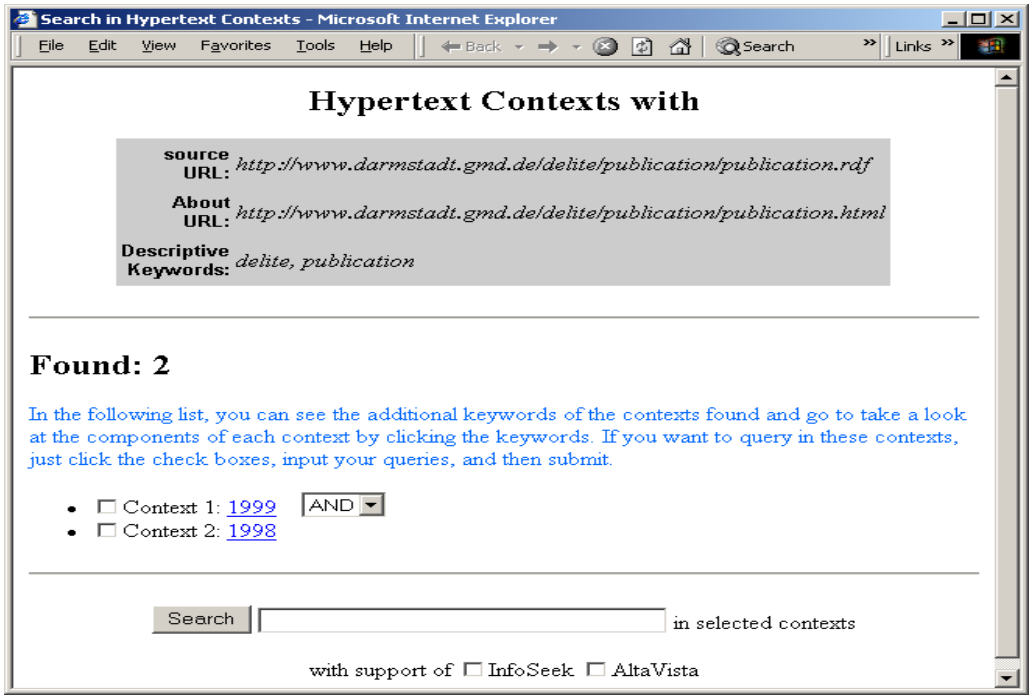


Figure 5.12 Display of found hypertext contexts and enabling searching in selected hypertext contexts

resulting hypertext context to define search boundaries.

5.4.4 Implementing Searches within Hypertext Contexts

To implement searches within the boundaries of specified hypertext contexts, the retrieval engine in a system works in this way: The **structure and semantic-based search engine** sends the query terms that users input to the **keyword-based search engine,** gets those results, and then discards the page that are not contained in the specified contexts from those results.

In our current implementation of HyperSM, as Glimpse [Glimpse] has been chosen as the keyword-based indexing and search engine, and it supports very flexible functions to limit the search to only parts of the files in a collection[1]. The **structure**

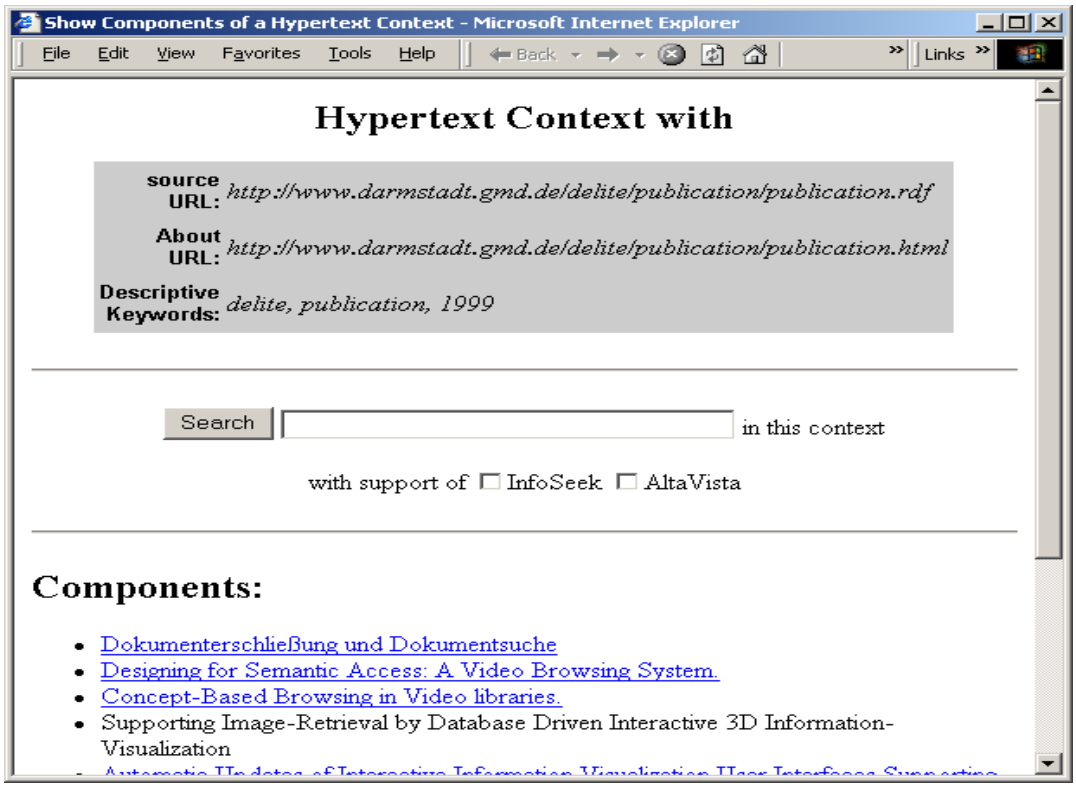


Figure 5.13 Display of components of a hypertext context and enabling searching in the context

[1] Glimpse provides several options to enable users to filter files in search. One option is –f, which reads a list of file names from a given file and uses only those files in search.

and semantic-based search engine works mainly as a search boundary specifier. It maps the URLs of the component pages of the specified hypertext contexts to the file names of the pages that the system collects[2] and saves the file names in a context file. This context file is then compressed and sent to Glimpse together with the query terms that users input. Finally, the **structure and semantic-based search engine** maps the filenames returned by Glimpse back to their URLs and provide the URLs as final search results to users**.** In this way, searching within the boundaries of specified hypertext contexts is implemented.

In the future, HyperSM will integrate with other Web search engines like, e.g. InfoSeek [InfoSeek], AltaVista [AltaVista], etc., as well as meta search engines like, e.g., SPOMSE [Huang et al. 2000]. It will send queries to one or more those systems according to users' selections (example interfaces as shown in Figure 5.12 and 5.13), combine the search results from those systems, and then use the specified hypertext contexts as filters to compute the final search results. With this combination and filtering process, it will provide users with more specific final results that are relevant to their information needs and save users much time for retrieving the information by themselves.

In case those search engines restrict the number of results to, e.g., 100 out of possibly several thousands, HyperSM requires a more tight cooperation with them. That is, HyperSM should be allowed to access the whole or a large part of the ranked result lists produced by them, so that the pages that are within the specified context but are in the tails of the lists will not be missed.

An issue related to this integration approach is scalability. If the result lists produced by other engines are really large, it should be necessary for HyperSM to take only a limited amount of the pages in the lists as input for the combining and filtering process. This amount should be able to be adjusted by users in their query activities depending on whether their information needs are satisfied. In this way, HyperSM can reduce the computation complexity in the process and provide users specific search results with a balanced quality (precision and recall) and speed.

[2] Currently HyperSM collects all remote pages locally with a mapping mechanism from urls to file names.

5.4.5 Let Users Specify Hypertext Contexts When They Browse and Search

With respect to a specific topic or subject domain, applying hypertext contexts as a mechanism to specify the information space to be examined in a search activity should prove to be useful for improving the quality of the search and of the filtering process. However, the mechanism is valuable only when rich hypertext context information is available in the Web space. Though document providers can encode such information with the new Web standards directly, it will still be necessary for a search system to provide facilities to enable users to express their hypertext context views when they browse and search the Web. The system then constructs the instantiations of hypertext contexts automatically, stores them and provides them to users for being selected in later queries.

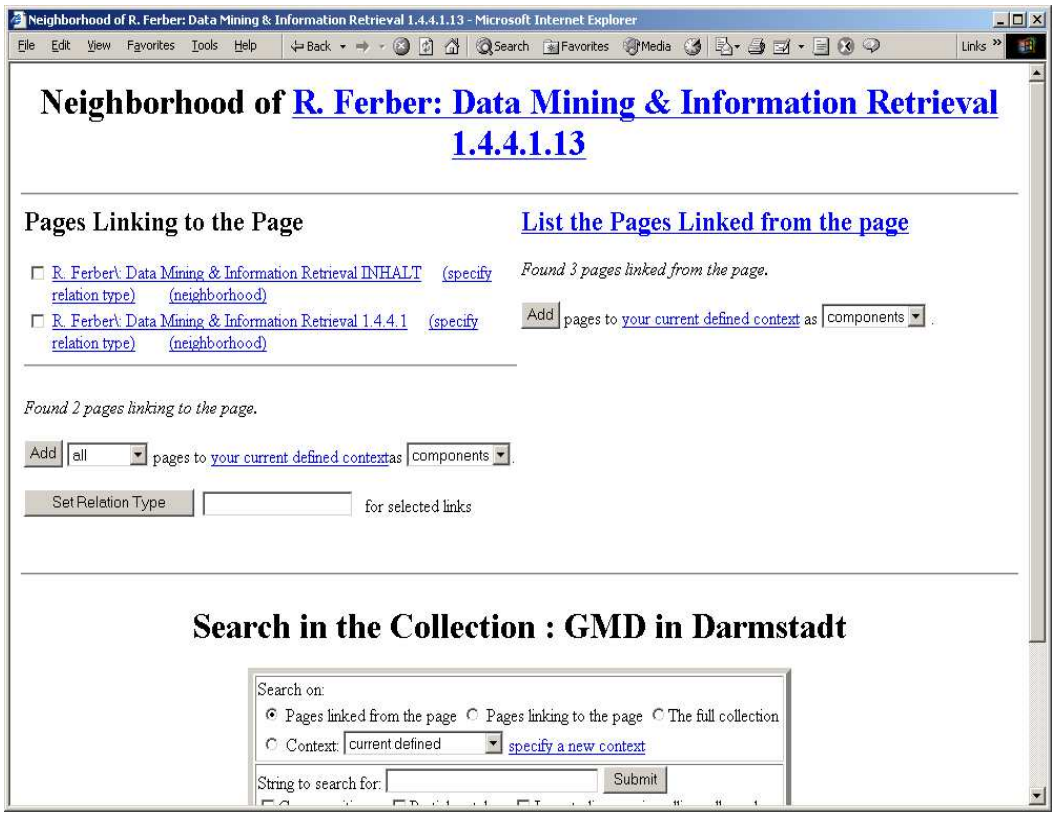


Figure 5.14 A neighborhood page

In the subsection 4.2.3 earlier, this thesis has presented a model for describing users' hypertext context views and the view construction process. Based on this model, we regard commonly displaying neighborhoods of pages and supporting search in neighborhoods as basic facilities to help users to form their hypertext context views. These facilities are practical because the neighborhood information of pages is stored in the link base of the system and is thus easily to be presented, and searching in neighborhoods is a specific kind of hypertext context-based search. It is believed that the context-based search itself also helps in the construction process of users' context views.

With these facilities, hypertext contexts can be specified flexibly while users browse and search the Web. Following the "specify a new context" link in a neighborhood page (an example is shown in Figure 5.14), users will receive a page as displayed in Figure 5.15. They can define a new context by inputting some urls (each separated with ",") as seed pages and by specifying whether the ancestors or/and descendants of the seed pages within a certain distance will be included in the context. They can specify whether to include selected single pages in the context, and whether to exclude some urls (each separated with ",") from the context.

Furthermore, the neighborhood pages allow users to select any neighborhoods of

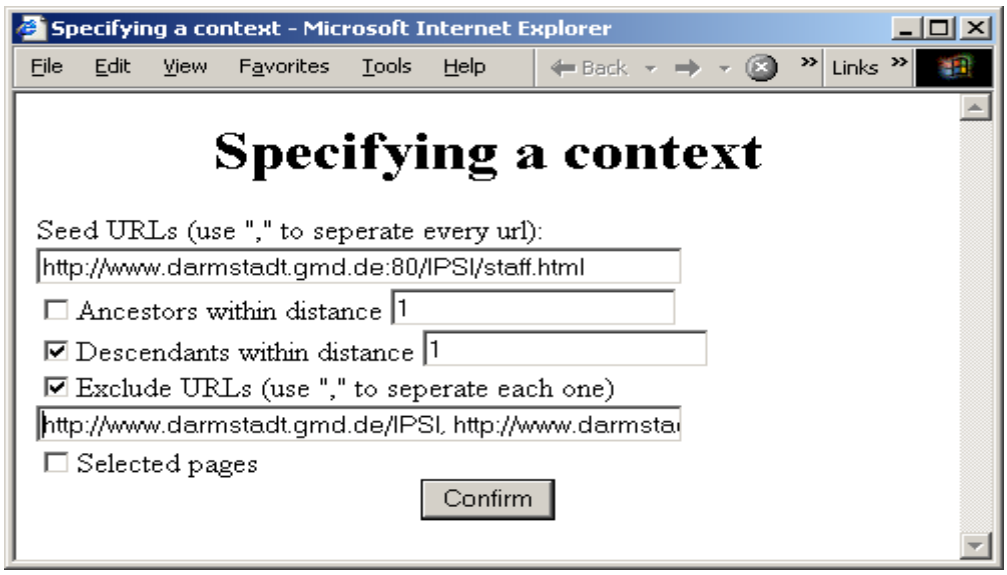


Figure 5.15 Specifying a hypertext context

the current page into the current defined context as components or as seed pages (recall that seed pages themselves are not components of the context unless specified explicitly in other ways). In the latter case, users can still specify whether the ancestors and/or descendants of the selected pages within some distance will be included in the context. This means that users can specify different distances for different seed pages.

If there is no current defined context but users click the link "your current defined context" or the button “Add”, they will get a new context, which includes the selected pages or is just empty and the system will advise them to specify one now.

After "confirming" their selections, users will be provided with all seed pages, each with the quantity of its ancestors/descendants and links to the ancestors/descendants, all the selected single pages and all the excluded pages in their context view (an example is given in Figure 5.16). They will know how many pages are included in the resulting context and may decide whether to follow the *"show the list"* link to take a

Figure 5.16 A user-defined hypertext context about DELITE member

look at the list of the components (pages) in the context. By taking a look at the list, they can still decide whether they want to exclude some of the pages and then confirm their selections again. This is indeed a loop that makes the evolution of the context possible.

At any time, when users are not satisfied with the search result in their defined context, they can define another context by enlarging or narrowing down the scope that the context covers or making totally different choices of seed pages and/or single pages. That is why we say context-based search is itself a facility to help users to form their context views.

Finally, hypertext contexts specified by authorized users can be annotated with a set of descriptive keywords (e.g. “DELITE staff member” as shown in Figure 5.16) and submitted to the hypertext context base for later reuse. That is, the hypertext contexts stored in the database can be used as basis for constructing new contexts, or used in future search activities, as the examples shown in Figure 5.14 and Figure 5.7.

5.5 Supporting Method 3: Structured Query and Search Based on Hypertext Composites

This section describes how the system implements the structured query and search method presented in Section 4.3 for taking advantage of hypertext composites.

5.5.1 Hypertext Composite Detection

To support structured query and search based on hypertext composite information in a collection, the system needs to be able to detect such information at first.

If the link information in a collection has already been gathered and stored in the link base of the system, any hypertext composites contained in the collection can be computed dynamically at search time (our prototype system does so at present) and then be used to derive structured search results based on them. However, for performance reasons, it should be better to compute the hypertext composites in a collection first and store them in the database of the system so that the system does

not need to spend time for this part of computation when performing structured search tasks.

How to gather and store link information in the system has been described in the subsection 5.3.1 in this thesis.

5.5.2 Computing Structured Search Results

To compute structured search results based on hypertext composites, the **keyword-based search engine** and the **structure and semantic-based search engine** in the system work together in this way: when receiving a query request, the **structure and**

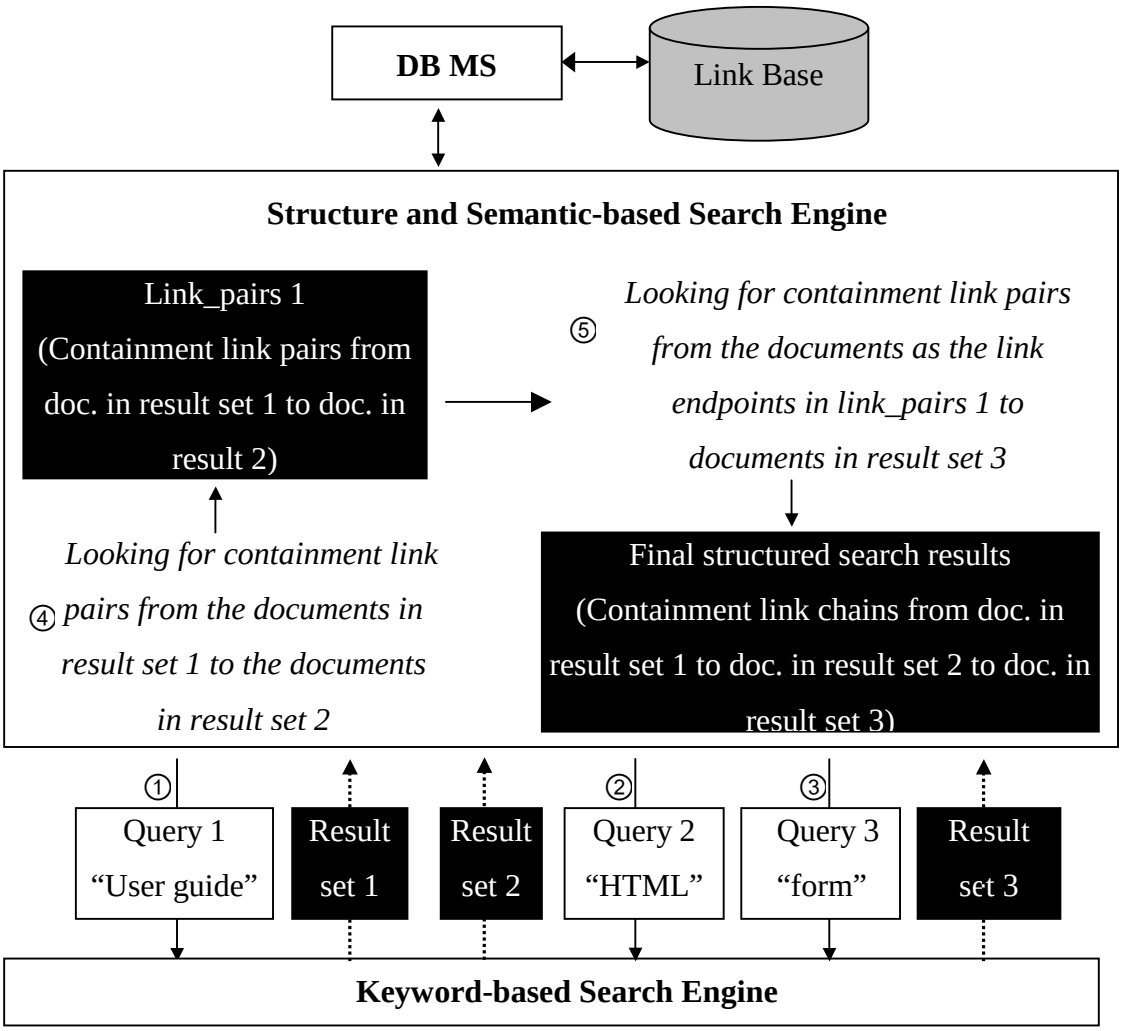


Figure 5.17 Deriving structured search result making use of hypertext composites (an example)

semantic-based search engine first sends query terms that users input in each query structural level to the **keyword-based search engine** and gets distinct responding results, and then uses the link information stored in the link base as filters to get structured search results.

For instance, the process to derive the structured search results for the example structured query with 3 levels

$$\overline{SQ} = \wedge(\text{"User guide"}, \text{"HTML"}, \text{"form"})$$

operates as displayed in Figure 5.17 and is described as follows:

The so-called *result set 1* is a set of documents that contain "User guide". The so-called *result set 2* is a set of documents that contain "HTML". The so-called *result set 3* is a set of documents that contain "form". The so-called *link_pairs 1* is a set of link pairs. In each link pair the "from" node belongs to the result set 1, the "to" node belongs to the result set 2. Finally, the so-called *final search results* are a set of link chains. In each chain, the first node belongs to the result set 1, i.e. contains the term "User guide". The second belongs to the result set 2, i.e. contains the term "HTML". The third belongs to the result set 3, i.e. contains the term "form". The link pairs or link chains are constructed based on the links of the types that represent containment relations in the link base.

5.5.3 Adaptive Form-Based Interface for Formulating Structured Queries

To enable users to formulate structured queries based on hypertext composites in an easy and flexible way, an adaptive form-based user interface is provided in the system. The adaptivity of the interface is mainly reflected in the adjustability of structured query levels. That is, if users first select 2 levels but get unsatisfactory results, they may ask the system to adjust the structured query levels to 3 or more. Every time when the value of the structured query levels is modified, the system will regenerate the form. An example query in such an interface is shown in Figure 5.18.

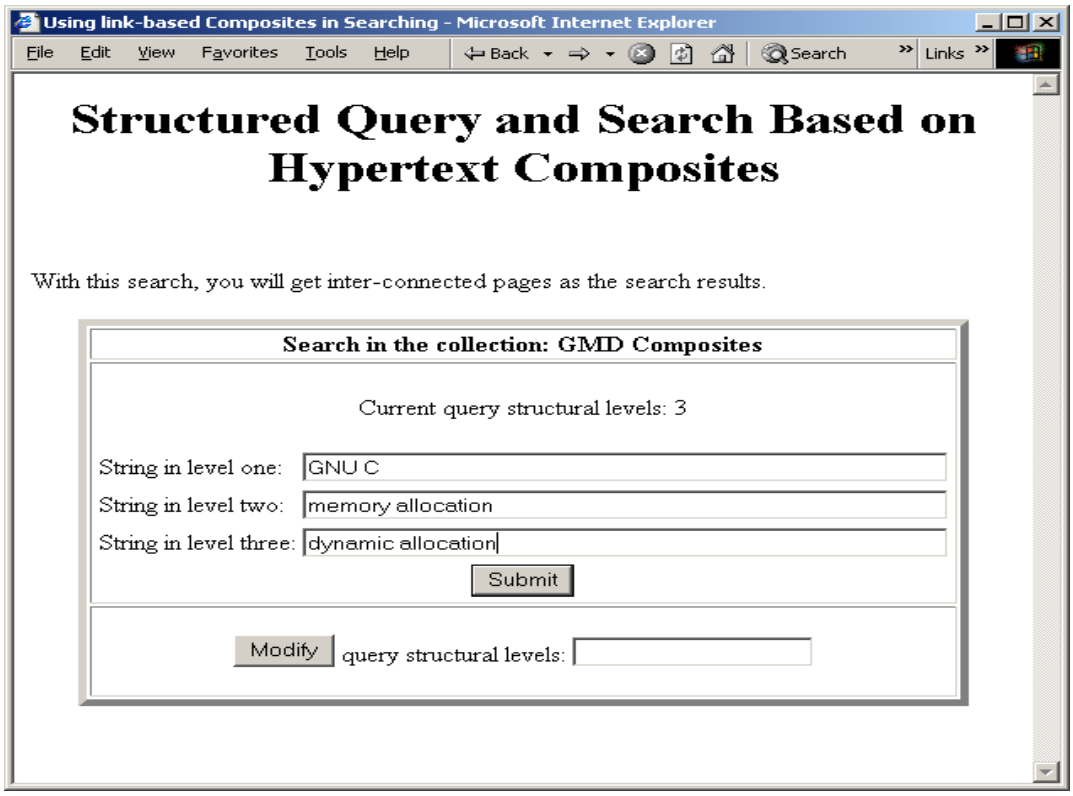


Figure 5.18 Form-based interface for formulating structured queries based on hypertext composites

5.5.4 Presenting Structured Search Results

As described in Section 4.3, the structured search results derived from the structured queries based on hypertext composites are not separate nodes but sets of inter-linked nodes. How to present them to users is also crucial for the system's success. A good presentation may improve users' satisfaction with the results and enable users to get more contextual information about the results and even the relevant resources.

At the moment the prototype system just presents the structured search results in a simple but integrated way. A table, which contains the designated structured query levels as the number of columns, is built to contain the results. Each row represents

one result. A screenshot for an example result presentation corresponding to Figure 5.18 is given in Figure 5.19.

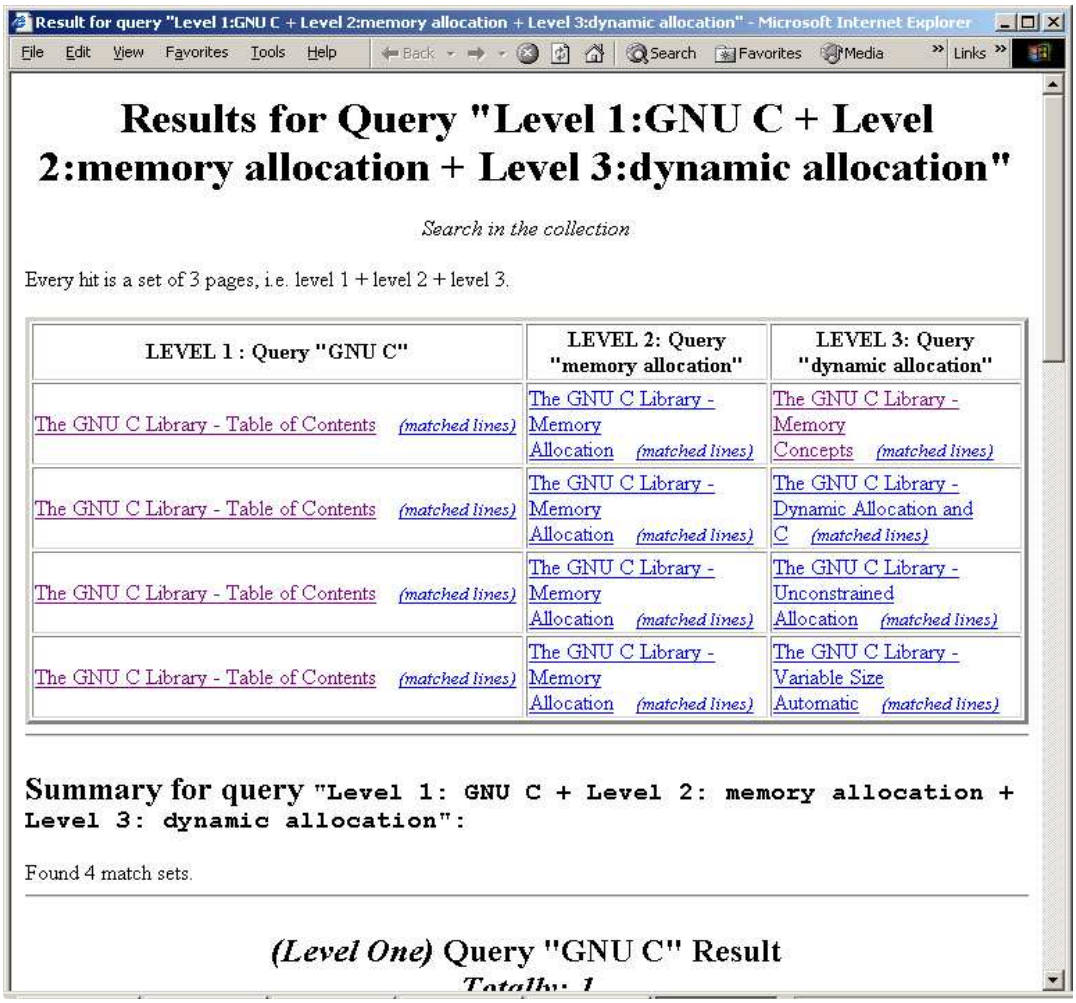


Figure 5.19 Presenting structured search results based on hypertext composites

5.6 Supporting Method 4: Structured Query and Search with Link-Based Domain Models

This section discusses how our HyperSM system implements the structured query and search method presented in Section 4.4 for taking advantage of link-based domain models.

5.6.1 Domain Model Processing

To apply link-based domain models in a structured query, the system needs to extract from RDF descriptions for the models RDF classes representing domain model concepts and RDF properties representing relationships between the concepts. It then stores this kind of information in an object relational database. To implement this function, a number of commercial and noncommercial RDF software components (as provided at http://www.w3.org/RDF/) can be chosen and adapted for use. So are some approaches to storing RDF data in a relational database (provided at http://WWW-DB.Stanford.EDU/~melnik/rdf/db.html).

It is not necessary for us to describe in detail how the store function is implemented. This thesis just presents the database schema designed specifically for holding multiple link-based domain models. As Figure 5.20 shows, the schema corresponds to the standard representation of link-based domain models as described in the subsection 3.2.2.6.

That is, the ***model table*** holds the URIs of all RDF resources for describing link-based domain models. The *modelId* field is an internal identifier field and is referenced by *modelId* in the other tables. The ***class-concept table*** holds RDF classes describing link-based domain model concepts. The ***property-relationship table*** holds RDF properties describing relation types in domain models. The ***statement table***

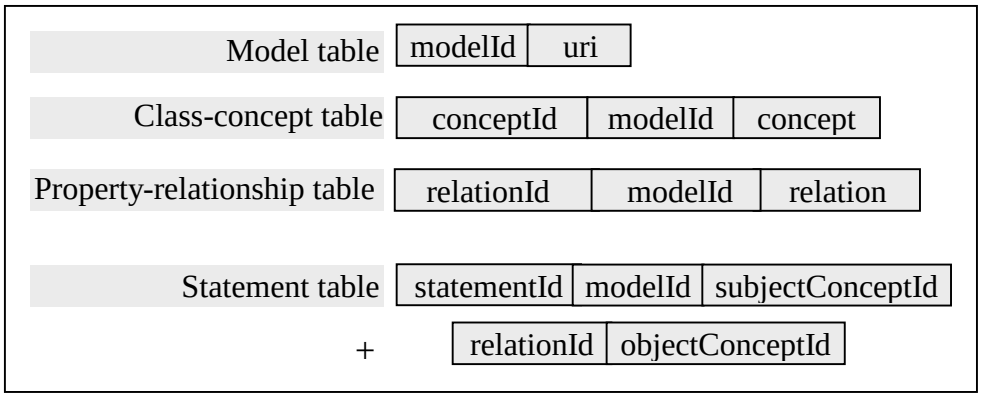


Figure 5.20 Database schema for holding multiple link-based domain models

holds the statements for describing the semantic links (relationship) between concepts in domain models. Each of these tables contains a field for primary serial numbers assigned by the DBMS and for references from the other tables.

For instance, the RDF schema describing the simple link-based domain model "A book is written by an author" as given in the subsection 3.2.2.6 will be stored in the database with the following items:

*in the **model table***: *(modelId | uri)*

20| http://www.domainmodels.com/booksite.rdf

*in the **class-concept table***: (*conceptId | modelId | concept)*

1 | 20 | book

2 | 20 | author

*in the **property-relationship table***: (*relationId | modelId | relation)*

1 | 20 | writtenBy

*in the **statement table***: *(statementId | modelId | subjectConceptId | relationId | objectConceptId)*

1 | 20 | 1 | 1 | 2

5.6.2 Web Resource Indexing with Domain Model Concepts

To enable structured queries, the system implements the indexing of Web resources (HTML, XML documents etc.) with domain model concepts. The essence of this indexing is to create a set of semantic links that describe the roles of the domain model concepts to the Web resources in the system, as described in the subsection 4.4.1

No doubt, for different purposes the roles of the concepts may be quite different, even if the domain model used is the same. The indexing engine should thus provide users a vocabulary for describing the necessary roles of domain model concepts or enable users to define such a vocabulary themselves for given application domains. Then users should be able to choose any defined roles to describe the relations between domain model concepts and Web resources to be indexed.

The engine can be implemented as a particular interactive tool. However, it may be a good alternative to take advantage of an available RDF authoring/editing system that enables users to create RDF descriptions and then build an extractor that can extract from the RDF descriptions created relationship information between domain model concepts and Web resources handled. This is because the vocabulary for the concept roles can be represented using an RDF schema (referred to as *role schema* later) and the relations between domain model concepts and Web resources can also be described with the RDF format. Furthermore, there are a number of commercial and noncommercial groups who are designing RDF software (as shown at http://www.w3.org/RDF/) so that the indexing engine may be implemented with less effort in this way.

The following example RDF encoding describes two relationships. One is between the concept "Book" and the resource "*http://books/maths.html"* (a book). The other is between the concept "Author" and the resource "*http://authors/Smith.html*" (an author).

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:booksite="http://descriptions/domainmodels/booksite/"
        xmlns:roledef="http://descriptions/domainmodels/conceptroledef/">
    <rdf:Description about="http://books/maths.html">
          <roledef:category
            resource="http://descriptions/domainmodels/booksite/Book"/>
           <roledef:label>Maths</roledef:label>
           <booksite:writtenby>
                  <rdf:Description about="http://authos/Smith.html">
                      <roledef:category
                          resource="http://descriptions/domainmodels/booksite/
                  Author"/>
                     <roledef:label>Smith</roledef:label>
                   </rdf:Description>
            </booksite:writtenby>
    </rdf:Description>
</rdf:RDF>
```

Both of the relationships are “category” defined in a schema that owns the namespace prefix “roledef” identified by the URI reference "http://descriptions/domainmodels/conceptroledef/". The *roledef:label* is used for representing the label assigned to the resource being indexed. As explained in the subsection 4.4.1, this label distinguishes this resource from the other resources indexed with the same concept. Furthermore, the relationship information between concepts - “*writtenby*” - has also been expressed in the encoding with the tag “*booksite:writtenby*” and thus the relationship between the book “*http://books/maths.html*” and the author *http://authos/Smith.html* has been annotated.

From such RDF descriptions the extractor can gather relationship information between domain model concepts and Web resources being indexed, i.e., the indexing results, and stores the results in the database. As well, the relationship information between the Web resources should also be extracted from the RDF descriptions and stored in the database so that structured searches based on domain models can be supported. The basic tables as shown in Figure 5.21 are designed to make the storage efficiently.

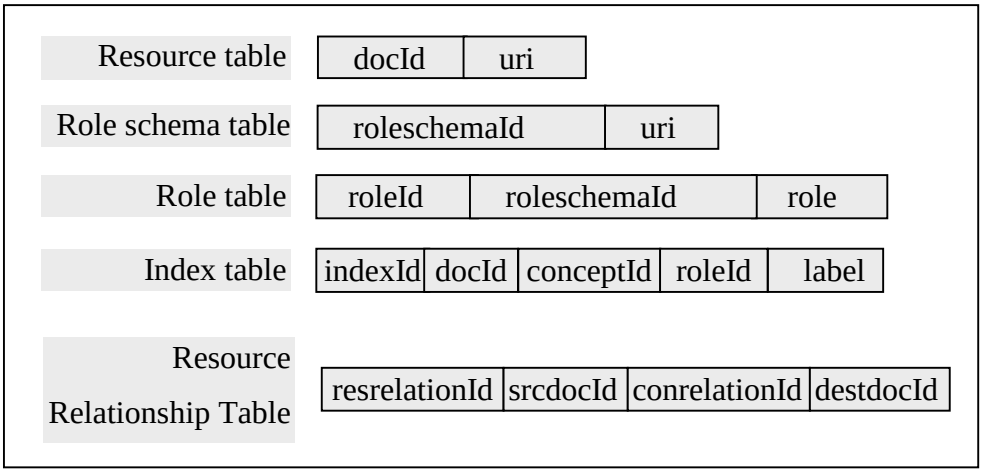


Figure 5.21 Database schema for storing Web resource indexing results with domain model concepts and the relationships between the resources indexed

The ***resource table*** contains the URIs of all Web resources indexed with domain model concepts in the system. The ***role schema table*** holds the URIs of all RDF

resources for describing roles of domain model concepts. The ***role table*** contains all roles defined in the role schemas collected in the system. The ***index table*** is for storing the indexing results. Finally, the ***resource relation table*** contains the relationships between the Web resources indexed.

Each of these tables contains a field (*docId*, *roleschemaId*, *roleId* and *indexed, resrelationId* distinctly) for holding a serial number assigned by the DBMS and for references from other tables. The fields in the other tables (except ***the resource relationship table***) referring to this field have a same name, e.g. the *roleschemaId* in the ***role table*** refers to the *roleschemaId* in the ***role schema table***. Especially, the *conceptId* in the ***index tab*le** refers to the *conceptId* in the ***class-concept table*** as shown in Figure 5.20. In the ***resource relationship table,*** the *conrelationId* refers to the *relationId* in the ***property-relationship table*** as shown in Figure 5.20, while the *srcdocId* and *destdocId* refer to the *docId* field in the ***resource table***.

For instance, the indexing results expressed in the example RDF encoding given before, which describes the relationship between the concept "Book" and the resource "*http://books/maths.html"* (a book) as well as the relationship between the concept "Author" and the resource "*http://authors/Smith.html*" (an author), may be stored in the database with the following items:

*in the **resource table*** : *(docId | uri)*

1 | http://books/maths.html

22 | http://authors/Smith.html

*in the **role schema table***: *(roleschemaId | uri)*

10 | http://descriptions/domainmodels/conceptroledef

*in the **role table***: *(roleId | roleschemaId | role)*

21 | 10 | category

*in the **index table***: *(indexId | docId | conceptId | roleId | label)*

1 | 1 | 1 | 21 | maths

3 | 22 | 2 | 21 | Smith

The relationship between the book "*http://books/maths.html*" with the author "*http://authors/Smith.html*" may be stored as:

in the ***resource relationship table***: *(resrelationId | srcdocId | conrelationId | destdocId)*

15 | 1 | 1 | 22

5.6.3 Form-Based Interface for Structured Queries with Link-Based Domain Models

To enable users to formulate structured queries with link-based domain models easier, the query construction support engine in the system provides users a specific form-based interface rather than a formal structured query language. This interface is built upon the general structured query model proposed in the subsection 4.4.2.

A general algorithm for constructing the dynamic form-based user interface for a given domain model is:

```
CreateForm(domain_model)
BEGIN
conceptlist <- all concepts in the domain model;
relationlist <- all relationships between concepts in the domain model;
create a selection field for listing all items in conceptlist, set the first selection
  as "ALL"
IF (an item in conceptlist is selected) THEN
 /* construct a  neighborhood-oriented structured query */
   BEGIN
    neighborlist [selected] <- all neighborhood concepts of the selected
      concept;
    FOREACH neighborhood concept in the neighborlist[selected]
        BEGIN
         display the relationship between this neighborhood concept and the
           selected concept;
         display this neighborhood concept;
         rolelist <- all  the pre-defined roles of this neighborhood concept for
           indexing;
         if (the number of items in rolelist > 1) THEN
           BEGIN
```

```
            create a selection field for listing all  items in rolelist;
          END
          create a text input query field for specifying query terms for this
            neighborhood concept;
        END
      END
    IF ("ALL" is selected) THEN
      / * construct a general structured query */
      BEGIN
        FOREACH concept in the conceptlist
          BEGIN
            display this concept;
            rolelist <- all  the pre-defined roles of this concept for indexing;
            if (the number of items in rolelist > 1) THEN
              BEGIN
                create a selection field for listing all  items in rolelist;
              END
          create a text input query field for specifying query terms for this
            concept;
        END
      END
    create a button for submitting the query;
    END
```

All computation in the process is based on the data in the database. The interface can be implemented as any kinds of dynamic page, e.g., ASP or JSP, only if the page is able to access the database directly or indirectly. Figure 5.22 shows an example form created by the algorithm. The domain model used for this example is as shown in Figure 6.1 later in this thesis. The structured query presented in this interface is a neighborhood-oriented one, which searches for DELITE members who *work in* the project "DELITE Online", *attend* the conference "DL'2000" and *present* the demo&prototype "DELITE Online". As the indexing role list (*rolelist*) for each concept has only one item, i.e. "category", no selection fields for the roles have been shown in the form.

If a domain model is very large, the construction of the dynamic interface for supporting the general structured queries based on the whole model seems not so convenient. This is because such an interface will be also very large and thus probably hard to be accepted. In this case, the algorithm can still be used to construct the interface for supporting the neighborhood-oriented structured queries. For supporting the general structured queries, some kinds of query mechanisms should be developed so that users will be able to query the domain model and specify the concepts to be contained in their queries.

To meet particular demands for the interface in a domain, the algorithm will then be refined and adapted.

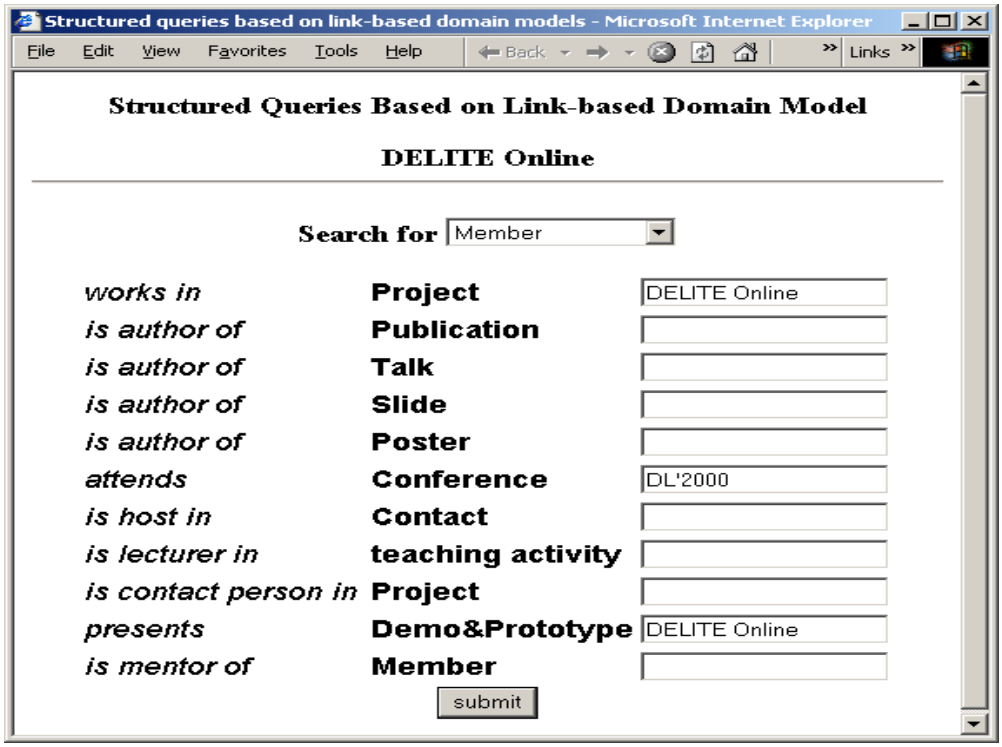


Figure 5.22 A sample query based on a link-based domain model (in form-based interface)

5.6.4 Deriving Search Results for Structured Queries

The search results for the structured queries based on a domain model are a set of Web resources indexed with the domain model concepts. The query terms used in the structured queries can be

(1) only labels in the Web resource indexing results with domain model concepts, or

(2) other keyword, phrases, value, or Boolean expressions.

In case (1), with all the indexing results efficiently stored in a relational database (as described in the subsection 5.6.2), to compute search results for structured queries is quite easy. The **structure and semantic-based search engine** in the system just needs to formulate SQL queries and send the queries to the DBMS.

In case (2), the **structure and semantic-based search engine** needs to send the query terms that users input to the **keyword-based search engine** to get the Web resources that contain those terms, and then makes use of the concepts and relationship information in the domain model as filters to produce the final domain-specific search results.

In our current implementation of HyperSM, the form-based interface does not distinguish the two cases. To derive search results for the structured queries, the system first dealing with the queries as case (1), and then, if no results returned, it tries to derive search results as dealing with case (2).

5.7 Summary

This chapter has presented a prototype system HyperSM that is designed to implement the hyperstructure-based search methods proposed in Chapter 4. The specific issues including database modeling, query construction support, retrieval engine and result presentation have been discussed for supporting each search method.

6 Evaluation Studies

The most important measure of a search system is the quality of its search results. This chapter describes some experimental evaluation studies carried out to test the result of the methods and tools explored in this thesis. As almost all of the published evaluation work e.g. [Hawking et al. 2001; Hawking et al. 2000; Hawking et al. 1999; Leighton&Srivastava 1999] about Web search engines, we have also focused on the measurement of precision, but not recall, because it is very difficult to assess recall in a huge, ever changing collection like the Web and each Web search engine searches only a varying sample of the Web.

The evaluation studies include:

- Preparing a test strategy
- Building a test collection
- Designing a test set of queries
- Testing searches in the prototype system HyperSM
- Testing searches in other available systems
- Comparison and analysis of search results from different systems or with different prerequisites (e.g. various search boundaries)

6.1 The Test Strategy

The general hypothesis in the evaluation studies is that the hyperstructure-based search methods presented in this thesis can help to improve the search quality on the Web. In detail, the hypothesis includes:

(1) The ranking/filtering mechanism applying link types will help to provide users search results that are relevant to their information needs in a better order than in traditional search engines.

(2) Applying hypertext context as a mechanism to specify the scope of the information space to be examined in a search will improve the quality of the

search results concerning a specific topic or subject domain and save users' effort to get the results.

(3) Supporting structured query and search based on hypertext composites can help to improve the precision of the search results and enable users get more contextual information about the search results.

(4) Domain-specific structured search methods can generate better results than general search methods that do not understand the domain-specific information. By applying link-based domain models in formulating structured queries, users can receive more specific results relevant to their information needs.

These hypotheses will be tested in the following sections.

6.2 Preparation of the Test Collection

As the current Web does not provide a large document collection with rich explicitly represented hyperstructural information for performing the evaluation, we have built a test collection for our experiments by taking advantage of the info agent and the auxiliary tools in the system. That is, the info agent was used to gather HTML pages from the Web, while the auxiliary tools were used to annotate node/link types

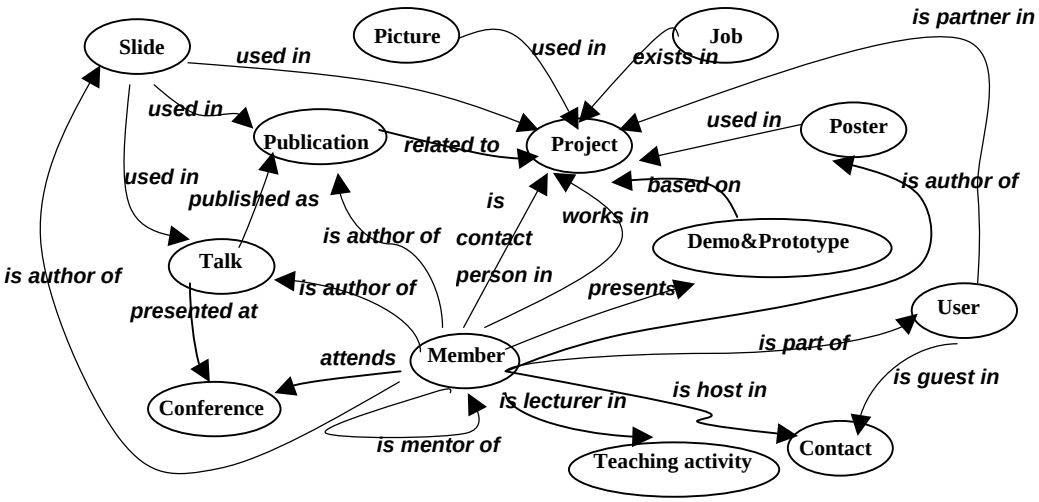


Figure 6.1 Link-based domain model used for the test collection

and construct hypertext contexts.

The test collection contains two partitions of Web pages. The first partition was collected from the GMD[3] Darmstadt site (starting from http://www.darmstadt.gmd.de/) and the number of the pages collected is 29574, of which 24498 own outgoing links and 27461 own incoming links. The second partition was collected from the whole GMD site (http://www.gmd.de/). The pages collected are only those about technical manuals that were recognized as composites or composite components (as they contain links with anchor texts like "*Table of Contents*", "*Previous*", "*Next*" and the like). The number of those pages is 11025, of which 10999 own outgoing links and 11014 own incoming links.

A link-based domain model as shown in Figure 6.1 was applied to organize the pages adaptable to the model and then to construct structured queries. Furthermore, the concepts in the model and the relationships between the concepts were used as node types and link types in introducing annotated nodes or links in the collection. These node types and link types were then applied in our experiments for testing the ranking and filtering mechanism proposed in this thesis[4].

6.3 Testing Method 1: Page Ranking and Filtering Based on Link Types

This part of the experiments was set up to support the hypothesis that the ranking/filtering mechanism applying link types will help to provide users with search results that are relevant to their information needs in a better order than traditional search engines.

One example query was to find the DELITE (a research division in GMD) experts in a certain research field, e.g. "retrieval", by taking into account their publications

[3] GMD – German National Research Center for Information Technology (www.gmd.de) was merged into the Fraunhofer Gesellschaft (www.fraunhofer.de) in January 2001. Its Darmstadt site (www.darmstadt.gmd.de) contains mainly information about two research institutes: IPSI and SIT, i.e. their staff members, publications, projects, events, and so on.

[4] In the experiments up to now, only the part of the pages and links that were suspected to be relevant to our test set of queries has been annotated. This should not have influenced our result comparisons, because even if the whole set of the collection is semantically annotated, one would end up in that domain for those queries anyway.

and projects. To attain this, we first annotated the links from the pages for publications to the pages for staff members with the type "hasAuthor" (the reverse of the type "is author of"), and the links from the pages for projects to the pages for staff members with the type "hasContributor" (the reverse of the type "works in"). Then we specified that the ranking propagational rate would be *1* if a link was of the type "hasAuthor" or "hasContributor" and the source and destination node of the link contained a certain keyword, e.g. "retrieval" (a kind of content similarity). Otherwise, the ranking propagation rate was set to *0*. This ranking profile setting is as shown in Table 6.1 (*Profile 1*).

Furthermore, based on specific filtering profiles, the pages for DELITE staff members, publications and projects were grouped distinctly into 3 hypertext contexts, "DELITE staff" (37 pages), "DELITE publications" (97 pages) and "DELITE projects" (30 pages). The rank of each page in the contexts "DELITE publications" and "DELITE projects" was set to 1 if the page contained the keyword "retrieval".

	Profile 1	**Profile 2**
	RPR	**RPR**
If a link is of the type "hasAuthor" or "hasContributor" and the source and destination node of the link contain "retrieval" (a simplied case of content similarity)	1	1/n (n = the number of the links of a certain type from a source node)
the other	0	0
Note: *The source nodes are publication or project pages.* *The Destination nodes are personal pages*		

Table 6.1 Specification of two ranking profiles in experiments

With the ranking profile specified, searching "retrieval" in the context "DELITE staff" (the query page as shown in Figure 5.7) produced a quite rational ranking list for the DELITE experts in the retrieval research area. The first 3 persons *(Thiel, Stein, Hemmje)* in the list were all senior scientists and mentors of other staff members in the field.

When another ranking profile (*Profile 2* in Table 6.1), say the ranking propagation rate for the selected link types and source and destination nodes was *1/n* (*n* is the number of the links of a certain type in a source node), was specified for the task, the resulting ranking list looked even better. As shown in Table 6.2, the first 3 persons were ranked at the same positions, while some others later were assigned more reasonable positions in the list. An interpretation of this difference may be that the first ranking profile does not take into account the different numbers of a specifically typed link in a page. It reflects the feeling that the impact of a linked page *p1* to a page *p2* is less if the page *p1* has links of a special type to many other pages. And this impact may vary in different applications.

Staff Member	Ranking with Profile 1	Ranking with Profile 2
1	39	15.5995
2	28	12.5664
3	14	7.0831
4	10	3.5831
5	3	1.3333
6	3	1.25
7	3	0.7
8	2	0.6666
9	1	0.3333
10	0	0

Table 6.2 Two experimental ranking results

6.4 Testing Method 2: Hypertext Context-Based Search

This part of the experiments was set up to support the hypothesis that applying hypertext context as a mechanism to specify the scope of the information space to be examined in a search will improve the quality of the results concerning a specific topic or subject domain and save users' effort to get the results.

One example query was that we wanted to find which people in GMD-IPSI (now FhG-IPSI) were doing work in a research field, e.g. information retrieval. To get the results, we defined a hypertext context "IPSI staff" that contained all the personal homepages of GMD-IPSI staff as its components *(sum: 86)*. We then submitted the query "information retrieval" in a form-based interface (examples as shown in Figure 5.7, Figure 5.12, Figure 5.13, and Figure 5.14) that supported the specification of the hypertext context as the search boundary. In this case, we received only 6 hits (as shown in Figure 6.2) and these 6 hits really indicated *most of* the persons who did work in the "information retrieval" area. The precision of the search was 100%, and the recall was 75%[5], because there were two other persons who claimed their research interests or activities in the "information retrieval" area not in their personal home pages but in their Curriculum Vitae pages (not contained in the hypertext context defined). Comparatively, if the search was done in the whole collection, we received 697 hits (at the time we performed the experiments) and had to filter the pages ourselves. Similarly, if we queried "multimedia" in the context in order to find multimedia experts in the institute, we received 18 hits (the precision was also 100%). If we queried "multimedia" in the whole collection, the number of the hits was over 1000. We can imagine how large the number of the search hits can be when the search is done in the whole Web with global search engines.

[5] A 100% recall can be attained only when the hypertext context specified for a query really contains all pages relevant to the query. In many cases, this is impossible. However, for searching in a large information space as the Web, we don't need to care too much about the recall of searches. This view has been commonly accepted in the Web search community.

In the experiments, we noted that the quality of context-based search results depended directly on the type of queries and the quality of the contexts. And the quality of user-defined contexts at present depended to a large degree on users' own

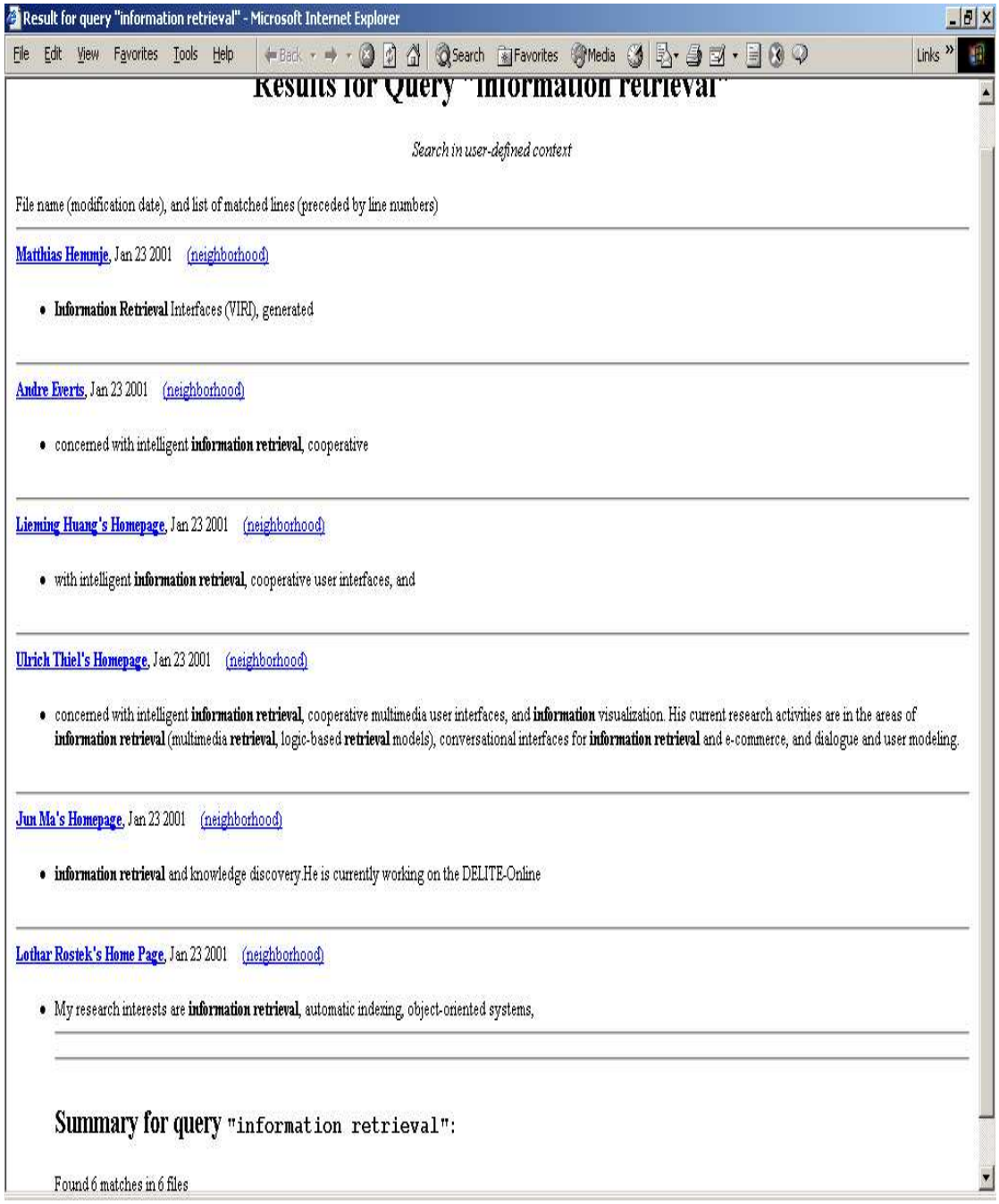


Figure 6.2 Experimental results for querying "information retrieval" in the context "IPSI staff"

relevance judgments about the related Web resources. This is because the facilities that are provided by the current version of the system to help users to form their context views are still quite limited. These facilities include showing neighborhoods, supporting search in neighborhoods, and context-based search itself as well.

The execution time, measured from the time when a user clicked the Submit button in a page enabling context-based search (examples as shown in Figure 5.12, Figure 5.13, and Figure 5.14) to the time when the results page appeared in the browser, increased as the amount of the hits grew. In this aspect there were no significant differences between a context-based search and search in the full collection or search in neighborhoods.

6.5 Testing Method 3: Structured Query and Search Based on Hypertext Composites

This part of the experiments was set up to support the hypothesis that supporting structured query and search based on hypertext composites can help to improve the precision of the search results and enable users get more contextual information about the search results.

In the experiments, we aimed at providing users with more specific search results when they queried about the technical documents, online user manuals, teaching materials and other kinds of well-organized hyperdocuments in the GMD site (http://www.gmd.de/).

An example request was to find the information about dynamic memory allocation in GNU C. To attain this, a structured query “Level 1: GNU C + Level 2: memory allocation + Level 3: dynamic allocation” was constructed, i.e., to search for node chains which contained “GNU C” in the first level, “memory allocation” in the second level and “dynamic allocation” in the third level. Finally, the system found 4 match sets (as shown in Figure 5.19). Comparatively, with no level specification the search hits from the simple query “GNU C” was 1262, while querying “memory allocation” produced 49 hits and querying “dynamic allocation” produced 30 hits. If all the three keywords together, i.e. “GNU C + memory allocation + dynamic allocation” were

submitted for querying, then only the page "The GNU C Library - Table of Contents" was found. It is apparent that in all these searches users have to spend much more time to browse and filter the information themselves to get the results they want. Otherwise they simply receive no results, if no pages in the collection contain all the keywords required.

From the results of the experiments we believe that any application domains that own rich hypertext composite information will benefit from the search method that makes use of such information in structured query and search.

6.6 Testing Method 4: Structured Query and Search with Link-Based Domain Models

This part of the experiments was set up to support the hypothesis that domain-specific structured search methods can generate better results than general search methods that do not understand the domain-specific information. By applying link-based domain models in formulating structured queries, Web users can get more specific results relevant to their information needs.

Applying the domain model as shown in Figure 6.1, an example query in the experiment was to find a staff member (either currently or formerly) who worked in the project DELITE Online, attended the conference ADL'2000 and presented a demo titled DELITE Online (as shown in Figure 5.22). The search hit was only one person, i.e., the author of this thesis. Such specific search results can usually not be retrieved directly from general Web search systems, which mostly do not understand domain-specific information at all. As Table 6.5 displays, in such systems, e.g. AltaVista and Google, users can only query with a combination of the relevant keywords even if their information need is very domain-specific. They can receive the pages that contain all those keywords (if such pages exist), but they then have to judge the relevance of the pages themselves by analyzing the semantic contents of the pages.

6.7 Testing Searches in other Available Systems and Result Comparison

In addition to our experimental searches in the test collection with the prototype system, we have also performed searches in other available systems. The query constructions for these searches were built up with the same query terms as used in the prototype system, but they were not the query constructions that could be supported only by the prototype system. Our goal was to seek more indications that the search methods proposed in this thesis would enable users to receive better search results which were more specific regarding their information needs, and users' efforts to interpret the results could be reduced.

Tables 6.3 to 6.6 display some search results we received from Altavista [Altavista] and Google [Google] compared with the results from our prototype system. All those experimental searches were restricted in the GMD site (for the request **4** shown in Table 6.6) and GMD Darmstadt site (for the requests **1 to 3** shown in Tables 6.3, 6.4 and 6.5) by specifying *url* (in AltaVista) or *site* (in Google)[6], as the pages in our test collection were only from these sites. This was to make the results from different systems to some extent comparable. We say "to some extent" here, because the collection used in different systems should still not be the same. One reason for this is that the Web sites are changing all the time – pages are added in or removed or changed. Another reason is that each system collects the pages from the sites at different time and with their own criteria.

For instance, as the test collection for our system was built some time before we tested searches with AltaVista and Google, some personal homepages appearing in the example result lists generated by these two systems were not contained in our test collection and thus could not be retrieved by our system, and vice versa also. There are also personal homepages whose contents have been largely modified after our test collection was built. Such pages (exactly their URLs) may be retrieved by AltaVista and Google as relevant with regard to some queries but not by our system. This is to say that it is not so valuable to calculate the accurate recall of the search results in the result comparison.

---

[6] Example query: *"GNU C" AND url:gmd.de* (in AltaVista); *"GNU C" AND site:gmd.de* (in Google).

We thus focused mainly on the number of search hits and relevant hits (shortened as *Rel.* in the tables) and further calculated the precision (shortened as *Prec.* in the tables) of the search results. In the calculation, only the pages that were directly relevant to the query requests were counted as relevant hits. For instance, for the query request **1** (Table 6.3) and **2** (Table 6.4), though a publication page with "information retrieval" appeared in the result list and users could infer from it that the authors were IR experts, this page was however not taken into account in calculating the precision.

The recall measurement has also drawn some attention. We noticed that for all query requests, when more specific queries, i.e. DELITE AND (staff OR member) AND "retrieval", etc. were formulated, the precision of the search results could be greatly increased (however it would be still lower than in HyperSM), while on the contrary the number of the relevant hits decreased (much lower recall). Users had to balance their queries with much effort in order to get all the specific search results that they could get in HyperSM with simple hyperstructure-based queries.

Distinctly, for query request 1 (Table 6.3) and 2 (Table 6.4), when simply querying with the terms that were used in searching in our prototype system, i.e. "retrieval", "information retrieval" and "multimedia", AltaVista and Google provided hundreds of search hits in which only tens were relevant with respect to the query requests – the precision was low. However, they provided in some cases more relevant hits than HyperSM, while at the same time they missed several quite relevant pages that appeared in the HyperSM result lists. In other words, the result lists generated by different systems are only partly overlapping between each other.

The reasons that AltaVista and Google produced more relevant hits were: (1) the collection differentiation problem – several hits in their lists were not contained in our test collection for HyperSM; (2) In their result lists there was more than one hits for one person, e.g. one's homepage and CV page, while in the HyperSM search results, for each person there was only one page, which was contained in the specified hypertext contexts used as the search boundaries; (3) Some persons claim their research interest and work not in their personal homepages but in the pages about their Curriculum Vitae. These Curriculum Vitae pages were not contained in the hypertext contexts used for the corresponding HyperSM search. This indicates to us

that the hypertext contexts used as search boundaries should be improved frequently when necessary, as the Web space changes and more and more queries have been implemented, in order that search results with a high quality can be produced.

The reasons that AltaVista and Google missed several quite relevant hits which appeared in the HyperSM search results were to a large degree mysterious for us, since we had no way to investigate their document collections and search algorithms. We only noticed that some pages contained in HyperSM result lists had been removed from the Web sites when we performed the test searches with these systems, while some pages were still there but did not appear in the results of these systems.

As for ranking of the search hits for query request 1, HyperSM produced a much better order (as shown in Table 6.2) than other systems. For instance, when the query was constructed as “DELITE AND (staff OR member) AND “retrieval”, Google produced 9 hits, only 3 of which were relevant and in the position of 2, 3, 9 in the result list.

Table 6.3 Comparison of experimental search results – *Query Request 1*

Query Request	HyperSM			Other Systems						
				Query Construction	AltaVista			Google		
	Method Note	Hits (Rel.)	Prec.		Hits	Rel.	Prec.	Hits	Rel.	Prec.
1. to find *DELITE experts* in the research field *retrieval*, by taking into accounts their publications and projects.	Searching "retrieval" within the boundary of the hypertext context "DELITE staff", Ranking results with specified profiles	10	1.0000	"retrieval"	424	14	0.0330	540	14	0.0259
				DELITE AND (staff OR member) AND "retrieval"	8	3	0.3750	9	3	0.3333

Note:

1. In **HyperSM:** the ranking of the 10 hits is as shown in Table 6.2.
2. For querying **"**DELITE AND (staff OR member) AND 'retrieval'",
   a. In **AltaVista**: the 3 relevant hits are the homepage or CV page of the DELITE staff member with the No. 1, 3, 10 in the HyperSM result list.
   b. In **Google**: The 3 relevant hits are the same as in AltaVista. They are in the positions 2, 3, 9 in the result list.

Table 6.3 Comparison of experimental search results – *query request 1*

Table 6.4 Comparison of experimental search results – *Query Request 2*

Query Request		HyperSM			Other Systems						
					Query Construction	AltaVista			Google		
		Method Note	Hits (Rel.)	Prec.		Hits	Rel.	Prec.	Hits	Rel.	Prec.
2. to find which *people* in the *IPSI* works in the field	*information retrieval*	Searching "information retrieval" within the boundary of the hypertext context "GMD-IPSI Staff"	6	1.0000	"information retrieval"	169	10	0.0592	254	13	0.0512
					IPSI AND (staff OR member) AND "information retrieval"	15	6	0.4000	16	6	0.3750
	Multimedia	Searching "multimedia" within the boundary of the hypertext context "GMD-IPSI Staff"	18	1.0000	"multimedia"	694	20	0.0288	840	25	0.0298
					IPSI AND (staff OR member) AND "multimedia"	40	14	0.3500	47	13	0.2766

Table 6.4 Comparison of experimental search results – *query request 2*

For the query request 3 (Table 6.5), AltaVista and Google could not provide direct answers to the complete query request. What they could provide were some low-precision results that met only a part of the request.

For the query request 4 (Table 6.6), AltaVista did not produce any results meeting the final query request. Users could only try to filter out the relevant pages from the search results for querying “GNU C” or “memory allocation”. Comparatively, Google produced much better results. Especially, when quite specific query constructions (i.e., *"GNU C" AND "dynamic allocation", or "GNU C" AND "memory allocation" AND "dynamic allocation"*) were submitted to search, Google figured out all the relevant pages that also appeared in the HyperSM results. The difference was that Google did not display the search hits in their original logical structural levels as HyperSM. As a result, users needed to browse from a search hit and judge the pages themselves before they got to the specific pages containing the information requested.

Table 6.5 Comparison of experimental search results – *Query Request 3*

Query Request		HyperSM			Other Systems						
					Query Construction	AltaVista			Google		
		Method Note	Hits (Rel.)	Prec.		Hits	Rel.	Prec.	Hits	Rel.	Prec.
3. to find the *staff member* who	*works in* the *project* "*DELITE Online*",	Structured query based on the link-based DO domain model	6	1.0000	"DELITE Online"	10	3	0.3000	13	5	0.3846
					(staff OR member) AND "DELITE Online"	3	2	0.6667	3	2	0.6667
	presents a *demo* titled "*DELITE Online*",		1	1.0000	"DELITE Online"	10	1	0.1000	13	1	0.0769
					(staff OR member) AND "DELITE Online"	3	0	0.0000	3	0	0.0000
	attends the *conference* "*ADL'2000*"		2	1.0000	"ADL'2000"	3	0	0.0000	2	0	0.0000
					(staff OR member) AND "ADL'2000"	0	0	0.0000	0	0	0.0000
	meets all the above requirements		1	1.0000	"DELITE Online" AND "ADL'2000"	0	0	0.0000	0	0	0.0000
					(staff OR member) AND "ADL'2000" AND "DELITE Online"	0	0	0.0000	0	0	0.0000

Table 6.5 Comparison of experimental search results – *query request 3*

Table 6.6 Comparison of experimental search results – *Query Request 4*

Query Request	HyperSM			Other Systems						
	Method Note	Hits (Rel.)	Prec.	Query Construction	AltaVista			Google		
					Hits	Rel.	Prec.	Hits	Rel.	Prec.
4. to find the information about *dynamic memory allocation* in *GNU C*	Structured query and search based on hypertext composites: Level 1: GNU C + Level 2: memory allocation + Level 3: dynamic allocation	4 match sets	1.0000	"GNU C"	111			339		
				"memory allocation"	51			211		
				"dynamic allocation"	6	0	0.0000	32	6	0.1875
				"GNU C" AND "memory allocation"	2	0	0.0000	17	5	0.2941
				"GNU C" AND "dynamic allocation"	1	0	0.0000	6	6	1.0000
				"GNU C" AND "memory allocation" AND "dynamic allocation"	0	0	0.0000	5	5	1.0000

Note:

The precisions corresponding to the query constructions "GNU C" and "memory allocation" were not calculated because they were not so important in the comparisons.

Table 6.6 Comparison of experimental search results – *query request 4*

6.8 Discussion

Generally, we saw the number of search hits for each test query was more (up to hundreds) or less (only several) in different systems. The larger the number of hits was, the more browsing and filtering work had to be done by users themselves until their specific information needs were satisfied, especially when what they expected were not single pages but a kind of structured results, i.e., interconnected pages.

It is usual that a global search system only collects/indexes a small part of the pages in a site based on its own criteria. For instance, by querying "url:darmstadt.gmd.de" in AltaVista to search for all the pages in the site "darmstadt.gmd.de", we received 9930 hits, which is much less than what we have collected from the site in our test collection (29574). This fact leads to another observation, i.e., search results for the same query terms are usually only partly overlapping in different systems (as indicated in the tables and the description above in this chapter). A serious problem related to this is, a global search system often misses the pages that can really satisfy users' specific information needs or can be good starting points for users to get to their target pages. In this scenario, we expect that all the important structural and semantic information in a site will be represented with the Web standards more explicitly and search methods that take advantage of this information will be provided. The methods presented in this thesis are examples to meet this expectation.

6.9 Summary

This chapter has presented the experimental evaluation studies carried out in this thesis. The results from the experiments have strongly supported the hypothesis that the hyperstructure-based search methods proposed in this thesis will help to improve the search quality in a system and save Web users' effort to get the search results that meet their information needs.

7 Conclusion

This thesis develops search methods that exploit the ever-enriched structural and semantic information in the hyperspace of the World Wide Web. The research consists of three closely related parts: the identification of the structural and semantic information available in the Web, the standard representation of such information, and the methods to apply such information for searching the Web and filtering and retrieving relevant information. The related work in the fields and the standardization efforts made by W3C [W3C] and IETF (Internet Engineering Task Force) [IETF] have indicated that the research on the topic of this thesis is of wide interest and meets the trend of Web development.

7.1 Contribution of the Thesis

The results of this thesis include four hyperstructure-based search methods, a prototype system, and the results from experimental qualitative evaluation studies. These have shown significantly that the hyperstructure-based search methods proposed in this thesis can improve the search quality on the Web, as the Web evolves from a poorly structured to a more structured, semantic-rich network. More concretely, by making use of hypertext composites and contexts, search results can be more specific with respect to users' information needs, and additionally, the users' efforts to interpret the search results can be reduced. Presenting structured search results based on hypertext composites as inter-linked nodes rather than separate nodes helps users understand the retrieved information better. By making use of semantic information in hyperstructures (e.g., types of links and nodes), better filters can be developed for selecting and ranking Web pages. These pages can be either intermediate information for further processing or final search results presented to users. By making use of domain models, domain-specific structure-based search methods can be developed, which may generate better results than general search methods that do not understand the domain-specific information.

7.2 Comparisons to Related Work

The hyperstructure-based approach taken in this thesis is an extension of the traditional structure-based search method, which mainly handles hierarchical structures (composed by non-linking mechanisms) in structured documents (e.g., XML). In addition to such hierarchical structures, the hyperstructure-based approach can also handle both hierarchical and non-hierarchical structures composed by linking mechanisms.

Compared to other link-based approaches that largely take into account the quantity of links in their search methods, this approach also makes use of the semantic information in links and link-based structures. It fits the trend of Web development with regard to capturing rich structural and semantic information and thereby capitalizing on the potential of new search methods. Though it asks for extra efforts from document providers to create additional information in their documents and represent the information with the new Web standards, the benefits that it can bring to the end users make these efforts worthwhile, especially for the users of digital libraries and knowledge management systems. This is similar to the work of describing and organizing document collections through classification and indexing in conventional libraries so that readers can utilize the value of the collections to the highest degree.

7.3 Future Work

As the new Web standards become more widely adopted and more and more structural and semantic information represented in the standard way is provided on the Web, an extended evaluation of the methods and the system will be performed in our future activities. Such an evaluation will cover not only the quality of search results with respect to precision and recall but also the technical system performance and scalability with respect to indexing and searching. It will also measure the storage requirements in the system and judge the friendliness of the user interfaces. Furthermore, it will be studied how the proposed search methods can be effectively used in practice as the Web grows.

Following the evaluation, improvements and further extensions to the methods and the system can be made. These will include a more intuitive visual interface for formulating queries taking advantage of structural and semantic information in hyperstructures, and for presenting search results (especially structured search results) in a way that users can understand them better and perceive hypertext context information more efficiently. Other alternative query execution plans will also be considered.

The hyperstructure-based search methods proposed in this thesis are only several examples that show how the structural and semantic information that are representable with the new Web standards can be applied efficiently for searching the Web and filtering and retrieving relevant information. More such methods can be developed in order to explore the full value of this kind of information in the future semantic Web. Possible contribution of other high-level meta information, e.g. MPEG or MPEG 7 type of information, will also be explored in our future activities.

References

[Afrati&Koutras 1990] Afrati, F., and Koutras, C. D., "A hypertext model supporting query mechanism," in *Hypertext: Concepts, Systems and Applications*, Proc. the European Conference on Hypertext, INRIA, France, November 1990, A. Rizk, N. Streitz, and J. Andre, Eds., Cambridge University Press, pp. 52-66.

[Agosti 1996] Agosti, M., "An overview of hypertext," in: Agosti, M. and Smeaton, A. (ed.), Information Retrieval and Hypertext, Chapter 2, Kluwer Academic Publishers, 1996.

[Agosti&Smeaton 1996] Agosti, M. and Smeaton A. (ed.), "Information retrieval and hypertext," Kluwer Academic Publishers, 1996.

[Allan 1996] Allan, J., "Automatic hypertext link typing," Proc. 7th ACM Conference on Hypertext, pp. 42-52, 1996.

[Allan 1997] Allan, J., "Building hypertext using information retrieval," in Information Processing and Management 33(2), 145-159, 1997.

[Amann&Scholl 1992] Amann B., and Scholl, M., "Gram: A graph data model and query language," Proc. the European Conference on Hypertext (ECHT'92), Milan, Italy, Nov. 30-Dec.4, pp. 201-211.

[Amitay et al. 2003] Amitay, E., Carmel, D., Darlow, A., Lempel, R., Soffer A., "The connectivity sonar: detecting site functionality by structural patters," Hypertext'03, August 26-30, 2003, Nottingham, United Kingdom.

[Andrews 1996] Andrews, K., "Applying hypermedia research to the World Wide Web," Position Paper for the Workshop on Hypermedia Research and the World-Wide Web, ACM Hypertext'96, Washington, DC, March 1996.

[Andrews et al. 1995] Andrews, K., Kappe, F., and Maurer, H., "Serving information to the Web with Hyper-G," Proc. 3rd International WWW Conference, Darmstadt, Germany, April 1995.

[Arocena et al. 1997] Arocena, G. O., Mendelzon, A. O., Mihaila, G. A., "Applications of a Web quary language," Proc. 6th International World Wide Web Conference, 1997.

[Bechhofer et al. 2001] Bechhofer, S., Horrocks, I., Goble C., Stevens, R., "OilEd: a reasonable ontology editor for the Semantic Web," Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396--408. 2001.

[Beeri&Kornatzky 1990] Beeri, C., and Kornatzky, Y., "A logical query language for hypertext systems," in *Hypertext: Concepts, Systems and Applications*, Proc. the European Conference on Hypertext, INRIA, France, November 1990, A. Rizk, N. Streitz, and J. Andre, Eds., Cambridge University Press pp. 67-80.

[Bernstein 1990] Bernstein, M., "Link apprentice," Proc. ECHT '90, INRIA, France, Cambridge Series on Electronic Publishing, pp. 212-223.

[Berners-Lee 1998] Berners-Lee, T., "What the Semantic Web can represent," at: http://www.w3.org/DesignIssues/RDFnot.html.

[Bharat et al. 1998] Bharat, k. and Henzinger, M. R., "Improved algorithms for topic distillation in a hyperlinked environment," Proc. 21st Annual International ACM SIGIR Conference (SIGIR'98), pp. 104 -111, 1998.

[Botafogo et al. 1992] Botafogo, R., Rivlin, E., Shneiderman, B., "Structural analysis of hypertext: Identifying hierarchies and useful metrics," ACM Transactions on Information Systems, 10(1992), pp. 142-180.

[Boy 1991] Boy, G., "Indexing Hypertext Documents in Context," Proc. Hypertext '91. San Antonio, Texas. ACM Press. pp. 51-61.

[Boyan et al. 1996] Boyan, J., Freitag, D. and Joachims, T., "A machine learning architecture for optimizing Web search engines," in the AAAI-96 Workshop on Internet-based Information Systems.

[Bozsak et al. 2002] Bozsak, E., Ehrig, M., Handschub, S., Hotho, et al. "KAON – towards a large scale Semantic Web," in: Bauknecht, K., Min Tjoa, A., Quirchmayr, G. (Eds.): Proc. 3rd International Conference on E-Commerce and Web Technologies (EC-Web 2002), 2002, pp. 304 – 313.

[Brin&Page 1998] Brin, S. and Page, L., "The anatomy of a large-scale hypertextual Web-search engine," Proc. 7th International World Wide Web Conference, 1998.

[Broekstra et al. 2001] Broekstra, J., Klein, M., Decker, S., Fensel, D., Harmelen, F., and Horrocks, I., "Enabling knowledge representation on the Web by extending RDF schema," Proc. 10th International World Wide Web Conference, 2001.

[Brusilovsky 1996] Brusilovsky, P., "Adaptive hypermedia: an attempt to analyse and generalize," in: Multimedia, Hypermedia, and Virtual Reality, pp. 288–304, Springer-Verlag, Berlin 1996.

[Brusilovsky&Schwarz 1997] Brusilovsky, P. and Schwarz, E., "Concept-based navigation in educational hypermedia and its implementation on WWW," Proc. of ED-MEDIA/ED-TELECOM'97 - World Conference on Educational Multimedia/Hypermedia and World Conference on Educational Telecommunications, pp. 112–117, Calgary, Canada 1997.

[Bush 1945] Bush, V., "As we may think," the Atlantic Monthly, July 1945.

[Campbell&Goodman 1988] Campbell, B., Goodman, J. M., "HAM: A general purpose hypertext abstract machine," Communications of the ACM, 31(7), July 1988, pp. 856-861.

[Carr et al. 1996] Carr, L., Hill, G., Roure, D. D., Hall, W., and Davis, H. "Open information services," Proc. 5th International World Wide Web Conference, 1996.

[Carriere&Kazman 1997] Carriere, J. and Kazman, R., "WebQuery: Searching and visualizing the Web through connectivity," Proc. 6th International World Wide Web Conference, 1997.

[Casanova&Tucherman 1991] Casanova, M. A. and Tucherman, L., "The nested context model for hyperdocuments," Proc. of Hypertext'91, pp. 193-201.

[Chakrabarti et al. 1998] Chakrabarti, S., Dom, B., Gibson, D., Kleinberg, J., Raghavan, P., Rajagopalan, S., "Automatic resource compilation by analyzing hyperlink structure and associated text," Proc. 7th International World Wide Web Conference, 1998.

[Chakrabarti et al. 2002] Chakrabarti, S., Joshi, M. M., Punera, K., and Pennock, D. M., "The structure of broad topics on the Web," Proc. 11th International World Wide Web Conference (WWW 2002), 2002.

[Chen et al. 2003] Chen, Z., Liu, S., Liu, W., Pu, G., and Ma, W., "Building a Web thesaurus from Web link structure," SIGIR'03, July 28 - August 1, 2003, Toronto, Canada, pp. 48-55.

[Chignell&Nordhausen 1991] Chignell, M.H., Nordhausen, B. Valdez, F. and Waterworth, J.A., "The HEFTI model of text to hypertext conversion, " *Hypermedia*, 3 (3), pp. 187-205.

[Chris 1995] Chris, J. Date, "An introduction to database systems. The systems programming series," Addison-Wesley, Reading, Massachusetts, sixth edition, 1995.

[Christophides&Rizk 1994] Christophides, V. and Rizk, A., "Querying structured documents with hypertext links using OODBMS," Proc. ECHT '94, Edinburgh, UK. ACM Press. pp. 186-197

[Clitherow et al. 1989] Clitherow, P., Reicken, D., and Muller, M., "VISAR: a system for inference and navigation in Hypertext," Proc. Hypertext '89. Pittsburgh, PA. ACM Press. pp. 293-304.

[Conklin 1987] Conklin, J., "Hypertext: An introduction and survey," Computer, September 1987, vol. 20, no. 9, IEEE Publications, pp. 17-41.

[Consens&Mendelzon 1989] Consens, M. P., and Mendelzon, A. O., "Expressing structural hypertext queries in GraphLog," Proc. 2nd ACM Conference on Hypertext (Hypertext'89), Pittsburgh, PA, November 1989, pp. 269-292.

[Croft&Turtle 1989] Croft, W.B. and Turtle, H., "A retrieval model for incorporating hypertext links," Proc. Hypertext '89, Pittsburgh, Penn., ACM Press, pp. 213-224.

[Crouch et al. 1989] Crouch, D.B., Crouch, C.J., and Andreas, G., "The user of cluster hierarchies in hypertext information retrieval," Proc. Hypertext '89, Pittsburgh, Penn., ACM Press, pp. 225-237.

[Davis et al. 1992] Davis, H.C., Hall, W., Heath, I., Hill, G. & Wilkins, R., " Towards an integrated information environment with open hypermedia systems," in: D. Lucarella, J. Nanard, M. Nanard, P. Paolini. eds. Proc. ACM Conference on Hypertext, ECHT '92, Milano, ACM.

[Dean&Henzinger 1999] Dean, J. and Henzinger, M., "Finding related pages in the World Wide Web," Proc. 8th International World Wide Web Conference, pp. 389-401, 1999.

[Decker et al. 1998] Decker, S., Brickley, D., Saarela, J. and Angele, J., "A query and inference service for RDF," In Proc. of the W3C Query Language Workshop (QL-98), Boston, MA, December 3-4, 1998.

[Decker et al. 1999] Decker, S., Erdmann, M., Fensel, D. and Studer, R., "Ontobroker: ontology based access to distributed and semi-structured information," in: R. Meersman et

al., editors, Database Semantics: Semantic Issues in Multimedia Systems, pages 351 – 369. Kluwer Academic Publisher, 1999.

[Delisle&Schwartz 1986] Delisle, N., Schwartz, M., "Neptune: A hypertext system for CAD applications," ACM SIGMOD'86 Conference, Washington DC, May 1986, pp 132-142.

[DeRose 1989] DeRose, S. J., "Expanding the notion of links," Proc. of Hypertext'89, pp. 249-257.

[ECDL 2001 Panel] "Digital Library Programs: Current Status and Future Plans, " Proc. of the 5th European Conference on Research and Advanced Technology for Digital Libraries (ECDL '01), September 4-9, 2001, Darmstadt, Germany. Berlin: Springer, 2001.

[Efe et al. 2000] Efe, k., Raghavan, V., Chu, C. H., Broadwater, A. L., Bolelli, L., and Ertekin, S., "The shape of the Web and its implications for searching the Web," Proc. of SSGRR 2000, July 31 – August 06, 2000, L'Aquila, Italy.

[Egan et al. 1989] Egan, D.E., Remde, J.R., Gomez, J.M., Landauer, T.K., Eberhardt, J. and Lochbaum, C.C., "Formative Design-Evaluation of SuperBook," ACM Transactions on Information Systems, 7 (1) pp. 30-57.

[Eiron&McCurley 2003] Eiron, N., and McCurley, K., "Untangling compound documents on the Web," Hypertext'03, August 26-30, 2003, Nottingham, United Kingdom.

[Farquhar et al. 1997] Farquhar, A., Fikes, R., and Rice, J., "The Ontolingua Server: A tool for collaborative ontology construction," International Journal of Human-Computer Studies, 46(6), 1997, pp. 707-727.

[Fischer&Ostwald 2001] Fischer, G., and Ostwald, J., "Knowledge management: problems, promises, realities, and challenges," IEEE Intelligent Systems & their Applications, 16(1), January/February 2001, pp. 60-72. Knowledge Management – Special Issue.

[Fountain et al. 1990] Fountain, A.M., Hall, W., Heath, I. & Davis, H.C., "MICROCOSM: an open model for hypermedia with dynamic linking," in: A. Rizk, N. Streitz and J. Andre eds. Hypertext: Concepts, Systems and Applications. Proc. European Conference on Hypertext, INRIA, France. Cambridge University Press, 1990.

[Frei&Stieger 1992] Frei, H. P. and Stieger, D., "Making use of hypertext links when retrieving information", Proc. 3rd ACM Conference on Hypertext, pp. 102-111, 1992.

[Frei&Stieger 1995] Frei, H. P. and Stieger, D., “The use of semantic link in hypertext information retrieval,” Information Processing and Management, Vol. 31, No. 1, 1995, pp. 1-13.

[Frisse 1988] Frisse, M. E., “Searching for information in a hypertext medical handbook,” Communications of the ACM, 31(7), pp. 880-886, 1988.

[Frisse&Cousins 1989] Frisse, M. E. and Cousins, S.B., “Information retrieval from hypertext: update on the dynamic medial handbook project,” Proc. Hypertext '89, ACM Press. pp. 199-212.

[Fuhr 2000] Fuhr, N., "Probabilistic datalog: Implementing logical information retrieval for advanced applications," Journal of the American Society for Information Science (JASIS), Volume 51, Number 2, 2000, pp. 95-110.

[Furner et al. 1996] Furner, J., Ellis, D., and Willett, P., “The representation and comparison of hypertext structures using graphics, ” in: Agosti, M. and Smeaton, A. (ed.), Information Retrieval and Hypertext, Chapter 4, Kluwer Academic Publishers, 1996.

[Garfield 1972] Garfield, E., “Citation analysis as a tool in journal evaluation,” Science, 178(1972), pp. 471-479.

[Garfield 1994] Garfield, E., “The impact factor,” Current Contents, June 20, 1994.

[Gates 1996] GATES, B., Keynote lecture presented at comdex'96, Las Vegas, November 19, 1996. Available via: http://www.microsoft.com/billgates/speeches/comdex96/keynote.asp.

[Green 1998] Green, S. J., "Automated link generation: can we do better than term repetition?" in Proc. 7th International World Wide Web Conference, Brisbane, Australia, 75-84, 1998.

[Gronbak&Trigg 1996] Gronbak, K. and Trigg, R., “Towards a dexter-based model for open hypermedia: Unifying embedded references and link objects,” Proc. ACM Hypertext’96, pp.16-20, 1996.

[Gronbak et al. 1997] Gronbak, K., Bouvin, N. O., Sloth, L., “Designing Dexter-based hypermedia services for the World Wide Web,” Proceedings of Hypertext’97, pp. 146-156, Southampton, UK, April 6-11, 1997.

[Gruber 1993] Gruber, T. R., “A translation approach to portable ontologies,” Knowledge Acquisition, 5(2): 199-220, 1993.

[Guinan&Smeaton 1992] Guinan, C. and Smeaton, A.F., "Information retrieval from hypertext using dynamically planned guided tours," Proc. ECHT '92, Milan, Italy, ACM Press, pp. 122-130.

[Haan et al. 1992] Haan, B. J., Kahn, P., Riley, V. A., Coombs, J. H., and Meyrowitz, N. K., "IRIS hypermedia services," in Communications of the ACM (CACM), 35(1), pp. 36-51, January 1992.

[Halasz 1991] Halasz, F., "Seven issues," Revisited (keynode address), Proc. 3rd ACM Conference on Hypertext, San Antonio, TX, 1991.

[Halasz 1988] Halasz, F., "Reflections on Notecards : seven issues for the next generation of hypermedia systems," Communication of the ACM, 31(7), pp.836-852, 1988.

[Halasz&Schwartz 1990] Halasz, F. and Schwartz, M., "The Dexter Hypertext Reference Model," Proc. Hypertext Standardization Workshop, pp. 95-134, U.S. Government Printing Office, Washington, DC, 1990.

[Halasz&Schwartz 1994] Halasz, F. and Schwartz, M., "The Dexter Hypertext Reference Model," Communications of the ACM, 37(2), pp.30-39, February 1994.

[Handschuh&Staab 2002] Handschuh, S. and Staab, S., "Authoring and annotation of Web pages in CREAM," Proc. 11th International World Wide Web Conference, Honolulu, Hawaii, May 2002.

[Haveliwala 2002] Haveliwala, T., "Topic-sensitive PageRank," Proc. 11th International World Wide Web Conference, Honolulu, Hawaii, May 2002.

[Hawking et al. 1999] Hawking, D., Craswell, N., Thistlewaite, P. and Harman, D., "Results and challenges in Web search evaluation," Proc. 8th International World Wide Web Conference, 1999.

[Hawking et al. 2000] Hawking D., Craswell, N., Bailey P. and Griffiths K., "Measuring search engine quality," Information Retrieval, Vol. 4 No. 1., 2000.

[Hawking et al. 2001] Hawking, D., Craswell, N. and Griffiths, K., "Which Search Engine is best at finding Online Services?" Proc. 10th International World Wide Web Conference Poster, 2001.

[Hearst 1994] Hearst, M.A., "Multi-paragraph segmentation of expository text," Proc. 32nd Meeting of the Association for Computational Linguistics, Los Cruces, NM.

[Heflin&Hendler 2000] Heflin, J. and Hendler, J., "Searching the Web with SHOE," AAAI-2000 Workshop on AI for Web Search, 2000.

[Henzinger 2000] Henzinger, M. R., "Link analysis in web information retrieval," IEEE Data Engineering Bulletin, 23(3): 3-8, September 2000.

[Horrocks 1998] Horrocks, I., "Using an expressive Description Logic: fact or fiction?" in Proc. 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98). Trento, Italy, June 2-5, 1998, pages 636-649. Morgan Kaufmann, 1998.

[Huang et al. 2000] Huang, L., Hemmje, M., Neuhold, E.J., "ADMIRE: An adaptive data model for meta search engines", Proc. 9th International World Wide Web Conference (WWW9). Amsterdam, The Netherlands, May 15-19, 2000

[Huck et al. 1998] Huck, G., Fankhauser, P., Aberer, K., Neuhold, E., "Jedi: Extracting and synthesizing information from the Web," Proc. 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS'98), New York City, August 1998, pp. 32-43.

[ISO 8613] ISO 8613. Information Processing -- text and office systems -- office document architecture (ODA) and interchange format, 1998. International Organisation for Standardization, Geneva.

[ISO 8879:1986] ISO 8879:1986. Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML). International Organization for Standardization, Geneva.

[Iwazume et al. 1996] Iwazume, M., Shirakami, K., Hatadani, K., "IICA: an ontology-based Internet navigation system," in the AAAI-96 Workshop on Internet-based Information Systems, 1996.

[Kahan et al. 2001] Kahan, J., Koivunen, M., Hommeaux, E. P., and Swick, R., "Annotea: An open RDF infrastructure for shared Web annotations," Proc. 10th International World Wide Web Conference, Hong Kong, 2001.

[Kaindl et al. 1998] Kaindl, H., Kramer, S, and Afonso, L. M., "Combining structure search and content search for the World-Wide Web," Proc. 9th ACM Conf. Hypertext and Hypermedia, Pittsburgh PA USA, pp. 217-224, 1998.

[Kleinberg 1998] Kleinberg, J., "Authoritative sources in a hyperlinked environment," Proc. ACM-SIAM Symposium on Discrete Algorithms, 1998. Also appears as IBM Research Report RJ 10076(91892), May 1997.

[LaMacchia 1996] LaMacchia, B. A., "Internet fish," Ph.D. thesis, MIT Department of Electrical Engineering and Computer Science, 1996.

[Larson 1996] Larson, R., "Bibliometrics of the World Wide Web: An exploratory analysis of the intellectual structure of cyberspace," Annual Meeting of the American Society of Information Science, 1996.

[Leary 1996] Leary, D., "The relationship between relevance and reliability in Internet-based information and retrieval systems", in the AAAI-96 Workshop on Internet-based Information Systems, 1996.

[Leary 2001] Leary, D., "How knowledge reuse informs effective system design and implementation," IEEE Intelligent Systems & their Applications, 16(1), January/February 2001. Knowledge Management – Special Issue.

[Leighton&Srivastava 1999] Leighton, H. V. and Srivastava, J., "First 20 precision among world wide web search services (search engines)," Journal of the American Society for Information Science, 50 (10): 882-889, 1999.

[Li et al. 2002] Li, L., Shang, Y., and Zhang, W., "Improvement of HITS-based algorithms on Web documents," Proc. 11th International World Wide Web Conference, pp. 527-535, 2002.

[Li-W et al. 2002] Li, W., Candan, S., Vu, Q, and Agrawal, D., "Query relaxation by structure and semantics for retrieval of logical Web documents," IEEE Transaction of Knowledge and Data Engineering, Vol. 14, No. 4, July/August 2002.

[Lucarella et al. 1993] Lucarella, D., Parisotto, S., and Zanzi, A., "MORE: multimedia object retrieval environment," Proc. 5th ACM Conference on Hypertext (Hypertext'93), Seattle, WA, November 1993, pp. 39-50.

[Manber et al. 1997] Manber, M., Smith, M., and Gopal, G., "WebGlimpse -- Combining browsing and searching," *Usenix 1997 Annual Technical Conference*, Anaheim, CA (January 1997), pp. 195-206.

[Marchionini 1995] Marchionini, G., "Information seeking in electronic environments," Cambridge University Press, 1995.

[Marchiori 1997] Marchiori, M., "The quest for correct information on the Web: Hyper search engines," Proc. 6th International World Wide Web Conference, 1997.

[Miller&Bharat 1998] Miller, R., Bharat, K., "SPHINX: A framework for creating personal, site-specific Web crawlers," Proc. 7th International World Wide Web Conference, Brisbane Australia, April 1998.

[Minker 1977] Minker, J., "Information storage and retrieval – a survey and functional description," SIGIR Forum. Association for Computing Machinery, Vol. 12, No. 2, Fall 1977, pp. 1-108.

[Mizuuchi&Tajima 1999] Mazuuchi, Y., and Tajima, K., "Finding context paths for Web pages," Hypertext'99, Darmstadt, Germany, 1999, pp. 13-22.

[Mohageg 1992] Mohageg, M. F., "The influence of hypertext linking structures on the efficiency of information retrieval," Human Factors, 1992, 34(30), pp. 351-367.

[Nejdl et al. 2002] Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmer, M., and Risch, T., "EDUTELLA: A P2P networking infrastructure based on RDF," Proc. 11th International World Wide Web Conference, 2002.

[Nelson 1965] Nelson, T., "A file structure for the complex, the Changing and the Indeterminate," ACM 20th National Conference, 1965.

[Page et al. 1998] Page, L., Brin, S., Motwani, R., and Winograd, T., "The PageRank citation ranking: Bringing order to the Web," Technical report, Stanford University, Stanford, CA., 1998.

[Picard 1998] Picard, J., "Modeling and combining evidence provided by document relationships using probabilistic argumentation systems," Proc. 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998), August 24-28 1998, Melbourne, Australia. ACM 1998, pp. 182-189.

[Pirolli et al. 1996] Pirolli, P., Pitkow, J., Rao, R., "Silk from a sow's ear: Extracting usable structures from the Web," Proc. ACM SIGCHI Conference on Human Factors in Computing, 1996.

[PITAC 2001] "Digital Libraries: Universal Access to Human Knowledge", President's Information Technology Advisory Committee (PITAC), Panel on Digital Libraries, U.S.A., February 2001.

[Quillian 1968] Quillian, M., "Semantic memory," In M. Minsky, editor, Semantic Information Processing, pp. 227--270. MIT Press, Cambridge, MA, 1968.

[Rada 1990] Rada, R., "Hypertext writing and document reuse: the role of a semantic net," Electronic Publishing 3(3), pp. 125-140 , 1990.

[Rafiei et al. 2000] Rafiei, D. And Mendelzon, O., "What is this page known for? Computing Web page reputations," Proc. 9th International World Wide Web Conference, pp. 823-835, 2000.

[Richardson&Domingos 2002] Richardson, M. and Domingos, P., "The intelligent surfer: probabilistic combination of link and content information in PageRank," Advances in Neutral Information Processing Systems 14, pp. 1441-1448, MIT Press, 2002.

[Rijsbergen 1979] Rijsbergen, C. J. Van, "Information retrieval," Second Edition Butterworths, London, 1979.

[Rivlin et al. 1994] Rivlin, E., Botafogo, R. and Shneiderman, B., "Navigating in hyperspace: designing a structure-based toolbox," Communications of the ACM, 37(2), pp. 87-96, 1994.

[Robertson et al. 1994] Robertson, J. Merkus, E., and Ginige, A., "The Hypermedia Authoring Research Toolkit (HART)," Proc. ECHT '94, Edinburgh, UK. ACM Press. pp. 177-185.

[Robertson&Jones 1976] Robertson, S.E. and Sparck Jones, K., "Relevance weighting of search terms," Journal of the American Society for Information Science, 27(3): 129–146.

[Rölleke&Blömer 1997] Rölleke, T., Blömer, M., "Probabilistic logical information retrieval for content, hypertext, and database querying," Proc. Hypertext - Information Retrieval - Multimedia '97 (HIM'97), Sept. 29 – Oct. 2, 1997, Dortmund, Germany.

[Rousseau&Hooydonk 1996] Rousseau, R. and Van Hooydonk, G., "Journal production and journal impact factors," Journal of American Society of Information Science, 47(1996), pp. 775-780.

[Rousseau 1997] Rousseau, R., "Sitations: an exploratory study," Cybermetrics, 1(1), 1997.

[Salton 1971] Salton, G., "The SMART retrieval system: experiments in automatic document processing," Prentice-Hall, Englewood Cliffs, 1971.

[Salton 1989] Salton, G., "Automatic text processing – the transformation, analysis and retrieval of information by computer," Addison-Wesley, Reading, MA, 1989.

[Salton&Allan 1993] Salton, G. and Allan, J., "Selective text utilization and text traversal," Proc. Hypertext'93 pages 131-144.

[Salton&Mill 1983] Salton, G. and Gill, M., "Introduction to modern information retrieval," Mc.Graw Hill, New York, 1983, 448 p.

[Salton et al. 1994] Salton, G., Allan, J., and Buckley, C., "Automatic structuring and retrieval of large text files," Communications of the ACM, 37(2), pp. 97-108, 1994.

[Salton et al. 1996] Salton, G., Allan, J., Buckley, C., and Singhal A., "Automatic analysis, theme generation, and summarization of machine-readable texts," in Agosti, M. and Smeaton, A. (ed.), Information Retrieval and Hypertext, Chapter 3, Kluwer Academic Publishers, 1996.

[Savoy 1993] Savoy, J., "Searching information in hypertext systems using multiple sources of evidence," International Journal of Man–Machine Studies 6 (38), pp. 1017-1030.

[Savoy 1996] Savoy, J., "Citation schemes in hypertext information retrieval," in Agosti, M. and Smeaton, A. (ed.), Information Retrieval and Hypertext, Chapter 5, Kluwer Academic Publishers, 1996.

[Schwartz&Delisle 1987] Schwartz, M., Delisle, N., "Contexts – A partitioning concept for hypertext," ACM Transactions on Office Information Systems, 5(2), April 1987, pp. 168-186.

[Schwarz et al. 1996] Schwarz, E., Brusilovsky, P., and Weber, G., "World-wide intelligent textbooks," Proc. ED-MEDIA'96 – World Conference on Educational Multimedia and Hypermedia, Boston, MA.

[Shum 1996] Shum, S. B., "The missing link: hypermedia usability research & the Web," Interfaces, British HCI Group Magazine, Summer, 1996. Reprinted in ACM SIGCHI Bulletin 28 (4), 68-75.

[Smeaton 1996] Smeaton, A. F., "An overview of information retrieval," in Agosti, M. and Smeaton, A. (ed.), Information Retrieval and Hypertext, Chapter 1, Kluwer Academic Publishers, 1996.

[Smith&Weiss 1988] Smith, J. and Weiss, S., "An overview of hypertext," CACM, July 1988.

[Spertus 1997] Spertus, E., "ParaSite: Mining structural information on the Web," Proc. 6th International World Wide Web Conference, 1997.

[Staab et al. 2001] Staab, S., Studer., R., Schnurr, H. and Sure, Y., "Knowledge processes and ontologies," IEEE Intelligent Systems & their Applications, 16(1), January/February 2001, pp. 26-34. Knowledge Management – Special Issue.

[Sumner et al. 1998] Sumner, R. G., Yang, K., and Dempsey, B. J., "An interactive WWW search engine for user-defined collections," Digital Libraries 98, Pittsburgh PA, USA.

[Tajima et al. 1999] Tajima, K., Hatano, K., Matsukura, T., Sano, R., and Tanaka, K., "Discovery and retrieval of logical information units in Web", Proc. of Workshop on Organizing Web Space (WOWS'99), in conjunction with ACM DL'99 Berkeley CA, USA, Aug. 1999, pp. 13-23.

[Trigg 1983] Trigg, R., "A network-based approach to text handling for the online scientific community," Ph.D. thesis, Dept. of Computer Science, Univ. of Maryland, November 1983.

[Trigg 1996] Trigg, R., "Hypermedia as integration: Recollections, reflections and exhortations," In: Keynote Address in Hypertext'96 Conference. Xerox Palo Alto Research Center.

[URI1 1994] IETF RFC 1738 IETF (Internet Engineering Task Force). RFC 1738: Uniform Resource Locators (URL), ed. T. Berners-Lee, L. Masinter, and M. McCahill. 1994.

[URI2 1995] IETF RFC 1808 IETF (Internet Engineering Task Force). RFC 1808: Relative Uniform Resource Locators, ed. R. Fielding, 1995.

[Weiss et al. 1996] Weiss, R., Veles, B., Sheldon, M., Nemprempre, C., Szilagyi, P., Gifford, D. K., "HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering," Proc. 7th ACM Conference of Hypertext, 1996.

[White&McGain 1989] White, H. D. and McGain, K. W., "Bibliometrics," Annual Review of Information Science and Technology, Elsevier, 1989. pp. 119-186.

[Wilkinson&Fuller 1996] Wilkinson, R. and Fuller, M., "Integration of information retrieval and hypertext via structure," in Agosti, M. and Smeaton, A. (ed.), Information Retrieval and Hypertext, chapter 11, Kluwer Academic Publishers, 1996.

[Woods 1985] Woods, W. A., "What's in a link: foundations for semantic networks," in Brachman, R. J. and Levesque H. (ed.), Readings in Knowledge Representation, Morgan Kaufmann, 1985.

[Xue et al. 2003] Xue, G., Zeng, H., Chen, Z., Ma, W., Zhang, H., and Lu, C., "Implicit link analysis for small Web search," SIGIR'03, Toronto, Canada, 2003, pp. 56-63.

URL:

[AltaVista] AltaVista: http://www.altavista.com/

[Clever] Clever: http://www.almaden.ibm.com/cs/k53/clever.html

[DESIRE] DESIRE: http://www.desire.org/

[Dublin Core] Dublin Core: http://purl.oclc.org/dc/

[Glimpse] Glimpse: http://glimpse.cs.arizona.edu/

[Google] Google: http://google.stanford.edu/

[HTML] Hypertext Markup Language Home Page. At: http://www.w3.org/MarkUp/

[HTML 3.2] HTML 3.2 Reference Specification (W3C Recommendation *14-Jan-1997*). At: http://www.w3.org/TR/REC-html32.html

[HTML 4.0] HTML 4.0 Specification (W3C Recommendation, revised on 24-Apr-1998). At: http://www.w3.org/TR/1998/REC-html40-19980424/

[HTML 4.01] HTML 4.01 Specification (W3C Recommendation 24 December 1999). At: http://www.w3.org/TR/html4/

[IETF] The Internet Engineering Task Force: http://www.ietf.org/

[InfoSeek] InfoSeek: http://www.go.com/

[Lycos] Lycos: http://a2z.lycos.com/

[OIL] OIL: http://www.ontoknowledge.org/oil/

[OntoAgents] OntoAgents: http://www-db.stanford.edu/OntoAgents/

[Ontobroker] Ontobroker: http://ontobroker.aifb.uni-karlsruhe.de/

[Ontoserver] Ontoserver: http://ontoserver.aifb.uni-karlsruhe.de/

[RDF] RDF: http://www.w3.org/RDF/

[RDF Schema] RDF Vocabulary Description Language 1.0: RDF Schema (W3C Working Draft 30 April 2002). At: http://www.w3.org/TR/rdf-schema/

[RDF Spec] Resource Description Framework (RDF) Model and Syntax Specification (W3C Recommendation 22 February 1999). At: http://www.w3.org/TR/REC-rdf-syntax/

[Semantic Web] SemanticWeb: http://www.semanticweb.org

[SHOE] SHOE: http://www.cs.umd.edu/projects/plus/SHOE/

[W3C] W3C: http://www.w3c.org/

[W3C Terms] Hypertext Terms. At: http://www.w3.org/Terms.html

[WebGlimpse] WebGlimpse: http://donkey.CS.Arizona.EDU/webglimpse/

[XLink] XML Linking Language (XLink) Version 1.0 (W3C Recommendation 27 June 2001). At: http://web3.w3.org/TR/xlink/

[XML] XML: http://www.w3.org/XML

[XML 1.0] Extensible Markup Language (XML) 1.0 (W3C Recommendation 10-February-1998). At: http://www.w3.org/TR/1998/REC-xml-19980210

[XML Path] XML Path Language (XPath) 2.0 (W3C Working Draft 16 August 2002). At: http://www.w3.org/TR/xpath20

[XML Query] XML Query: http://www.w3.org/XML/Query

[XML Query Requirements] XML Query Requirements (W3C Working Draft 15 February 2001). At: http://www.w3.org/TR/xmlquery-req

[XML Query Use Cases] XML Query Use Cases (W3C Working Draft 16 Aug 2002). At: http://www.w3.org/TR/xmlquery-use-cases

[XML Schemas: Structures] XML Schema Part 2: Structures *(W3C Recommendation 2 May 2001).* At: http://www.w3.org/TR/xmlschema-1/

[XML Schemas: Datatypes] XML Schema Part 2: Datatypes *(W3C Recommendation 2 May 2001).* At: http://www.w3.org/TR/xmlschema-2/

[XPointer] XML Pointer Language (XPointer) Version 1.0 (W3C Last Call Working Draft 8 January 2001). At: http://www.w3.org/TR/WD-xptr

[XSL] XSL: http://www.w3.org/Style/XSL/

[XSLT] XSL Transformations (XSLT) Version 1.0 (W3C Recommendation 16 November 1999). At: http://www.w3.org/TR/xslt

[Yahoo] Yahoo: http://www.yahoo.com/

Appendix A: List of Figures and Tables

Appendix B: Selected Publications

Zhanzi Qiu, Matthias Hemmje, Erich J. Neuhold, “Using Link Types in Web Page Ranking and Filtering”, in Proc. 2nd International Conference on Web Information Systems Engineering (WISE 2001), December 3-6, Kyoto, Japan, (pp. 311-320). Los Alamitos, CA, IEEE Computer Society.

Zhanzi Qiu, Matthias Hemmje, Erich J. Neuhold, “Using Hypertext Composites in Structured Query and Search”, In: Bauknecht, Madria & Pernul (Eds.), 2nd International Conference on Electronic Commerce and Web Technologies (EC-Web 2001), (pp. 326-336). Berlin [etc.]: Springer, 2001 (Lecture notes in computer science; 2115).

Zhanzi Qiu, Matthias Hemmje, Erich J. Neuhold, “Using Link-Based Domain Models in Web Searching”, in Proc. 2000 KYOTO International Conference on Digital Libraries: Research and Practice, November 13-16, 2000, Kyoto University, Kyoto, Japan, (pp. 364-371). Los Alamitos, CA, IEEE Computer Society.

Zhanzi Qiu, Matthias Hemmje, Erich J. Neuhold, “ConSearch: Using Hypertext Contexts as Web Search Boundaries”, in: Etzion, Opher & Scheurmann, Peter (Eds.), International Conference on Cooperative Information Systems (CoopIS 2000), (pp. 42-53). Berlin [etc.]: Springer, 2000 (Lecture notes in computer science; 1901).

Zhanzi Qiu, Matthias Hemmje, Erich J. Neuhold, “Towards Supporting User-Defined Hypertext Contexts in Web Searching”, in Proc. IEEE Advances in Digital Libraries 2000 (pp. 48-57). Los Alamitos, CA, IEEE Computer Society.

Zhanzi Qiu's Curriculum Vitae

Education

- Ph.D. candidate in Computer Science, Darmstadt University of Technology, Darmstadt, Germany, from October 1998 to March 2004
- M. Sc. Information Science, Peking University, Beijing, China, January 1990
- B. Sc. Information Science, Peking University, Beijing, China, July 1987

Professional Positions

- In: **Fraunhofer Integrated Publication and Information Systems Institute** (FhG-IPSI, previously GMD-IPSI), Darmstadt, Germany
    - Research associate / Ph.D. candidate within the division DELITE, October 1998 to March 2004
    - Guest researcher within the division MIPP, November 1995 to December 1997
- In: **Peking University**, Beijing, China
    - Lecturer, Institute of Computer Science and Technology, August 1992 to October 1998
    - Associate Lecturer, Institute of Computer Science and Technology, February 1990 to July 1992
    - Teaching Assistant, Department of Library and Information Science, September 1987 to July 1988 (part-time)
    - ISO 9001 Quality Assurance manager, Founder R&D Center, Founder Group Co., April 1998 to September 1998
- In: **Library of the Chinese Academy of Sciences**, Beijing, China
    - Software Developer, April 1988 to November 1989 (part-time)

Awards

- First Grade of State Science and Technology Progressive Award from State Science and Technology Commission, the People's Republic of China, December 1995
- Special Prize of Beijing Science and Technology Progressive Award from Evaluation Committee of Beijing Science and Technology Progressive Award, Beijing, China, March 1993
- Title of "Beijing Excellent Teacher of 1993" from Educational Work Committee of Beijing Committee of Communist Party of China, Office of Culture and Education of People's Government of Beijing, Beijing Personnel Bureau, Beijing Committee of Educational Worker's Trade Union of China, Beijing, China, September 1993
- Guang_Hua_An_Tai Youth Scientific and Technological Research Achievement Prize from Peking University, Beijing, China, November 1992