DISTRIBUTED BOOKMARK SHARING PRIMITIVES

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING AND INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Kürşat İnce
April, 1999

DISTRIBUTED BOOKMARK SHARING PRIMITIVES

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND INFORMATION SCIENCE

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Kürşat İnce

April, 1999

TK
5105.5
.T53
1999

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Asst. Prof. Ph.D David Davenport(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Asst. Prof. Ph.D. Özgür Ulusoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Asst. Prof. Ph.D. Attila Gürsoy

Approved for the Institute of Engineering and Science:

[illegible]

Prof. Dr. Mehmet Baray, Director of Institute of Engineering and Science

ABSTRACT

DISTRIBUTED BOOKMARK SHARING PRIMITIVES

Kürşat İnce

M.S. in Computer Engineering and Information Science
Supervisor: Asst. Prof. Dr. David Davenport
April, 1999

The popularity of the Internet and the Web, along with the resources on this distributed global network is increasing exponentially. Today, the Web is used for almost every purpose from entertainment to communication, from advertisement to research. Researchers use the Web because the information is fresh, it is close to you as much as your computer, and it is huge. But it lacks structured indexing and/or categorization mechanisms that makes the search process difficult.

People use one or more of four methods to access the information on the Web: search engines, directory services, personal bookmarks, or asking someone who knows where the information is. Search engines and global directory services try to index the Web, but they have their own problems such as too many returned results, low quality, etc. Peoples' bookmarks do not help as much as they should, because they are usually unstructured, being personal. Distributed indexing is one possible solution to this existing problem. Research is going on in distributed indexing, but the resulting systems are not usable by the public yet. Bookmark sharing through bookmark managers is another possible solution provided that the bookmark manager in use can be accessed easily, can store enough information about the resource, and can search its repository effectively. Finally, we propose a new approach called distributed bookmark sharing (DBS), which is both a bookmark manager and a search tool. It is like a Virtual Web on the top of the existing Web.

In our research we built the prototype parts of a DBS, a stand-alone DBS client and a DBS server. We provide mechanisms to insert well-described bookmarks using *Resource Description Templates* and *Resource Description Predicates.* Resource Description Predicates are also used in queries. The DBS server can handle different descriptions of the same resource. The system also provides a simple feedback mechanism to promote and demote the resources. The DBS system, with the help of description templates and description predicates, makes insertions and queries easy and efficient.

Keywords: distributed bookmark sharing, bookmark sharing, distributed indexing, bookmark managers, bookmark organizers, resource description templates, resource description predicates

ÖZET

DAĞITIK BOOKMARK PAYLAŞIM TEMELLERI

Kürşat İnce
Bilgisayar ve Enformatik Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yrd. Doc. Dr. David Davenport
Şubat, 1999

Internet ve Web'in ve bu global ağdaki kaynakların popüleritesi gün geçtikçe daha da artmaktadır. Günümüzde Web eğlenceden iletişime, reklamdan araştırmalara kadar hemen hemen her konuda kullanılmaktadır. Araştırmacılar Web'i, Web'deki bilgiler daha yeni olduğu için, Web bir bilgisayar kadar yakında olduğu için ve Web çok büyük olduğu için tercih ederler. Fakat Web yapısal bir indeksleme ve/veya kategorizasyon mekanizmasından yoksun olduğu için Web'deki bilgi kaynaklarını arayıp bulmak zordur.

İnsanlar Web'deki bilgiye şu dört yoldan birini veya birkaçını kullanarak ulaşırlar: arama motorları, konu dizinleri, kişisel bookmarkları, veya o bilginin nerede olduğunu bilen birilerine sorarak. Arama motorları ve konu dizinleri Web'i indekslemeye çalışırlar, ama sorunsuz değildirler: çok fazla sayıda kaynağı sonuç olarak gönderirler ve bunların kalitesi belirli değildir. Kişisel bookmarklar da yeterince yardımcı değildir: çünkü yapısal olmayıp fazla kişiseldirler. Dağıtık indeksleme, Web'deki problemler için olası bir çözümdür. Bu konuda araştırmalar devam etmektedir, ancak sonuç sistemler henüz halka açık değildirler. Bookmark yöneticileri aracılığı ile bookmarkların paylaşımı kolay kullanılabilir olması, kaynaklarla ilgili yeterli bilgiyi saklayabilmesi ve bunları tarama imkânı verebilmesi şartıyla başka olası bir çözümdür. Bizler, hem bookmark paylaşım yöneticisi hem de arama aleti olan Dağıtık Bookmark Paylaşımını (Distribute Bookmark Sharing, DBS) geliştirdik.

Araştırmamızda, bir DBS istemcisi ve bir DBS sunucusu olan DBS prototipini geliştirdik. Kaynak Tanımlama Şablonları ve Kaynak Tanımlama Yüklemleri kullanarak bookmark ekleme mekanizması sağladık. Kaynak Tanımlama Yüklemlerini sorgulamada kullandık. DBS sunucusunu aynı kaynağa ait farklı tanımlamaları kabul edecek şekilde geliştirdik. Kaynakların yükselmesi ve alçalması için geri bildirim mekanizmasını sağladık. DBS sistemi, kaynak tanımlama şablonları ve kaynak tanımlama yüklemlerini yardımı ile ekleme ve sorguları daha etkili hale getirdi.

Anahtar kelimeler: dağıtık bookmark paylaşımı, bookmark paylaşımı, dağıtık indeksleme, bookmark yöneticileri, bookmark organizatörleri, kaynak tanımlama şablonları, kaynak tanımlama yüklemleri

To my family

ACKNOWLEDGMENTS

I am very grateful to my supervisor Asst. Prof. David Davenport for his valuable guidance and support during this study.

I would like to thank Dr. Tacettin Köprülü for his comments on my previous thesis proposals and this final thesis report. I would like to thank my friends (especially in DSP and IFW) for sharing their bookmarks. I would like to thank Betül, Dilek, and Dilara for their motivating words. And finally, I would like to thank Nurgül for being with me all the times.

I would like to thank my jury members Asst. Prof. Özgür Ulusoy and Asst. Prof. Attila Gürsoy for their comments and suggestions.

Contents

List of Figures

Chapter 1

Introduction

For centuries, people store and/or convey information as written texts on paper and other media. Computers have changed the way this has been done, and now we are using the digital technology. With automated systems, it is now possible not only to store but also to search and retrieve without considering the size of text.

1.1 Information Retrieval and the Internet

Information retrieval (IR) is automatically storing documents (of whatever kind) in a document database, searching the database for user queries, and retrieving the documents that are relevant to the queries.

Information retrieval process consists of four main phases: [Lew92]

- The documents have to be stored in the document database (library). But the library can be so large -in terms of number of documents or in terms of document size- that we have to find some effective way of representing documents. IR systems convert documents into their representatives. This process is known as indexing the documents.
- The user must express her/his information needs in the form of a request understandable by the IR system. The user provides queries to the system. This process is known as querying the system.
- The IR system has to compare the user queries to the stored document representatives, and make decisions about which documents to retrieve and in what order. This process is known as comparison.

- Sometimes the user of the query may not be satisfied with the retrieved documents. In this case, s/he modifies the query explicitly for better matches. Several iterations may be needed to achieve acceptable results. In some cases, the IR system allows users to provide feedback on the retrieved documents. This feedback is used to modify the query explicitly. This process of modifying queries is known as relevance feedback.

The Internet and the World Wide Web (the Web) are one of the most important tools for both education and research. The Web is a huge library, which contains terabytes of information in digital form. But it lacks structured indexing and/or categorization mechanisms which makes the search process difficult.

The popularity of the Internet and the Web, and the resources on this global network are increasing exponentially. Today, the Web is used for almost every purpose from entertainment to communication and from advertisement to research. Researchers use the Web because the information on the Web is fresh, it is close to you as much as your computer being almost free. But the structure of the Web doesn't allow them easily to find the information they need.

The documents on the Web are HTML documents, which are in fact hypertexts. A hypertext contains links to other hypertexts, and so on. The Web is a pool of such hyperlinked documents.

Finding relevant information is problematic because: [LB96]

- The Web is unstructured,
- The number of documents grow exponentially in time,
- The number of hyperlinks grow exponentially in time,
- Not every link you follow gets you to the true target.

1.2 Resource Discovery on the Web

The Web is a huge digital collection, where people can find information on almost any topic. People find Web resources using one or more of the following:

1.2.1 Search Engines

People may use one of the 100+ search engines available on the Web. These search engines crawl the Web, index the documents and store the indexes in central databases. People submit their queries to these servers, which searches the index for the queries. Although many of the users are satisfied with the results, centralized search engines have problems:

- The exponential growth of the Internet makes handling of information indexing difficult. When the size of the Internet doubles, that increases the load of the search engines four times. [Kos97]
- Search results are satisfactory, but many of the search engines return different results. This shows that none of them provides a comprehensive coverage of the Web. [Kos97]
- Search engines uses web spiders that crawl the Web sites. This causes temporal network congestion on the Web site. Furthermore, 100+ search engines want to index the Web sites without knowing that it has already been indexed by another search engine. Worst of all, the search engines want to index the Web site periodically.

  Using web spiders also decreases network bandwidth of the central index server and the other sites on the route.
- The quality of returned results are not good. Returning as much results as a server can, usually doesn't help. Current information retrieval techniques provide about 60-70 percent precision. [Kos97]

1.2.2 Directory Services

People can also use one of the information directories. These directories store information to resources in a topic hierarchy. The user can browse through this hierarchy tree to find the relevant information. Some of the search engines provide such a facility. Directories also have some problems:

- Categorization of the resources is difficult. There are so many documents to categorize that automated categorization is needed. Resources can be listed under more than one category. Manual (human) categorization can be possible, but it will be slower. Also in this case categorization will not be objective since people usually categorize the resources differently [AD94].
- Exponential growth of the Internet is also a problem for directory services. When the size of the Internet doubles, that increases the load of the search engines nearly four times [Kos97].

- Web spiders that are crawling the Web which results network congestion is also a problem. This causes temporal network congestion on the Web site.
- Directories can not list all available resources for a given topic. They are not comprehensive either.

1.2.3 Personal Bookmarks

Most people create bookmarks for the resources they find useful. Some of them use their bookmarks to find the information in those resources. The problems with bookmarks are as follows:

- They may not have bookmarks for every topic they are interested in.
- Management of bookmarks is a problem. Most of the Web browsers (Netscape, Internet Explorer... and others) provide built in hierarchical bookmark managers. These bookmark managers can be used to organize the bookmarks in a hierarchical folder structure. In such a scheme each folder is a subject area. But people usually keep bookmarks in a flat structure, which makes the management more problematic.
- Some bookmarks may need to be stored in more than one subject folder. Some bookmark managers (e.g. Netscape) provide aliasing mechanisms for bookmarks. Unfortunately they do not provide a cross-reference for bookmarks [Kea97a]. There is no way to know under which categories a bookmark is placed.
- Bookmarks are stored locally. They can not be shared between individuals very easily [Kea97a].
- Some of the bookmarks are used more often, but bookmark managers don't use this fact to rank the bookmarks [Kea97a].
- If hierarchy level is deep, then seeking for a bookmark at lower levels can be a problem [Kea97a].

1.2.4 Ask Somebody

Finally, people ask the resource to others who knows where the information is, but they may not find somebody for every topic they are interested in.

1.3 Distributed Bookmark Sharing

Distributed indexing is one possible solution to the existing problems of the Web. Research is going on in distributed indexing, but the resulting systems are not usable by the public yet. Bookmark sharing through bookmark managers is another possible solution provided that the bookmark manager in use can be accessed easily, can store enough information about the resource, and can search its repository effectively.

Distributed bookmark sharing (DBS) is a new approach to solve the problems with current information discovery techniques, and to utilize the bookmarks of Internet users. It is based on the idea that, if I like some page, and create a bookmark for that page, someone else will also like it, and make use of it.

DBS is not just another distributed index or search engine. It is both a bookmark manager and a search tool.

In our research, we built the prototype parts of a DBS, a stand-alone DBS client and a DBS server. We provide mechanisms to insert well-described bookmarks using Resource Description Templates and Resource Description Predicates. Resource Description Predicates are also used in queries. The DBS server can handle different descriptions of the same resource. The system also provides a simple feedback mechanism to promote and demote the resources. The DBS system, with the help of description templates and description predicates makes insertions and queries easy and efficient.

In Chapter 2, we give background information on current state of bookmark manager, distributed indexes, and resource description templates. In Chapter 3, we describe DBS design and architecture in detail. In Chapter 4, we describe our research on DBS in detail. In this research we built a prototype parts of DBS. In Chapter 5, we describe the details of our implementation and how it differs from our initial DBS design. In Chapter 6, we discuss our tests and observations. Finally in Chapter 7, we discuss DBS vs. our implementation and tests, and also provide enhancements to our current implementation and current DBS architecture.

Chapter 2

Related Work

Distributed bookmark sharing is closely related to bookmark management, distributed indexing, and description templates. In this section, we provide overview of current work for these three topics.

2.1 Bookmark Management, Organization and Sharing

The aim of a bookmark manager or organizer is to store bookmarks in such a way that adding new ones and searching/fetching existing ones can be possible. Most of the Web browsers today provide built-in bookmark management.

Bookmark managers usually provide the following functionality:

- Add new bookmarks without much trouble.
- Add description to the bookmarks
- Create/use folders for a hierarchical bookmark scheme.
- Insert the same bookmark to more than one topic (aliasing)
- Search for a bookmark, based on the name, title, description, etc.
- Browse through the bookmarks

The following are missing:

- Check for validity of the bookmarks. There are external utilities to do this.
- Export/import certain types of bookmarks -there is no bookmarking standard. There are external programs for doing conversion between different types.

2.1.1 Built-in Bookmark Managers

Anybody who uses Internet and the Web is aware of the bookmark managers which are provided by the browsing software. In this subsection, we are going to compare these bookmark managers.

NCSA Mosaic

Bookmarks in Mosaic is called "hotlists". Mosaic's bookmark manager just stores the bookmarks and displays them as a sub-menu when needed. It doesn't provide a mechanism to describe the resource. You have to remember the title of the resource in order to access it. You can not search the hotlist for description or title. It doesn't provide a hierarchy for bookmark either. Everything is a flat structure.

Mosaic is the first of its kind, and it was very popular in the early days of the Web.

Microsoft Internet Explorer

Microsoft Internet PC project resulted in one of the popular browsers. Today, Internet Explorer (IE)[1] is built into the desktop operating systems of Microsoft. IE bookmarks are called as "favorites".

As a browser and a bookmark manager, IE is one of the best. You can insert a new bookmark easily, either by selecting from the menu or by drag-and-drop. The bookmarks can be stored as hierarchies, and during insertion of a bookmark, you can select where to insert, so you don't have to pass through the whole list to rearrange the bookmarks. IE also provides an interface to arrange and browse through the bookmarks. You cannot add descriptions to the bookmarks and you cannot query your favorites either.

[1]http://www.microsoft.com

Netscape Communicator

Communicator[2] (or Netscape as most users call it) is another popular browser. You can insert bookmarks easily, from the menu or by drag-and-drop. The bookmarks can be stored hierarchically, and by drag-and-drop you can select where to insert the bookmark. Otherwise, Communicator stores the bookmark at the highest level which requires further processing by the bookmark manager. You can give descriptions to the bookmark, but you have to invoke the bookmark manager for this explicitly. The bookmark manager provides a search mechanism for the bookmarks. You can search for keywords that appear in the title or in the description. You can also browse though the bookmarks by title, URL, or submission date, etc.

Earlier versions of Communicator used to provide functionality for validity check of the bookmark. Today you have to use some other external tools.

Opera Software Opera

Opera from Opera Software[3] is another popular browser for the Internet. One of the distinguishing feature of Opera is, it always displays the bookmark manager. The bookmark manager allows drag-and-drop bookmark insertion and bookmark hierarchies. You cannot add descriptions, or query the bookmarks.

2.1.2 itList

When you use built-in bookmark managers, the bookmarks are stored locally on your hard drive. itList [iWA98] is an on-line bookmark manager which can be access on any computer on the Internet. So you really don't have to carry your bookmarks with you when subscribe and use itList.

Inserting a new bookmark is as easy as sending an email to the itList server. Your bookmark is automatically inserted into you list of bookmarks when the mail reaches to the server. Since the server is open to the public, itList also uses a simple authentication mechanism for insertion. The server provides functionality for editing the bookmark list. Your bookmarks can not be stored hierarchically, but you can define categories which can basically does the same job. You can browse through the bookmarks that are submitted to the server, but you cannot search for specific bookmarks.

[2]http://www.netscape.com

[3]http://www.operasoftware.com

2.1.3 Webtagger

In the previous section, we have discussed problems with using bookmarks -and bookmark managers- to find resources on the Web. Webtagger [Kea97a] is an ongoing research on bookmark management. The aim is to overcome the problems with bookmarks which are discussed in Section 1.2.3.

Webtagger is a bookmark service where authorized users can store, organize, search, and evaluate search results.

Webtagger can be used in two ways:

- As a personal bookmark manager (for personal memory),
- Or as a medium to share bookmarks among the group members (for group memory).

In both cases people categorize their bookmarks and add them to their repositories. When they search the bookmarks, they also provide some kind of feedback to the Webtagger system. Webtagger uses that information to rearrange the rank of the bookmarks.

Webtagger is implemented as a proxy. So potentially any Web browser can be used as a Webtagger client, or user interface. However, in this case web access will be slow, since the Webtagger intercepts requests. Modification of the browser might be a better choice but this will require more time to implement and can not be universally deliverable.

The user interface provides the following functionality:

Categorization At the time of submitting a bookmark, or at some later time the user can categorize the bookmark with predefined set of categories, or create a new category if it doesn't match one. More than one category label can be given to the bookmarks. There is no automatic categorization feature. The user has to do it manually.

Retrieval The user can query the repository based on keywords or based on categories.

Feedback The retrieval results can be used to give feedback about the resulting bookmarks.

Import The user can ask the system to import existing (Netscape) bookmarks to the system. Based on the existing bookmark hierarchy the server adds the bookmarks in the corresponding categories.

Browse The user can also view the entire repository based on category.

Webtagger provides a simple means of organizing and sharing bookmarks using lattice structure in categorization rather than a hierarchical one. This eliminates the cross-referencing problem of aliasing bookmarks.

As a personal bookmark manager, Webtagger is promising. On the other hand, as a bookmark sharing medium, Webtagger has the following problems:

- Group memories are not moderated. As a result, anyone can insert any bookmarks, which may decrease the value of the group memory.
- When someone adds a new bookmark, other users may not be aware of the new bookmark.
- People may be using different categorization schemes. In this case they may add the same bookmarks to different categories.

2.1.4 Bookmark Organizer (BO)

Built in bookmark managers provide some sort of primitive classification based on users manual classification. Manual classification and organization can be difficult when there are more than a few bookmarks to classify or when the user lost track of his classification scheme.

BO [MS96] is a semi automated bookmark classification and organization tool.

This is how people bookmark a resource:

1. Surf the Web
2. Like the resource
3. Store it in local repository using built in bookmark manager
4. Classify the bookmark (often forgotten)

Once you skip the 4th step, and add more and more bookmarks, then we'll have what we call a flat organization (or no organization at all).

Classification of resources and forming a hierarchy out of a flat organization is the most difficult part. While bookmarks are being classified it may be difficult to remember why the resource was added. You have to go and check it. Some of the current built-in bookmark managers allow you to add descriptions to the resources. But that is also usually forgotten. As a result, manual classification takes a long time.

One other problem is what to do when you import bookmark file of others. The classification scheme of different users may not be the same. As a result, what you added to your repository is not effectively usable.

BO, as a semi automated classification tool, can help to manage all of these problems. It uses some modified version of standard categorization techniques. Standard techniques can not be directly applied since bookmark repositories are smaller in size, change more frequently, and may require user interaction. On the other hand, bookmarks point smaller documents, which increases the performance of the categorization process.

BO provides a hybrid approach to bookmark organization. In this approach manual and automatic classification are equal partners. The user freezes the nodes, and let the BO do the rest of the classification.

The user can mark high level categories, which are near to the top of tree hierarchy as frozen. These nodes are taken as the starting point for classification.

BO is an external program that is executed by the user from time to time. The only requirement for BO is the browser support for hierarchical structure and annotations.

BO first parses the hierarchy and detects the changes made to the repository since last invocation. Later it applies semi automatic method on the documents that are in a sub-tree to find the closest ancestor frozen node and re-cluster its sub-tree. Finally it applies full automatic method on the documents that are on the top level or that are not in a sub-tree.

2.2 Distributed Indexing

Distributed indexing is decentralized indexing of the resources on the Web. Research is going on in distributed indexing, but the resulting systems (distributed search engines) are not usable by the public yet.

We have discussed problems with centralized search engines in Section 1.2.1. Distributed information resources, such as the Web, have to be indexed distributed. Distributed indexing is one possible solution to the existing problems of the Web. Exponential growth of the Internet can be handled by employing more distributed index servers. This is different than building new centralized search engines having the advantage of less network bandwidth consumption. Distributed index servers indexes non-overlapping segments of the Web space. New resources will be indexed by new distributed index servers. Non-overlapping segments ensure that the Web site is indexed only once, although we cannot say some other distributed search engine does not index the Web site. In addition periodic re-indexing cannot be avoided. Distributed indexing allows us to index or handle large volume of information effectively.

The requirements with distributed indexing are:

- They have to be adaptive in order to handle dynamic content.
- They have to provide a common way to deal with all kind of resources for heterogeneity.
- Finally, they have to prevent low quality information.

2.2.1 WHOIS++

WHOIS [HSF85] is a protocol that is used to provide information about its registered entries (can be individuals, hosts ...etc.) to its users. This simple directory service represents each entry as a collection of text with minimal structure.

WHOIS++ [Dea95, BFS96] based on WHOIS protocol, is a newer protocol which supports distributed directory services. WHOIS++ provides the following extensions to the original WHOIS protocol:

- More structured information by means of the templates. Each template has an ordered set of attribute/value pair to store the information.
- Each WHOIS++ server is assigned a unique server handle, and each record in the server has a unique record handle. This provides a unique identification of a resource on the Internet.

Organizing information by means of predefined templates makes it easier for administrators to gather and maintain records. This may encourage them to make such information available. The users of the service become more familiar with the templates and are able to query the server more intelligently.

WHOIS++ has two different components:

- The base server, which contains only filled templates. This acts as a WHOIS server, and answers the queries submitted by some clients.
- The indexing server, which contains forward knowledge (record abstracts) and pointers to other base servers or indexing servers.

Indexing is used to tie the WHOIS++ servers together to form a distributed directory service. Each base server and indexing server who wishes to participate this global network have to generate forward knowledge for the records it contains.

Indexing servers can collect forward knowledge for any servers it wishes. As a result, a query submitted to an indexing server is effectively forwarded to other WHOIS++ servers which the indexing server knows about. It generates results with pointers to actual WHOIS++ server where the record is found.

2.2.2 What'sHot

What'sHot [Kos97, Kos96] is a collection of distributed indexes.

The first step to distributed indexing is a proxy search engine (PSE), which caches queries submitted to the search engines, and the query results. As a result, people in a corporation can share their search results transparently. This idea works because people often require the same information, especially in a corporation [Kos97]. That is the same reason why USENET newsgroups are the first place to search for a resource.

Proxy search engine intercepts queries of the local users and attempts to answer them using a local index, which works as a cache. If it fails to answer a query or the users are not satisfied with the information available locally, the PSE submits the requests to a central database and gets the answers from there. It caches the answers locally and forwards them to the user.

Proxy's cache stores abstracts: information descriptor objects. Abstract is a unit of meta information (with 1-2 K bits) for description of the resource.

The main advantage of PSE is that search queries are resolved locally, much faster and with no network overhead. It is ideal for Internet service providers (ISP).

The second step to distributed indexing is shared bookmark managers. The users' evaluation of the resource sets the lifetime of the resource on the proxy. This is known as cooperative information filtering [Kea97b, KSS97]. The feedback can be implicit (results are considered useful) or explicit (user gives feedback about the resource). Both are used in a combination to find actual feedback value in What'sHot. This feedback can be used for ranking of the resources in the cache. As a result, What'sHot is able to propose the most valuable (high ranked) resources first.

The next step to distributed indexing is the specialization of the PSE servers. This is a good way to achieve better information service. This leads to concentration of knowledge at certain places. People can find the resource easily. PSE servers can be converted to special servers by analyzing the cache and usage statistics to find out which of the abstracts are used and how. The topic will be chosen from those. Finally, users are made aware of the specialized PSE and new rules are developed that will change the lifetime of the abstracts in the cache.

Finally, we have a set of PSE servers, most of which are specialized. It makes sense to enable automatic use of this specialization in the search process. If a PSE fails to find an answer, it first goes to a fixed number of other PSE servers and only then to a central database.

Initial query routing will be random -to a server from an initial set of available PSE servers- but after some time the local PSE will learn how to route the queries to which of the other PSE servers.

What'sHot assumes the following:

- Authors provide abstract for resources they make available, or it is possible for What'sHot to extract the abstract from the resource.
- People are willing to give feedback.

Each What'sHot server has a database and software to perform the following:

- Maintain index and search the local cache of the abstracts
- Process incoming queries locally and send results back to the client
- Make decisions on routing the query to other brokers if the user is not satisfied with the results
- Make decisions on caching of remote query results locally
- Publish abstracts of local users
- Periodically recommend most popular abstracts to the other brokers
- Make decisions on caching locally abstracts that are recommended by other brokers

What'sHot is used as follows:

The URL for the resource and an abstract of the resource are stored in the repository. The more popular the resource is, the more it stays in the repository, and the more it is split among other repositories. The user uses one of What'sHot servers Web page as her/his search engine. If possible, the query is resolved locally. Different from other search engine results, the results returned from What'sHot has relevancy check-boxes. The user checks some of those if s/he finds them useful. If the user is not satisfied with the answers, s/he provides a negative feedback. In that case What'sHot server routes the query to other servers. The search and query routing mechanism is fully transparent to the users. Using What'sHot as a search engine is like using any of the search engines.

Another use of What'sHot is resource routing. The user may send her/his profile to a broker. The broker forwards new resources to the user based on the user profile.

2.2.3 Harvest

Harvest [Har96] is an integrated set of tools developed by the Internet Research Task Force Research Group on Resource Discovery. This set of tools provide the following functionality of the Harvest system:

- Gather the resource from a specific site or from the Internet
- Extract the information from the resource
- Organize the information to fit into Harvest (Summary Object Interchange Format SOIF) templates
- Index SOIF templates
- Search the index
- Cache the search results
- Replicate information across other Harvest systems.

One important advantage of Harvest is that it gathers the resource and extracts the summary information from that automatically. This is different than WHOIS++ and What'sHot, where the abstracts are constructed manually.

Harvest consists of several subsystems:

- The Gatherer subsystem collects indexing information from the resources available from the sites or the Internet.

  The Gatherer retrieves information using various methods (FTP, HTTP, etc.) and then summarizes these resources to generate structured indexing information. For example a Gatherer can retrieve a technical report from a personal homepage and extract title, author name, and abstract from the report, and also summarizes the report. Harvest Brokers can then retrieve the indexing information from the Gatherer to use in a searchable index available via a Web interface.

- The Broker subsystem retrieves indexing information from one or more Gatherers, incrementally indexes collected information, and provides a Web interface for the user queries.
- The Replicator subsystem replicates Brokers' data to other Brokers over the Internet.
- The Object Cache subsystem provides information to the users. It allows users to retrieve FTP, HTTP, etc. data quickly and efficiently. .

Harvest systems over the Internet are connected through the Harvest Server Registry. Creators of the Harvest suggest new Harvest servers should be registered to the Harvest Server Registry for optimum effectiveness and efficiency. This registry holds information about all available Harvest Gatherers, Brokers, Replicators, and Cache over the Internet. As a result, Brokers and/or Replicators can know what information to exchange with whom. That is how effectiveness and efficiency increases by creating new Harvest servers for the Internet.

2.3 Description Templates

Templates are usually used in information extraction tasks to create descriptions, summaries and other information.

In this section we will see three of the most useful templates in the Internet domain.

2.3.1 Linux Software Map (LSM) Templates

The Linux software is distributed with a description file. This file is known as LSM (Linux software map) file [BW95].

The LSM template contains the following 12 colon separated attribute-value pairs:

Title Title of the software

Version Version of the software

Entered-date Archiving date

Description Natural language description of the software

Keywords Keyword description of the software

Author Author of the software

Maintained-by Maintainer of the software

Primary-site Primary distribution site of the software

Alternate-site Alternative distribution site of the software

Original-site Original software site

Platforms Operating systems that the software runs/compiles

Copying-policy How can the software be distributed

When people are submitting new software to the archive, they are encouraged to submit them with their corresponding LSM descriptions. Otherwise, archive maintainer may refuse to archive the software.

Tools exist to process LSM files and index the software.

LSM templates are useful for software packages. They are created to describe software only, they can not be used to describe other resources on the Internet.

2.3.2 Internet Anonymous FTP Archive (IAFA) Templates

The Internet Engineering Task Force (IETF) Working Group on the Internet Anonymous FTP Archives (IAFA) have produced the IAFA templates, which defines a range of indexing information that can be used to describe the contents and service provided by anonymous FTP archives. [Bec, Mem98a, Mem98b] Like LSM templates, IAFA templates contain attribute-value pairs. They are not restricted to software only, and can be used to describe documents, multimedia objects, etc.

Available IAFA templates are as follows:

- Software:
- Service:

- Document:
- Mailarchive:
- USENET:
- Image:
- Sound:
- Video:
- FAQ:
- TRAINMAT:
- DATASET:

The following attributes are common in all of the templates: [Mem98a, Mem98b]

- Handle:
- Category:
- Title:
- Language:
- URI:
- Description:
- Keywords:
- Subject description scheme:
- Subject description:

Tools exist to process IAFA templates. [Bec]

2.3.3 Summary Object Interchange Format (SOIF) Templates

Harvest [Har96] Gatherers and Brokers communicate using Summary object interchange format templates. Gatherers generate content summaries for individual resources in SOIF, and serve these summaries to Brokers that wish to collect and index them.

The following attributes are common in all of the templates:

Abstract Brief abstract about the object

Author Author(s) of the object

Description Brief description about the object

File-size Number of bytes in the object

Full-text Entire content of the object

Gatherer-host Hosts on which the Gatherer ran to extract information from the object

Gatherer-name Name of the Gatherer that extracted information from the object

Gatherer-port Port number on the Gatherer-host that servers the Gatherer's information

Gatherer-version Version number of the Gatherer

Update-time The time that gatherer updated the content summary for the object

Keywords Searchable keywords extracted from the object

Last-modification0time The time that the object was last modified

MD5 MD5 checksum of the object

Refresh-rate The number of seconds after Update-time when the summary object is to be re-generated

Time-to-live The number of seconds after Update-time when the summary object is no longer valid

Title Title of the object

Type The object's type. Some of the object types are: Archive, audio, compressed, compressed-tar, GNU-compressed, GNU-compressed-tar, postscript, TeX...etc.

2.4 General Discussions

The following facts were observed during the survey of background information [Kos97, Kos96, Kea97a]:

- Especially in specialized environments, if someone is looking for some information, someone else might have already looked for it.
- Users are satisfied with almost any results from search engines because they use those to find more relevant links and resources.
- Expert selection of valuable resources (like the USENET) is useful.
- The Web is so huge that you can easily get lost when searching for resource. It is larger than necessary, since there are duplicate resource, low value resource, and others. Virtual Web is a new Web built on the existing one, with high quality information, without duplicate entries.

Chapter 3

Distributed Bookmark Sharing

Distributed bookmark sharing (DBS) is a new approach to solve the existing problems of the Web. DBS is not just another distributed index or search engine. It is both a bookmark manager and a search tool.

The next section will give definitions of the system components to develop a common dictionary of terms in this report. Section 3.2 will describe the DBS architecture. Section 3.3 will describe the three phases of DBS working mechanism: storage, retrieval, and search.

3.1 Definitions

Submit Add a new resource to the DBS system.

Retrieve Check for/browse through the bookmarks in the DBS system.

Search Query the DBS system for possible good resources. This is different than browsing through the bookmarks.

URL Uniform Resource Locator, the Internet address of a resource/document on the Internet.

Resource The URL and its description that is stored in the DBS system.

Bookmark Another name that is given to the resources in DBS.

Participant Users of the DBS system, their computers, and the client on that computer.

Node A participant to the DBS system.

3.2 DBS Architecture

DBS system consists of DBS servers and DBS clients working together.

In the heart of the DBS system there are DBS clients -participants or nodes- that share bookmarks through the DBS server. A participant is a user with a computer and a small program (DBS client) running on the computer. DBS clients are connected to one of DBS servers through a network.

Distributed bookmark sharing is done via DBS clients working together with the DBS server. They provide an interface to users to submit, retrieve, and search available -submitted- bookmarks.

DBS servers are connected together through the Internet. They store resources from the DBS clients, send resource to the DBS clients, search for resources and send results to the clients. A DBS server can act as a client to another DBS server for search function only. They cannot submit new resources, but they may store results of the searches that are locally submitted to other servers. Figure 3.1 shows the general architecture of DBS system.

3.3 DBS Overview

There are three phases in this system: the storage phase, the retrieval phase and the search phase. In the storage phase, the participant adds a new bookmark to the system. In the retrieval phase, the participant browses the bookmarks in the system. In the search phase the participant queries the system for specific bookmarks.

The rest of this section will be about participation of the users, storage of new bookmarks, retrieval of existing bookmarks and search of the pages.

3.3.1 Participation

The user participates to DBS system after s/he installs the DBS client participant and the DBS server. Currently, no security or privacy issues are considered. Anybody can access the server, if s/he has the necessary client.

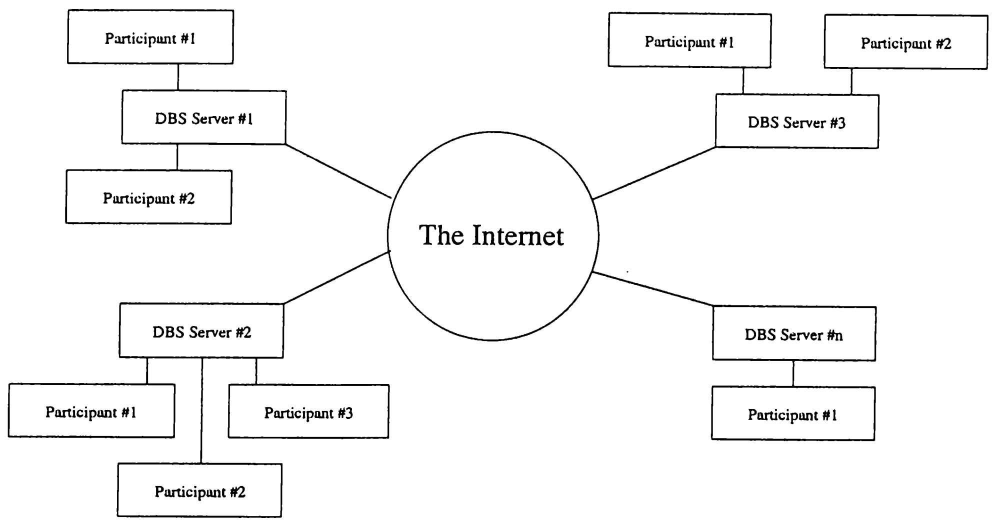


Figure 3.1: DBS Architecture

3.3.2 Submission

When the participant hits a web page, which s/he thinks that is interesting to her/him and to the other participants, s/he adds a bookmark for that page by means of the DBS client.

The participant adds a new bookmark to the system by using DBS client. Adding a new bookmark needs to be as simple as possible. S/he also provides a description for the bookmark. The DBS client forwards the bookmark to DBS server.

3.3.3 Retrieval

The participant may want to retrieve bookmarks based on some selection criteria, the site of the bookmark, submitter, etc., or s/he may want to browse through the bookmarks.

3.3.4 Search

When the participant submits a query to the DBS system by means of the DBS client on his computer, that query is forwarded to DBS server.

The participant enters the query terms by means of the DBS client on his computer. The DBS client forwards that query to DBS server. DBS server resolves the queries locally or forwards the queries to other DBS servers. It combines the results, calculates the value of the resources, and sends the results back to the DBS client. DBS client displays the results.

3.3.5 Feedback

The participant can provide a feedback, either positive or negative, for the results DBS server sends.

The DBS client forwards the feedback to the DBS server for a resource. The server updates its definition of the resource to reflect the feedback. As a result, the resource can be promoted or demoted.

3.4 DBS vs. Other Systems

In this section, we have summarized the distinguishing features of DBS:

- DBS is both a bookmark manager and a search tool.
- DBS is a complete bookmark manager. The users can store their bookmarks, browse through the bookmarks, and search for bookmarks. Additionally, DBS provides a mechanism to rank the bookmarks and another mechanism to get the user feedback.
- DBS does not use a hierarchical bookmark scheme. Every bookmark is stored at the same level. But the browsing and query mechanism provides easy ways to find bookmarks in the DBS system.
- DBS is a complete distributed index. However, DBS can not handle rapid changes and dynamic content of the Web, since it can not be aware of the new resources until some user submits that resource. DBS provides a mechanism to promote and demote resources which protects users from low quality information.

Chapter 4

Research on DBS Primitives

What interests us most in this research is how to discribe resources, how to query for the resources, and how to calculate the resource quality in a distributed (multiuser) environment. A study on different users' bookmarks is given first. After that we describe what the DBS primitives are and how they are used.

4.1 User Bookmark Profiles

Before starting our research on DBS, we wanted to learn about people's bookmarks and wanted to build a profile for them from their bookmarks.

We asked people to share their bookmarks with us. We got 11 Netscape Communicator bookmark files with a total of 1519 entries, of which 1316 was unique.

We observed the following after a quick analyze:

- The size of the files were ranging from 15 to 500 bookmarks.
- 7 of the files were structured hierarchically.
- None of the users provided a description to any of the bookmarks.
- Bookmarks were related to their research areas or their hobbies.
- The 203 none unique entries were either predefined by the bookmarks manager or of general interests such as daily newspapers or their favorite search engines.

We also checked the resources by browsing through them. This time we observed the following:

- 1/4 of the bookmarks (326) were outdated, and were removed from the Web.
- 1/4 of remaining 1193 (278) resources provided a meta description of the resource. This description was in natural language or in terms of keywords for that resource.

4.2 Bookmark Sharing Primitives

As it is stated before, we restricted ourselves to a stand-alone DBS server, and researched for better ways to make it work. In this section we will give a short summary on our research areas. The following sections will describe them in detail.

4.2.1 Bookmark description

Most of the current bookmark managers do not allow you to give descriptions to the bookmarks. Some of them do, but the description has to be given after submitting the bookmarks. Usually the description is given in natural language.

We think that resource descriptions need to be integrated to bookmark submission. On the other hand, natural language descriptions suffer from the following:

- Natural language descriptions are hard to search. The query mechanism for searching descriptions has to handle keywords as well as phrases, negative meanings, etc.
- Natural language descriptions are hard to combine. Consider the following: Two different users of the DBS system insert the same URL as a bookmark. How can the server merge two descriptions and generate a new one?

In DBS system, resource description is integrated to bookmark submission in two ways:

- Resource description templates provide a template-based description of the resources. Description templates are predefined in DBS system.
- Resource description predicates provide a predicate-based description of the resources. Description predicates are dynamic and new predicates can be added by the user for a better description of the resource.

4.2.2 Bookmark quality

Most of the current bookmark managers do not rank the bookmarks. We think that bookmark usage statistics can be used to do this. In this case, most recently used, or more often used bookmarks can be listed above the other bookmarks.

Bookmark managers do not allow you to give feedback about the bookmarks either.

Bookmark quality is the key in DBS while we rank the query results. DBS uses feedback, predicate values, and user experience level to calculate quality of the resource. These numeric values are used while sorting query results in descending order.

4.2.3 User trust level

DBS is a distributed information sharing service. We think that not all the bookmark submitters are at the same trust level. This is related to her/his experience on the topic.

DBS uses user submissions and queries to calculate the user trust level numerically. This value is used to bookmark quality also.

4.3 Resource Description Templates

Distributed information indexes, such as Harvest have to define the resource as compact as possible. Description of the resource has to be transfered from a server to another, and also stored locally. In Harvest this is achieved by SOIF templates. Templates provide a compact and well-defined description for the resource.

Natural language cannot be use to described a resource, as we have seen earlier in Section 4.2.1. In fact, both of the above problems can be solved by integrating an information extraction system. Such systems are available for restricted domains only. The input to an information extraction system is a written text, which can be news from a journal, or an article. The output of an information extraction process is either the summary, or description of the input text. Summary is still in natural language, but the description is usually in terms of templates defined for that domain [HS96, ea97, CL96, Cun97].

DBS also make use of description templates. These templates provides the following simplifications in DBS system:

- Users can describe the resources by filling a template. Nothing else is required from the user at this insertion phase.

- Resources can be stored and exchanged efficiently among the DBS servers.
- Queries can be solved efficiently.
- Information that is stored as templates is structured as opposed to written text which is not structured.

DBS description templates have 6 top level categories and 27 subcategories. This section will describe how we decide on these categories and on attribute-value pairs that are used with these categories.

4.3.1 IAFA Templates -Revisited

The Internet Engineering Task Force (IETF) Working Group on the Internet Anonymous FTP Archives (IAFA) have produced the IAFA templates, which defines a range of indexing information that can be used to describe the contents and service provided by anonymous FTP archives.

IAFA defines the following 11 type of templates/categories.

- Software:
- Service:
- Document:
- Mailarchive:
- USENET:
- Image:
- Sound:
- Video:
- FAQ:
- TRAINMAT:
- DATASET:

These categories provide the initial start point for DBS description templates. The following problems exists with these categories:

- There are no subcategories which makes us think that these categories are general for any resource of the system. In fact, what we have seen is Document, software, image, sound, video, mailarchive, USENET, dataset, and FAQ share the same attribute pairs [Mem98a]. There will be and there are attributes which might make sense in all cases.
- Image, sound, video are in fact of the same type, so these might have been considered as multimedia objects.
- FAQ is a document since it provides an information. This and above observation make us think that these categories are specific for such kind of resources.

The following attributes are common in all of the templates:

- Handle:
- Category:
- Title:
- Language:
- URI:
- Description:
- Keywords:
- Subject description scheme:
- Subject description:

We still have problems with these common attributes. It is not really clear what a "Subject description scheme" and a "Subject description" is. Worst, these force people to be expert on IAFA templates to give a correct description of the subject of the resource.

Roads IAFA Template Usage Statistics [Mem98b] show that none trivial attributes of the template tend to be left unfilled most of the time by the users.

From the user point of view, we don't really want to spend time on attributes which we won't make use of. For DBS we have defined a new set of categories and attributes to encourage people fill in the description templates.

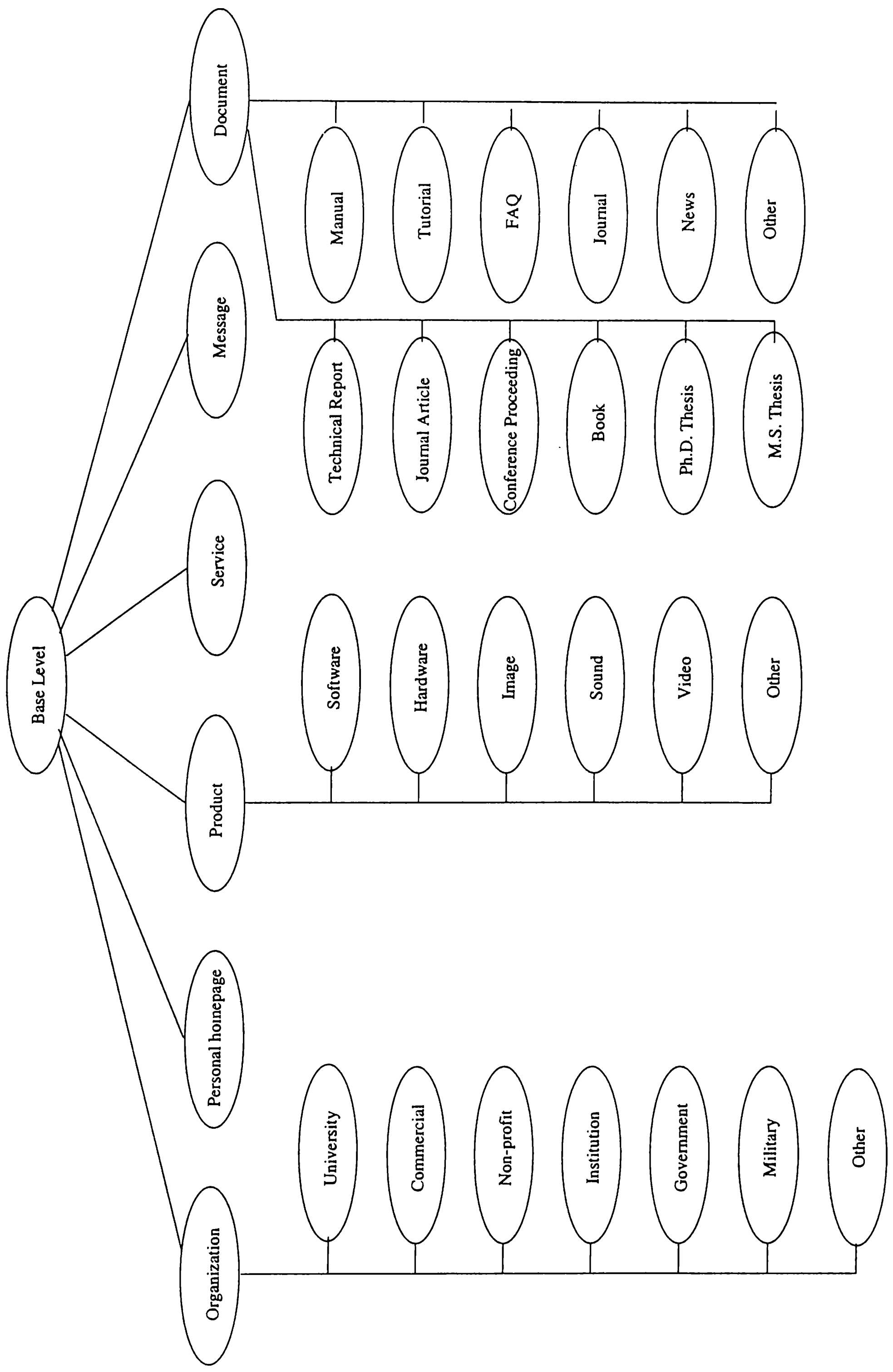


Figure 4.1: DBS Description Template Categories

4.3.2 DBS Description Templates

Unlike IAFA, DBS description templates are created as an hierarchy. We made use of the object oriented approaches to define the categories and subcategories. This is a natural way of creating hierarchy of categories and their attributes.

Base Level

At this level we have the attributes that are shared by all of the resources.

- Title: The title of the resource
- Language: The natural language of the resource
- URL: The uniform resource locator for the resource
- Description: Description of the resource in terms of description predicates
- Keywords: Keyword for the resource

We are aware that there are still problems with the base level. For example, attribute "Language" might not make sense for every resource bookmarked, such as images.

You might observe that we have "Description" and "Keywords" attributes to describe the resource. Description templates also describe the resource. So what's happening here? "Description" and "Keywords" attributes provide a detailed description of the resource whereas, the description templates provide a category for that resource. We will talk more about "Description" attribute in Section 4.4.

There are 6 top level categories which inherits from this base level at the moment: "Organization", "Document", "Personal Homepage", "Service", "Product", and finally "Message".

Organization

This category defines the common attributes for "Organization" type of resources. Available attributes for this category are:

- Postal-address: Street address of the organization
- City: In which city
- State: In which state

- Country: In which country the organization is located
- Contact-email: e-mail
- Phone: Phone number
- Fax: Fax number for contacting to the organization.
- Locating-resource: Which of the methods, search, directory structure, or unstructured listing, is used by the organization to provide the resources to the public.

You may think that some of these attributes are no use, such as the "Postal-address", and will never filled by the users. In fact, most of these were not filled for our initial set of resources.

Some of the organizations provide resources as a list or URLs available at the organizations' Web site. They don't really believe in the power of Web, or their Web sites are not large. But most of the time resources are structured into directories (categories). After the Web site becomes really big, organizations' Web sites provide search mechanisms for their visitors.

We have 7 subcategories inherited from "Organization" at the moment:

- University
- Commercial
- Non-profit
- Institute
- Government
- Military
- Other

No additional attributes are required to describe these subcategories at this moment. But we might add attributes like "Research-topics", "Departments", "Products", etc.

"Other" subcategory is used whenever the organization cannot fit in one of the about subcategories.

Document

This category defines the common attributes for "Document" type of resources. Available attributes for this category are:

- Author(s): Author(s) of the document
- Title: Title of the document
- Year: Year of publication
- Citation: Citation information for the document

The non-trivial "Citation" attribute provides a formal definition of the document. This is useful if the document retrieved will be used as a reference in some report.

We have 10 subcategories inherited from "Document" at the moment:

- Technical Report: This resource has one additional attribute: "Institution"
- Journal Article: This resource has one additional attribute: "Journal-name"
- Conference Proceeding: This resource has one additional attribute: "Conference name"
- Book: This resource has one additional attribute: "Publisher"
- Ph.D. Thesis: This resource has one additional attribute: "School"
- M.S. Thesis: This resource has one additional attribute: "School"
- Manual:
- Tutorial:
- FAQ:
- Other:

Product

This category defines the common attributes for "Product" type of resources. Available attribute for this category are:

- Producer: Who produced the product.

We have 5 subcategories inherited from "Product".

- Software
- Hardware
- Image
- Sound
- Video

"Software" subcategories might have additional attributes like "Version-No", but this is not really required.

Service

This category is used when the resource provide a service to the public. This service can be a dictionary server, or a list of collected links. Service has the following attribute:

- Locating-resource: Which of the methods, search, directory structure, or unstructured listing is used by the service to provide the resources to the public.

This attribute might not make sense when the service is a chat server.

Personal Homepage

This category is used when the resource is a personal homepage. It doesn't have any additional attributes. But we might add "Person-Name" as an attribute. The "Description" attribute of "Base Level" can be used for this purpose.

Message

This category is used when the resource is a mail message. It can be a mail sent to a list server. Some of the mail servers provide a Web interface to query the archives of the server. It can be used to bookmark USENET news also, provided that the message is located on the USENET server, such as Dejanews.[1]

[1]http://www.dejanews.com

4.4 Resource Description Predicates

"Description" and "Keywords" are the distinguishing attributes for the resources. If two people submit a resource, almost all the information provided by these people will be identical, but the "Description" and "Keywords" will be different. That is because the same resource will provide different information for these two people.

As we have said before, we checked bookmarks of people. Since these people didn't provide a description for the URLs, we checked (visited) the bookmarks for a description. Whenever possible we used the HTML source of the resource to find a description.

Some Web publishers provide a description for the resource in the HTML document itself. This meta information, enclosed in "META" tag, provides description of the resource and keywords for the search engines.

Unfortunately, the descriptions were in natural language and they could not be used in DBS as it can be seen in Section 4.2.1.

We observe that some of the words appear in more than one description.

For example, the following is taken from Web Diner Web server:[2]

```
<TITLE>Gif Animation Tips from the Web Diner</TITLE>

<META name="description" content="Web Diner tutorial to help you
easily create cool gif animations using shareware tools. Add motion to
your web site - it's simple!">

<META name="keywords" content="cool, gif, animation, motion, movement,
download, easy, audio, sound, cool, wav, midi, free, inspiration,
stories, instructions, help, tips, begin, beginning, beginner,
template, tutorials, help, links, add, URL, Web Diner, collection,
learn, web, sites">
```

Another example, this time it is taken from 22nd National Information Systems Security Conference Homepage:[3]

```
<TITLE>22nd National Information Systems Security Conference
Page</TITLE>
```

[2]http://www.webdiner.com/annexe/gif89/snowstp1.htm
[3]http://csrc.nist.gov/nissc

```
<META NAME="keywords" CONTENT="nist conferences, ncsc conference, info
security conferences, information security conferences, infosec
conferences, national information systems security conference, nissc,
niss conference, information security, information systems security,
infosec, nist, national institute of standards and technology, ncsc,
national computer security center">

<META NAME="description" CONTENT="The annual National Information
Systems Security Conference, co-hosted by the National Institute of
Standards and Technology and the National Computer Security Center, is
the premier conference for information security.  It provides a unique
international forum for discusions, tutorials, demonstrations,
information sharing, and networking. The co-hosts use this conference
to make public the INFOSEC message.">
```

In the above examples, the resources *provide* some kind of service through links to other site, or through tutorials. What is provided by these descriptions is the description of the resource, considering the Web site is there to provide some resource for the people. What we have to do is to name the provided resources or services by resource.

DBS uses description predicates to describe the resource by means of a predicate, and an argument to that predicate. The initial set of predicates are so general that they can be used to describe any resource without much trouble.

The following predicates can be used to categorize the resource:

- article_on( )
- book_on( )
- column_on( )
- papers_on( )
- news_on( )
- tutorial_on( )
- university_of( )
- institute_of( )
- journal_of( )

The following predicates can be used to describe the services provided by the resource:

- examples_on( )
- free( )
- index_to( )
- information_on( )
- link_to( )
- resource_for( )
- site_for( )
- source_for( )
- source_of( )
- source_on( )

The following predicates can be used to describe a product:

- creation_of( )
- development_of( )
- download( )

The following can be used to provide a specific description about the resource:

- person( )
- protect_from( )

Finally, the following can be used to describe any other things:

- fact( )

The best thing about description predicates is combining different "Descriptions" of the same resource. All you have to do is to take the union of the set of predicates describing the resource. Same thing can be done for the "Keywords" also.

Appendix C provides more examples on describing resources.

4.5 Bookmark Quality

When a new resource is submitted to the DBS system, the system assigns a default value to all of the predicates and keywords that describe the resource. The value of the predicates and keywords value change when

- The resource is submitted more than once by one or more users.
- When the user provides a feedback for that resource.

Query results are sorted by the value of the resource for that query. The user gets the more valuable resource on the top of results list.

In Information Retrieval (IR), user feedback is important, because it can be used to modify the query and return more useful documents. IR systems can get the feedback implicitly or explicitly as follows:

- Retrieval results can be considered as relevant. This is the weakest criterion we can use.
- After reading the description of a match, the user decides to retrieve the whole document. This is a stronger criterion. But we still don't know if the document is useful or not.
- Finally the user gives a full feedback for the retrieved document. The feedback can be positive or negative: people may want to give negative feedback to show their disappointments. This is the strongest criterion. But not all users are willing to share their feedback that with others.

All of the above should be used, in some combination, to find the actual relevance.

DBS system explicitly asks for a feedback from the user.

4.6 User Trust Level

DBS is a distributed information sharing service. We think that not all the bookmark submitters are at the same trust level. This means that you may or may not trust every submitted resource depending on who submitted it. This is related to her/his experience on the topic.

Every new user has a default trust value. This initial value changes whenever the user uses the DBS system. We have the following information about a given user:

- Submissions to the system: Experts in a subject area tend to submit bookmarks in their subject. So descriptions of submitted queries and number of submissions are good parameters to calculate user trust level.
- Queries to the system: Their queries will be on the same subject also. Because they want to see if any new resources available on the topic, and they want to access their own bookmarks on the server.
- Feedbacks to the system: Their feedback will be consistent for the same query and the same resource.

Chapter 5

Implementation

Our prototype implementation consists of two separate applications, a stand-alone DBS client (DBSClient) and a DBS server (DBSServer). DBSClient provides the GUI to submit, to query resources, and to display the query results. DBSServer provides all the functionality to store the resources, to evaluate resource quality and user trust level.

5.1 DBSClient

The DBSClient GUI is used to submit resources to the DBSServer, to query available resources at the server, to display the query results, and to provide user feedback to the server.

The DBSClient is implemented as a Java Applet which makes it to be portable to almost any operating environments.

Figure 5.1 shows the main window of DBSClient.

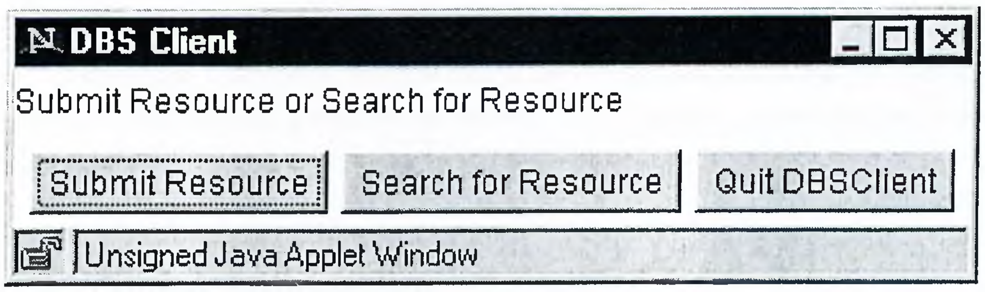

DBS Client
Submit Resource or Search for Resource
Submit Resource
Search for Resource
Quit DBSClient
Unsigned Java Applet Window

Figure 5.1: DBSClient Main Window

5.1.1 Resource Submission

The DBSClient implements a resource submission wizard which simplifies template filling (based on description templates) and resource description (based on description predicates and keywords).

- The user enters the base attributes such as "URL", "Title", etc. Figure 5.2 shows attributes that are common to all resources.
- Then the user assigns a resource category, one of "Organization", "Document", "Personal Homepage", "Service", "Product", and "Message".
- GUI asks for other required attributes for that category.
- After that the user enters sub-category and other information about the resource, the DBSClient asks for a description (using description predicates and keywords) of the resource.
- DBSClient sends the filled template and the user name to the DBSServer for further processing and storage.

Description by Keywords

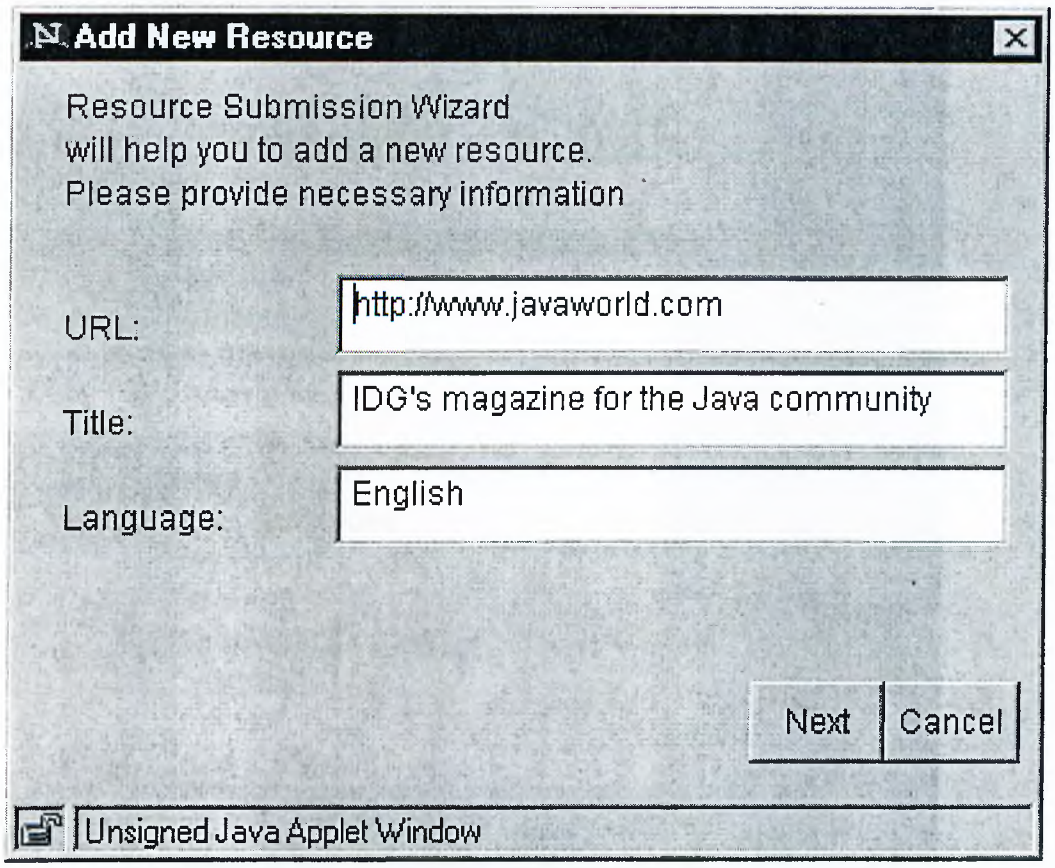
Figure 5.3 shows the Keywords Window that is used to add keyword to the description of the resource. The user can add as many keywords as needed. S/he can automatically add any keywords that are defined in the resource as "META" tags. "Retrieve" button is used for this purpose. It asks the server to download the requested resource and extract the "META" keywords from the resource. The user then analyzes the returned list to match her/his description.

Description by Predicates

Figure 5.4 shows the Description Window that is used to add description based on predicates. The user can add as many predicates as needed. S/he can also create new predicates for now and future use.

5.1.2 Query Submission

The Query Window is almost the same as resource Description Window. But this time it forces the user to search with available predicates and to restrict the question formulation.


Add New Resource

Resource Submission Wizard
will help you to add a new resource.
Please provide necessary information

URL: http://www.javaworld.com

Title: IDG's magazine for the Java community

Language: English

Figure 5.2: Base Level Attributes

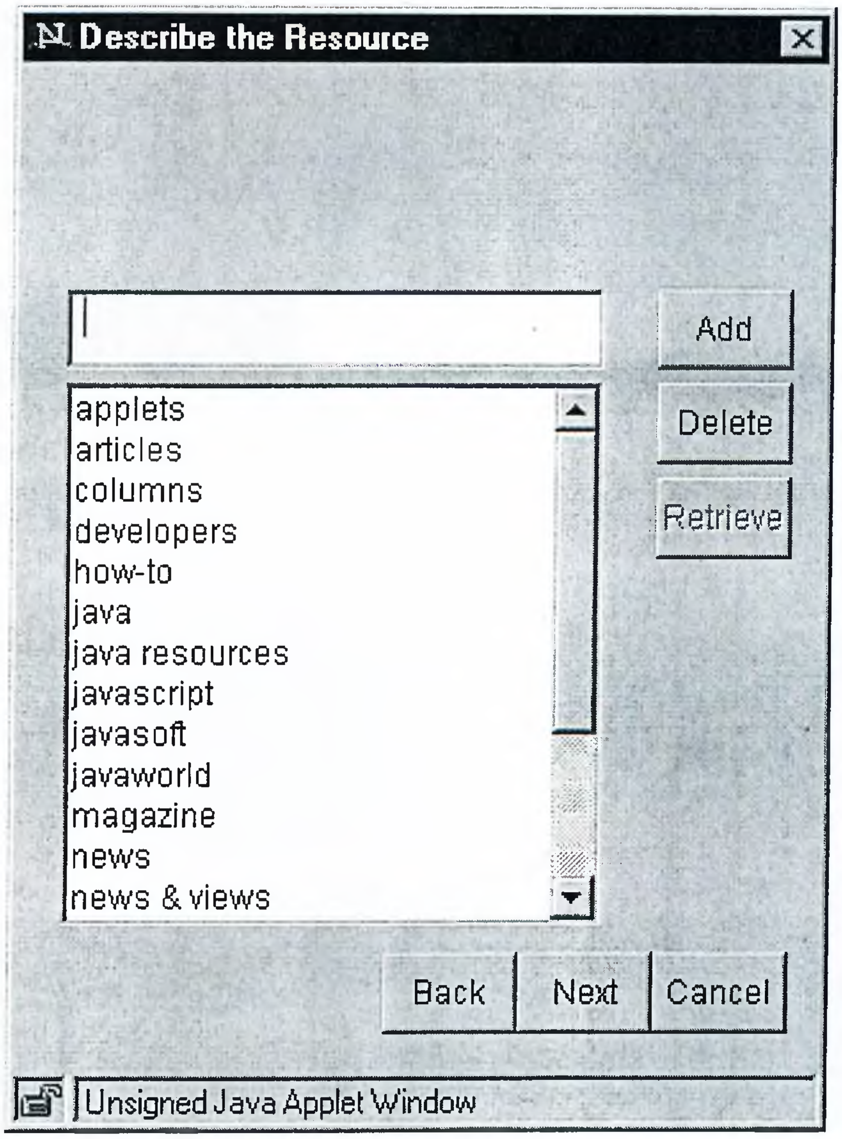
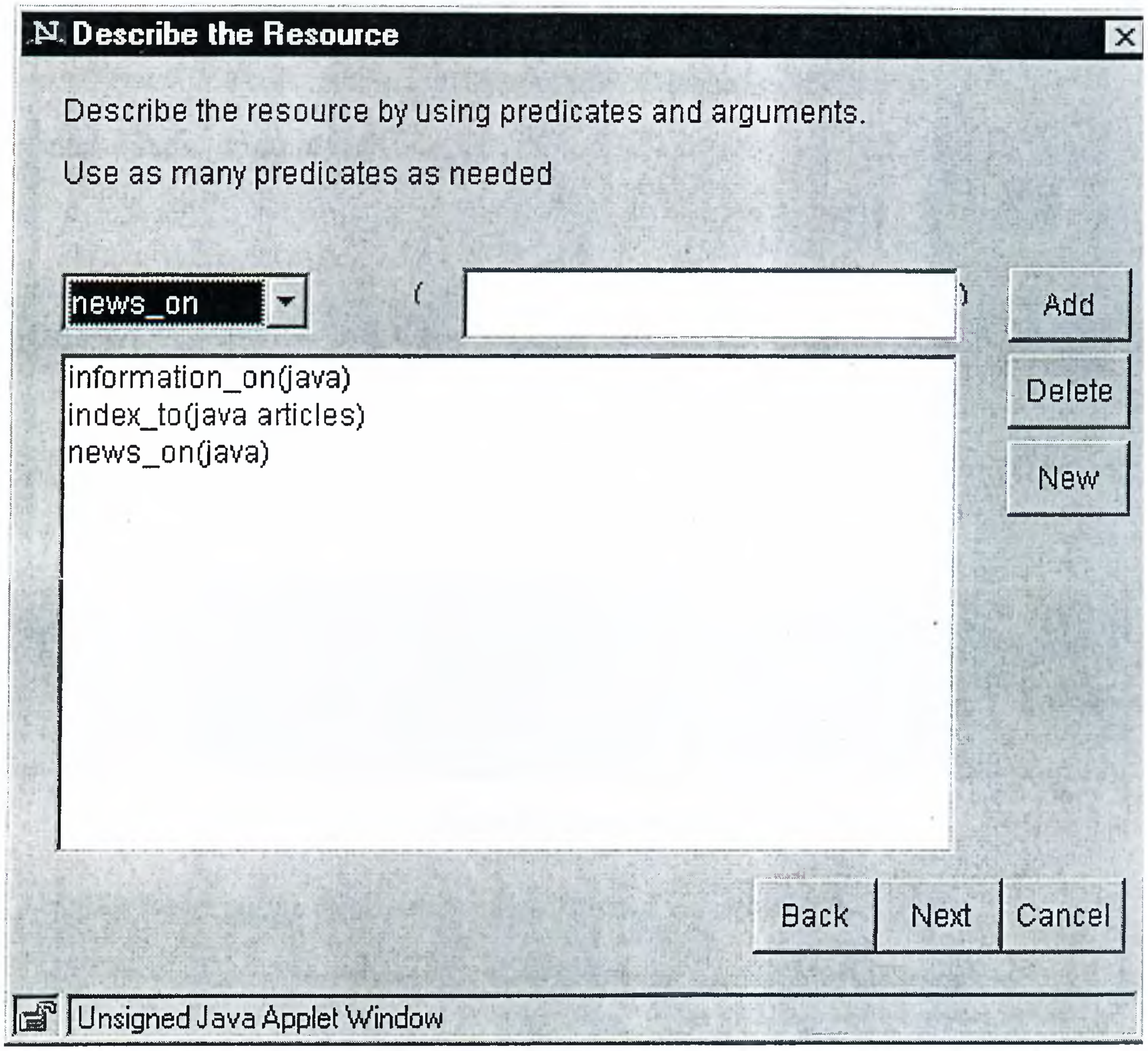


Figure 5.3: Keywords


Describe the Resource

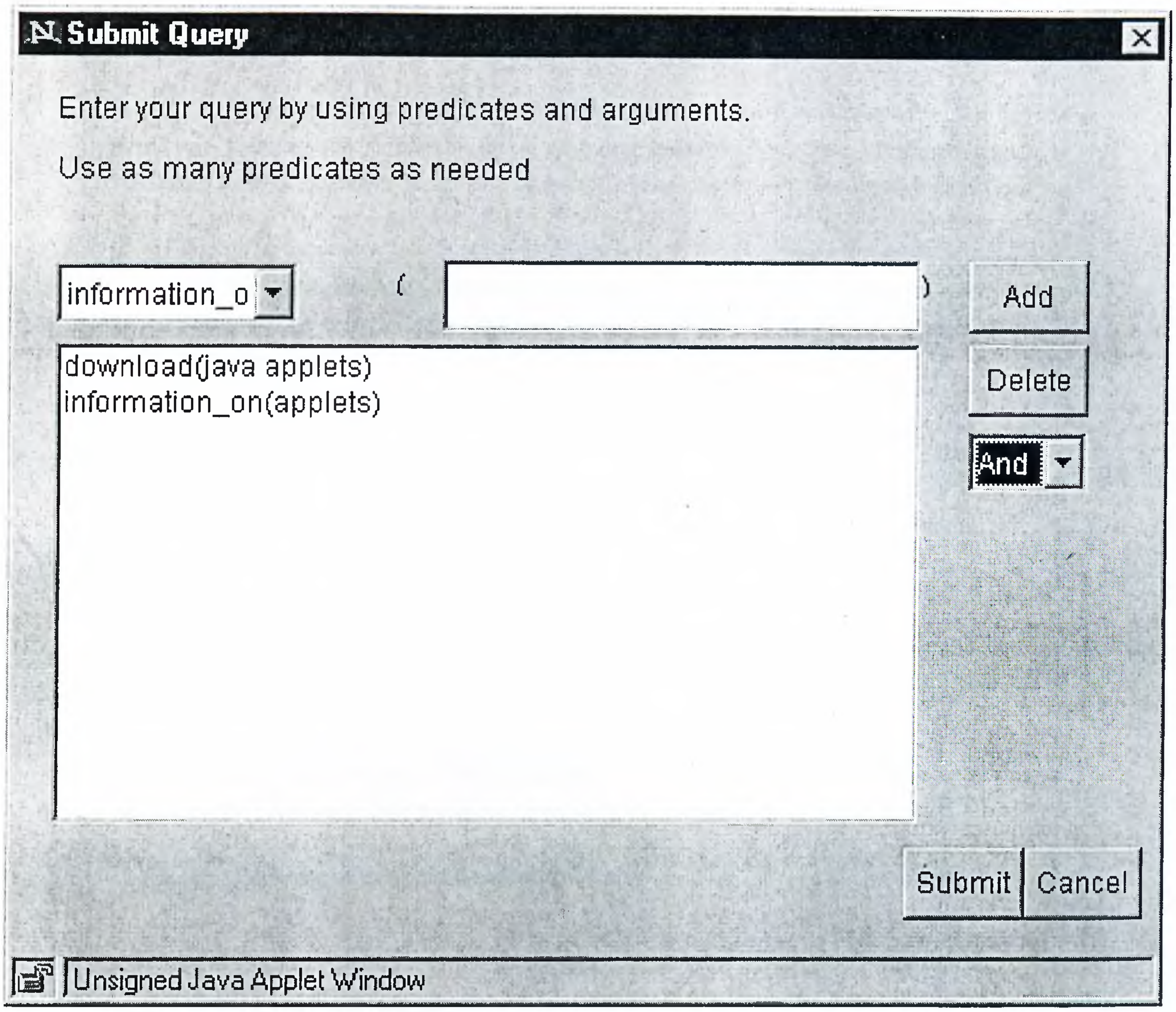
Describe the resource by using predicates and arguments.

Use as many predicates as needed

news_on ( )

information_on(java)
index_to(java articles)
news_on(java)

Figure 5.4: Resource Description


Submit Query

Enter your query by using predicates and arguments.

Use as many predicates as needed

information_o ( )

```
download(java applets)
information_on(applets)
```

Figure 5.5: Query Window

This doesn't restrict possible queries types. The user can still ask for any resource.

To formulate a query the user can use as many terms as needed. The query is conjunction ("And") or disjunction ("Or") of all query terms. In the first case, the description of the resource must contain all of the term in the query. In the other case, at least one match is sufficient.

Figure 5.5 shows the query window which uses predicates.

5.1.3 Query Results and Feedback

Result of the query, i.e. the resources that match to the query, is displayed in the Results Window. This window shows the list of matching resources and basic information such as URL and title of the resources. The scores of the resources are also displayed. Based on their scores the resources list is sorted.

The Results Window has a button to retrieve the resource via the active browser. It also has a button to submit the user feedback for the resources to the server. The feedback is one of the following:

- Irrelevant
- Not so good
- Good
- Very good
- Excellent

The Results Window has a "Modify Query" button. The user can use this to return to the Query Window and have a chance to modify the query.

Figure 5.6 shows an example Results Window.

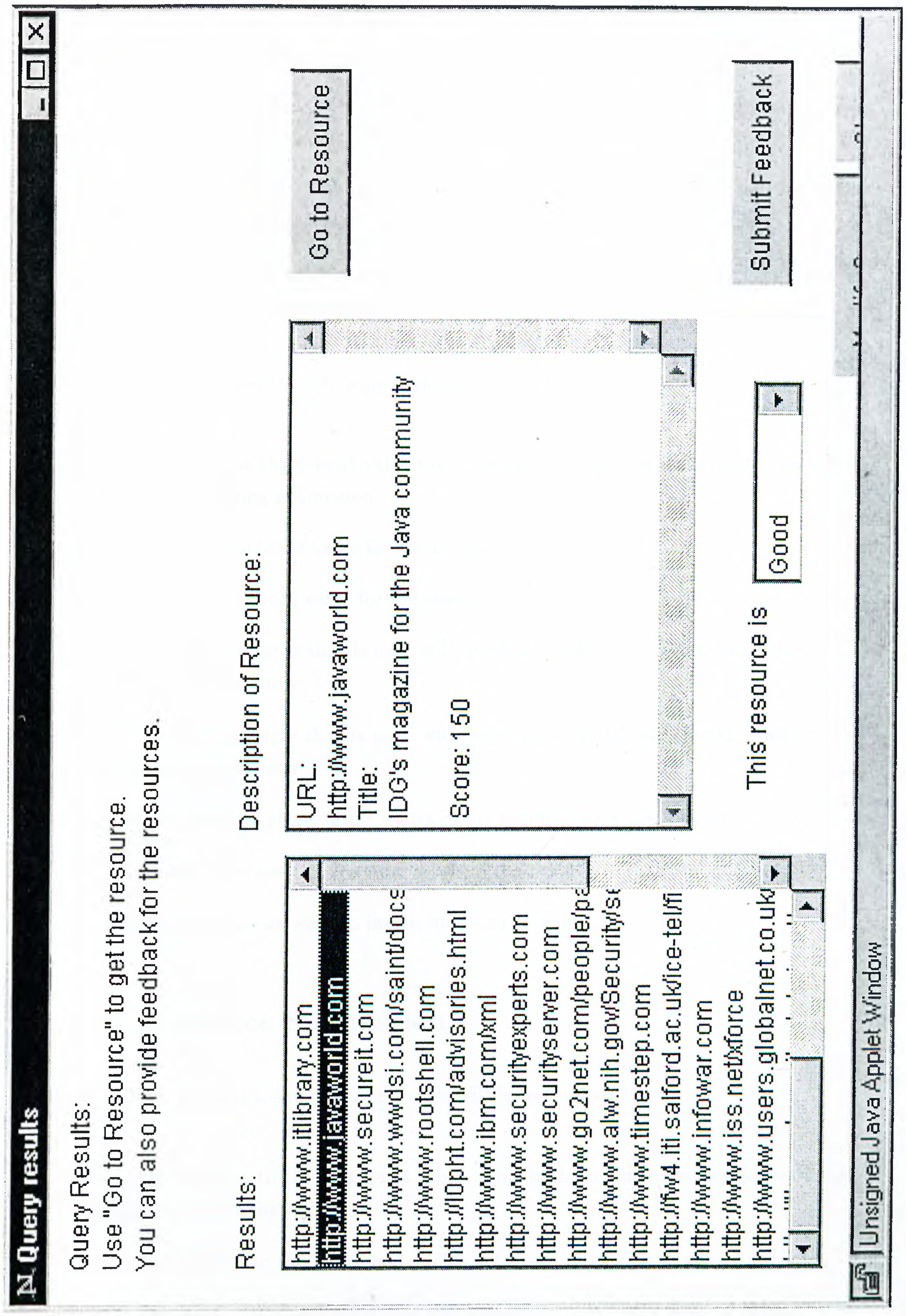
5.2 DBSServer

The DBSServer is used to store resources in a database, process the queries, send the results back to the client, and to retrieve user feedback from the client.

The DBSServer is implemented in C using BSD socket interface, and linked with gdbm, GNU Database Management System. Both DBSServer and gdbm library can be compiled and run on almost any Unix systems.

We defined the following tables using gdbm:

- Resource table: This table stores all the information, including the description and keywords, about the resources. Each entry in this table is of the form <URL, AttrValues>.
- User table: This table stores the URL submitted by the users. Each entry is of the form <User, URL-List>.



Query Results:
Use "Go to Resource" to get the resource.
You can also provide feedback for the resources.

Results:

http://www.itlibrary.com
http://www.javaworld.com
http://www.secureit.com
http://www.wwdsi.com/saint/docs
http://www.rootshell.com
http://l0pht.com/advisories.html
http://www.ibm.com/xml
http://www.securityexperts.com
http://www.securityserver.com
http://www.go2net.com/people/pa
http://www.alw.nih.gov/Security/s
http://www.timestep.com
http://fw4.iti.salford.ac.uk/ice-tel/fi
http://www.infowar.com
http://www.iss.net/xforce
http://www.users.globalnet.co.uk

Description of Resource:

URL:
http://www.javaworld.com
Title:
IDG's magazine for the Java community

Score: 150

This resource is Good

Figure 5.6: Results Window

- Predicate index table: This table stores the index for predicates. Each entry is of the form <Predicate, URL-List>.
- Predicate values table: This table stores the values of predicates for a given resource. Each entry is of the form <(URL, Predicate), Value>.
- Keyword index table: This table stores the index for Keywords. Each entry is of the form <Keyword, URL-List>.
- Keyword values table: This table stores the values of Keywords for a given resource. Each entry is of the form <(URL, Keyword), Value>.

We have also defined the following parameters/variables:

- ADD_VALUE is the default value that is assigned to the predicates and keywords of the resource during submission.
- $pValue$ is the predicate value for the resource
- $kValue$ is the keyword value for the resource
- K is constant multiply that is used with predicate value to calculate the value of the resource for that query
- L is constant multiply that is used with keyword value to calculate the value of the resource for that query
- $VPred$ is the total of predicate values of the resource for a given query
- $VKeyword$ is the total of keyword values of the resource for a given query
- bm is the resource we want to insert, query, or promote.

5.2.1 Resource Submission

When the user submits a new resource to the DBSServer, the server

- adds the resource to the Resources table. This is the main repository for retrieving everything about the resource.
- adds the resource to the Users table. This is to keep track of who submitted what.
- adds indexes for the resource using the description predicates and keywords. The indexes are used to retrieve the relevant resource at the time of query processing.

- generates new keywords from the description predicates. The arguments to predicates are good candidates to be keywords. These new keywords are also indexed.
- adds the default values for description predicates and keywords of the resource. We have a unique value for every predicate/keyword of every resource. These values will be used to find the quality of the bookmark at query processing time.

If the resource doesn't exist in the repository, then it is added without any processing of the description. The default value ADD_VALUE is assigned to all the predicates and the keywords of the resource as shown below:

```
**foreach** pred **in** bm.pred
    pValue = $ADD\_VALUE$
**foreach** keyword **in** bm.keywords
    kValue = $ADD\_VALUE$
```

If the resource already exists in the repository, then we add previous value of the predicates and keywords to ADD_VALUE as shown below:

```
**foreach** pred **in** bm.pred
    pValue = $\text{pValue}_{prev} + ADD\_VALUE$
**foreach** keyword **in** bm.keywords
    kValue = $\text{kValue}_{prev} + ADD\_VALUE$
```

In this case, the description and keywords for the resources merged using set union operation.

$\text{RDesc} = \text{RDesc}_{prev} + \text{bm.description}$
$\text{RKeywords} = \text{RKeywords}_{prev} + \text{bm.keywords}$

Full algorithm for resource submission is given in Appendix A. An example submission is given in Appendix B.

The server uses the Porter stemmer [FBY92] to find the stems of derived words just before indexing. It is used to increase the recall of the query.

5.2.2 Query Processing

When the user queries the DBSServer, the server

- retrieves relevant resources for each of the predicates (and their arguments) by using the indexes available on the predicates and the keywords.
- calculates the intersection or union (depending on whether query is a conjunction or disjunction of terms) of these list of resources.
- evaluates quality of the resources by means of predicate and keyword values for that resource.
- sorts the resulting values with resources
- transfers the result to the DBSClient

The DBSServer not only searches for a match on the predicate, but also searches for the argument (keyword) of the predicate. If there is a match, the keywords value is also added to the resource value. But the predicate value is still dominant.

```
foreach Resource in Resource-List
    Resource.value = $K$ x $Vpred + L$ x $Vkeyword$
```

Full algorithm for query processing is given in Appendix A. An example query is given in Appendix B.

5.2.3 Query Results and Feedback

After the query is processed by the server, the result (list of matching resources) is send back to the DBSClient. There is no more to do at this step.

The user can send feedback for any of the resources (one at a time) at this time. The DBSServer updates the predicate value and keyword value for that resource with this feedback.

This will either promote the resource for the same query if the feedback is positive, or will demote the resource if the feedback is negative.

In fact, description predicates and keywords are those whose value changes. The same resource results a different value for a different query. However, for the same query only the order of the resources may change.

```
foreach pred in qs.qterms
    $pValue = pValue_{prev} + Rating$
    $kValue = kValue_{prev} + Rating$
```

Full algorithm for promotion/demotion is given in Appendix A. An example feedback is given in Appendix B.

Chapter 6

Tests and Observations

We have installed DBSClient and DBSServer to Bilkent University CEIS Department Web server and to Havelsan, Inc. Information Security Department Web server. The following sections present the initial states of these two installations and their current status.

6.1 Installations and Tests

We think that the interests of people at Bilkent and at Havelsan are different. For that reason we sat up custom state for both of these sites. By "state" we mean initial set of resources, initial set of predicates, and default scores for predicates and keywords.

6.1.1 Initial Status

The setup at Bilkent contains the following:

- Before going public, we added 30 resources all of which were about Java. We thought that more people are interested in Java than any other topic.
- We used the following predicates to describe these 30 resources. This list is also the initial set of predicates.
  - article_on()
  - column_on()
  - development_of()

  - download()
  - fact()
  - free()
  - index_to()
  - information_on()
  - link_to()
  - papers_on()
  - site_for()
  - source_on()

- All the predicates and keywords that describe the resources are given *ADD_VALUE* by default.

The setup at Havelsan contains the following:

- Before going public, we added 30 resources all of which were about computer and Internet security.
- We used the following predicates to describe these 30 resources. This list is also the initial set of predicates.
  - article_on()
  - development_of()
  - fact()
  - index_to()
  - information_on()
  - journal_of()
  - leader_of()
  - link_to()
  - news_on()
  - papers_on()
  - site_for()
  - source_for()
  - source_on()
- All the predicates and keywords that describe the resources are given *ADD_VALUE* by default.

After this set up, we informed the people at Bilkent and Havelsan about Distributed Bookmark Sharing, and DBSClient. The DBSServers are still running on these Web servers.

The next section presents the current state in DBSServers.

6.1.2 Current Status

Current state of DBSSevers are as follows:

At Bilkent, people added 20 new resources, which makes a total of 50. We didn't enforce any restrictions on topics, so these 20 new resources are about Internet, parallel programming, network programming, and softwares.

At Havelsan, people added 30 new resources. This time we have new resources about security, as well as on new topics such as XML, Linux, viruses, and Java.

For better descriptions people at Bilkent added the following new predicates to the set of existing predicates:

- book_on()
- column_on()
- creation_of()
- download()
- examples_on()
- homepage_of()
- institute_of()
- manual_for()
- organization()
- person()
- report_on()
- tutorial_for()
- university_of()

New predicates at Havelsan site are as follows:

- bug()
- download()
- examples_on()
- files_related_to()
- free()
- list_of()
- messages_on()
- product()
- projects_about()
- protect_from()
- provide()
- software()
- tutorial_for()
- university_of()

People at Bilkent have submitted 60 queries to DBSServer. The topics of these queries are Internet and Java, parallel programming, network programming, and free softwares.

People at Havelsan have submitted 20 queries. Most of the topics are on Java and Linux. There is less number of queries on Firewalls.

Next section discusses the differences and improvements between the initial set up and current state of DBSServer.

6.2 Observations

First of all, we got several messages regarding the DBS system. We think that people are happy with the system, but it still needs some improvements. We will talk about them later in this report.

The following are the observations we have at the end of test period:

- The first thing we observe is, people submitted resources which are related to their interest area. This is what we have suggested in Section 4.6.
- On the other hand, people have searched for resources in their interest area. This is another thing that was suggested in 4.6. This might be because of the need for new (fresh) information in that area.
- Initial set of predicates were so general that, people created new and more specific ones for better descriptions. Some of these predicates are "examples_on()", "files_related_to()", "projects_about()", and "protect_from()".
- Some of the new predicates are related to Description Templates. Example of such predicates are "messages_on()", "organization()", "tutorial_for()", "software()".
- The newer predicates are used at least ones while describing resources. That is what we expect from the users. But they are not used while querying the system. Queries were formulated more general predicates, such as "information_on()" or "article_on()".

Chapter 7

Conclusion and Future Work

In this thesis, we have discussed four different ways of resource discovery on the Web and have suggested improvements to overcome problems some of resource discovery problems.

Using shared bookmark managers such as Webtagger [Kea97a] or automatic bookmark organizers such as Bookmark Organizer [MS96] was one possible solution.

Another possible solution was using distributed indexing systems such as WHOIS++ [Dea95, BFS96], What'sHot [Kos97, Kos96] or Harvest [Har96].

We proposed Distributed Bookmark Sharing as a new approach to resource discovery problems.

7.1 Conclusion

We have defined DBS, a hybrid of distributed bookmark manager and search tool.

For the thesis we searched bookmark sharing primitives, i.e., simple and efficient ways of describing bookmarks, and also ways to calculate bookmark quality. We used resource description templates and resource description predicates to describe the resources. We used predicate values of the resources with submitters trust level and expertise to calculate the quality of the bookmark for that query and resource.

We implemented parts of a DBS system, a stand-alone DBS server and a DBS client to test description templates and other primitives. We installed the systems to two locations and asked people to use the system.

We think that there is still work to be done.

We can further define new projects and enhancements to DBS system as described next.

7.2 DBS Improvements

DBSClient and DBSServer can be improved as listed below:

- Better and intelligent user interface,
- Adding relevant predicates automatically to the description as necessary (from description templates to description predicates),
- Allowing browsing in the resource repository,
- Extracting keywords both from the META tag and the body of the resource,
- Indexing the resource at the server site, and store the resource for future use,
- Updating the index of the resources periodically,
- Another query interface which will search the keywords in the above indexes.

7.3 Distributed Bookmark Indexing

DBS system can be setup for a special topic. For example, DBS server at Bilkent CEIS Department stores only Java resources, and DBS server at Havelsan, Inc. stores security resources.

Such DBS systems can give birth to the Distributed Bookmark Indexing (DBI) System. This system will contain a federation of cooperative DBS systems. So a DBI system will contain virtually any topic.

DBI system will be an hierarchical tree, just like the USENET groups. DBS systems will be replaced by distributed topic servers, and those topic servers will be linked to a higher more general topic server, and so forth, until we have the root topic server on the top.

A resource that is submitted to local DBI server will be routed to the most proper server based of the description of the resource. A query will be forwarded to the most proper server also. Since the DBI servers are specialized, this resource/query forwarding will not be a problem. Effective resource description methods from DBS system will help DBI while combining descriptions from two or more people.

Appendix A

DBS Server Algorithms

Resource Submission

```
addNewResource(resource r)
begin
    bookmark bm;
    bm = parseResource(r);
    addResource(bm,r);
    addPrediates(bm);
    addKeywords(bm);
end

addResource(bookmark bm, resource r)
begin
    if exists <bm.url, X> in ResourceTable then
        newResource= mergeDescriptions(X,r);
        add <bm.url, newResource> into ResourceTable
    else
        add <bm.url, r> into ResourceTable
    if exists <bm.submitter, X> in UserTable then
        newRList= mergeURLs(X,bm.url);
        add <bm.submitter, newRList> into ResourceTable
    else
        add <bm.submitter, bm.url> into ResourceTable
end

addPredicates(bookmark bm)
begin
```

```
    foreach pred in bm.pred do
        if exists <(bm.url, pred), Value> in PredTablethen
            newValue = Value + ADD_VALUE;
            add <(bm.url, pred), newValue> into PredTable
        else
            add <(bm.url, pred), ADD_VALUE> into PredTable
        if exists <pred, X> in PredIndexTablethen
            newRList = mergeURLs(x, bm.url);
            add <pred, bm.url> into PredIndexTable
        else
            add <pred, bm.url> into PredIndexTable
end

addKeywords(bookmark bm)
begin
    foreach keyword in bm.keywords do
        if exists <(bm.url, keyword), Value> in KeywordTablethen
            newValue = Value + ADD_VALUE;
            add <(bm.url, keyword), newValue> into KeywordTable
        else
            add <(bm.url, keyword), ADD_VALUE> into KeywordTable
        if exists <keyword, X> in KeywordIndexTablethen
            newRList = mergeURLs(x, bm.url);
            add <keyword, bm.url> into KeywordIndexTable
        else
            add <keyword, bm.url> into KeywordIndexTable
end
```

Query Processing

```
processQuery(query q)
begin
    qs = parseQuery(q);
    URL-List = null;
    foreach pred in qs.qterms do
        find resources that match <pred, URLs> in PredTable
        add URLs to URL-List
        extract keyword form qterm
        find resources that match <keyword, URLs> in KeywordsTable
        add URLs to URL-List
    foreach URL in URL-List do
        VResource = K x Vpred + L x Vkeyword
    sort URL-List using VResource
    return sorted URL-List
end
```

Feedback

```
promote(URL url, query q, int rating)
begin
    qs = parseQuery(q);
    foreach pred in qs.qterms do
        find <(url, pred), Value> in PredValueTable
        newValue = Value + rating;
        add <(url, pred), newValue> into PredValueTable
        find <(url, keyword), Value> in KeywordValueTable
        newValue = Value + rating;
        add <(url, keyword), newValue> into KeywordValueTable
end
```

Appendix B

Bookmark Submissions

This section provides examples on how bookmark submission works in practice:

Let's consider the following resource first:

URL	http://www.alw.nih.gov/Security/security-docs.html

Predicates	article_on(computer security)
	article_on(security)
	link_to(security articles)
	papers_on(security)
	information_on(computer security)
	papers_on(computer security)

Keywords	security computer security security articles network security

Adding this resource to DBS system updates Predicate index table, keyword index table, predicate values table, and keyword values table accordingly:

Predicate index table:

article_on(computer security)	NIH
article_on(security)	NIH
link_to(security articles)	NIH
papers_on(security)	NIH
information_on(computer security)	NIH
papers_on(computer security)	NIH

Keyword index table:

:

security	NIH
computer security	NIH
security articles	NIH
network security	NIH

Predicate Values:

NIH % article_on(computer security)	10
NIH % article_on(security)	10
NIH % link_to(security articles)	10
NIH % papers_on(security)	10
NIH % information_on(computer security)	10
NIH % papers_on(computer security)	10

Keyword Values:

NIH % security	10
NIH % computer security	10
NIH % security articles	10
NIH % network security	10

Now let's add the following resource next:

URL	http://l0pht.com/advisories.html

Predicates	fact(computer security)
	fact(security)
	fact(advisories)
	fact(vulnerabilities)
	article_on(vulnerabilities)
	article_on(computer security)

Keywords	advisories security computer security hackers vulnerabilities exploits crackers

DBS system modifies its tables accordingly:

Predicate index table:

article_on(computer security)	NIH, l0pht
article_on(security)	NIH
link_to(security articles)	NIH
papers_on(security)	NIH
information_on(computer security)	NIH
papers_on(computer security)	NIH
fact(computer security)	l0pht
fact(security)	l0pht
fact(advisories)	l0pht
fact(vulnerabilities)	l0pht
article_on(vulnerabilities)	l0pht

Keyword index table:

security	NIH, l0pht
computer security	NIH, l0pht
security articles	NIH
network security	NIH
advisories	l0pht
hackers	l0pht
vulnerabilities	l0pht
exploits	l0pht
crackers	l0pht

Predicate Values:

NIH % article_on(computer security)	10
NIH % article_on(security)	10
NIH % link_to(security articles)	10
NIH % papers_on(security)	10
NIH % Information_on(computer security)	10
NIH % papers_on(computer security)	10
l0pht % fact(computer security)	10
l0pht % fact(security)	10
l0pht % fact(advisories)	10
l0pht % fact(vulnerabilities)	10
l0pht % article_on(vulnerabilities)	10
l0pht % article_on(computer security)	10

Keyword Values:

NIH % security	10
NIH % computer security	10
NIH % security articles	10
NIH % network security	10
l0pht % advisories	10
l0pht % security	10
l0pht % computer security	10
l0pht % hackers	10
l0pht % vulnerabilities	10
l0pht % exploits	10
l0pht % crackers	10

Finally we add the following resource:

URL	http://www.alw.nih.gov/Security/security-docs.html

Predicates	link_to(security articles)
	papers_on(security)
	papers_on(computer security)
	site_for(computer security)
	site_for(java security)
	site_for(network security)
	site_for(unix security)
	site_for(Windows NT Security)

Keywords	unix security windows NT security java security network security security articles ActiveX security

Final status of DBS tables are as follows:

Predicate index table:

article_on(computer security)	NIH, l0pht
article_on(security)	NIH
link_to(security articles)	NIH
papers_on(security)	NIH
Information_on(computer security)	NIH
papers_on(computer security)	NIH
fact(computer security)	l0pht
fact(security)	l0pht
fact(advisories)	l0pht
fact(vulnerabilities)	l0pht
article_on(vulnerabilities)	l0pht
site_for(computer security)	NIH
site_for(network security)	NIH
site_for(Windows NT Security)	NIH
site_for(java security)	NIH
site_for(unix security)	NIH

Keyword index table:

security	NIH, l0pht
computer security	NIH, l0pht
security articles	NIH
network security	NIH
advisories	l0pht
hackers	l0pht
vulnerabilities	l0pht
exploits	l0pht
crackers	l0pht
unix security	NIH
windows NT security	NIH
java security	NIH
ActiveX security	NIH

Predicate Values:

NIH % article_on(computer security)	10
NIH % article_on(security)	10
NIH % link_to(security articles)	10 + 10
NIH % papers_on(security)	10 + 10
NIH % information_on(computer security)	10
NIH % papers_on(computer security)	10 + 10
l0pht % fact(computer security)	10
l0pht % fact(security)	10
l0pht % fact(advisories)	10
l0pht % fact(vulnerabilities)	10
l0pht % article_on(vulnerabilities)	10
l0pht % article_on(computer security)	10
NIH % site_for(computer security)	10
NIH % site_for(network security)	10
NIH % site_for(Windows NT Security)	10
NIH % site_for(java security)	10
NIH % site_for(unix security)	10

Keyword Values:

NIH % security	10
NIH % computer security	10
NIH % security articles	10 + 10
NIH % network security	10 + 10
l0pht % advisories	10
l0pht % security	10
l0pht % computer security	10
l0pht % hackers	10
l0pht % vulnerabilities	10
l0pht % exploits	10
l0pht % crackers	10
unix security	10
windows NT security	10
java security	10
activeX security	10

Query submission

Now we submit the following query to the system in previous examples:

information_on(computer security)

We get the following results back:

NHL	(10+10)*10+10*5=150
l0pht	10*5=50

Feedback

If NHL is excellent for information_on(computer security), and we provide a feedback, DBS system modifies its tables with this new information:

Predicate Values:

NIH % article_on(computer security)	10
NIH % article_on(security)	10
NIH % link_to(security articles)	20
NIH % papers_on(security)	20
NIH % information_on(computer security)	10 + 2
NIH % papers_on(computer security)	20
l0pht % fact(computer security)	10
l0pht % fact(security)	10
l0pht % fact(advisories)	10
l0pht % fact(vulnerabilities)	10
l0pht % article_on(vulnerabilities)	10
l0pht % article_on(computer security)	10
NIH % site_for(computer security)	10
NIH % site_for(network security)	10
NIH % site_for(Windows NT Security)	10
NIH % site_for(java security)	10
NIH % site_for(unix security)	10

Keyword Values:

NIH % security	10
NIH % computer security	10
NIH % security articles	20
NIH % network security	20
l0pht % advisories	10
l0pht % security	10
l0pht % computer security	10 + 2
l0pht % hackers	10
l0pht % vulnerabilities	10
l0pht % exploits	10
l0pht % crackers	10
unix security	10
windows NT security	10
java security	10
ActiveX security	10

Appendix C

Examples with HTML Meta Description Tags and User Resource Descriptions

From http://www.securityexperts.com

<title>The Security Experts</title>

<meta name="description" content="The Security Experts">

<META Name="keywords" Content="Security Consultants, Information Security, Penetration Testing, Tiger Team, Security Audits, Security Reviews, Audits, Security, Computer Security Consultants, Computer Security, Enterprise Security, Computer Consultants, Consultants, Information Warfare, Threat Analysis, Security Education, Security Training, Security Awareness, Vulnerability Studies, Penetration Tests, Intrusion Detection, Security Assessment, Network Architecture, Security Protection, Security Detection, Security Reaction, Risk Analysis, Contingency Planning, Liability Reviews, Business Impact Analysis, Secure Remote Access, Secure Transactions, Trade Secret Protection, Competitive Intelligence, Counter Surveillance, Enterprise Security, Security Architecture, Modelling/Integration, Emergency Response, Violation Monitoring, Evidence Protection, Audit Readiness, vulnerabilities, bugs, vulnerability database, hole, holes, exploits, vpns, virtual private network, VPN, penetration testing">

(taci) % (http://www.securityexperts.com/) % (The Security Experts) % (English) % (article_on(seucurity) % article_on(computer security) % article_on(network security)) % (vulnerability % vulnerability studies % security % security architecture % penetrations tests % vilolation monitoring % security integration % enterprice security % evidence protection % intrusion detection % security education % secure remote access % secure transactions % network scanning % information warfare % auditing) % (Organization) % () % () % () % () % () % () % () % (% directory % directory % listing % directory % listing) % (Commercial)

From http://www.javapowered.com

<TITLE>Java - Applets - Java Applets - javapowered.com</TITLE>

<META NAME="description" CONTENT="The Java Site fr Caffeinating you Web Page with Hundreds of Free Applet Downloads and previews."> <META NAME="keywords" CONTENT="Java, Applets, Java Applets, Free, Collections, Web Page Design, WebSite Design, Web Authoring, 100% Java">

(arcin) % (http://www.javapowered.com/) % (Java - applets - java applets - javapowered.com) % (English) % (free(java applets) % free(applications) % free(downloads) % download(java applets)) % (java % applets % java applets % web page design % webpage design % free) % (Organization) % () % () % () % (US) % () % () % () % (% directory) % (Non-Profit)

(kince) % (http://www.javapowered.com/) % (Java - applets - java applets - javapowered.com) % (English) % (free(java applets) % free(applications) % free(downloads) % download(java applets) % information_on(web page design)) % (java % applets % java applets % web page design % webpage design % free) % (Organization) % () % () % () % (US) % () % () % () % (% directory) % (Non-Profit)

Bibliography

[AD94] Chidanand Apte and Fred Damerau. Automated learning or decision rules for text categorization. *ACM Transactions on Information Retrieval Systems*, 12(3), 1994.

[Bec] David Beckett. Iafa templates in use as internet metadata. http://www.hensa.ac.uk/www/iafatools/paper/paper.html.

[BFS96] C. W. Bunyip, J. Fullton, and S.Spero. Rfc 1913: Architecture of the whois++ index service, February 1996.

[BW95] T. Boutell and L. Wirzenius. Linux software map, June 1995. http://siva.cshl.org/lsm/lsm/html.

[CL96] Jim Cowie and Wendy Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, January 1996.

[Cun97] Hamish Cunningham. Information extraction -a user guide. Technical Report CS-97-02, Institute for Language, Speech and Hearing (ILASH), February 1997.

[Dea95] Deutsch and et al. Rfc 1835: Architecture of the whois++ service, August 1995.

[ea97] Jerry R. Hobbs et al. Fastus: A cascaded finite-state transducer for extracting information from natural-language text. Technical report, Artificial Intelligence Center, SRI International, May 1997.

[FBY92] William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval Data Structures and Algorithms.* Prentie Hall, 1992.

[Har96] Darren R. Hardy. Harvest user's manual version 1.4. Technical Report CU-CS-743-94, University of Colorado at Boulder, 1996.

[HS96] Darren R. Hardy and Michael F. Schwartz. Customized information extraction as a basis for resource discovery. *ACM Transactions on Computer Systems*, 14(2):171–199, May 1996.

[HSF85] K. Harrenstein, M. Stahl, and E. Feinler. Rfc 954: Nicname/whois, October 1985.

[iWA98] itList Web Admin. itlist web site, 1998. http://www.itlist.com.

[Kea97a] Richard M. Keller and et al. A bookmarking service for organizing and sharing urls. *Computer Networks and ISDN Systems*, 29:1103–1114, 1997.

[Kea97b] Joseph A. Konstan and et al. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3), March 1997.

[Kos96] A. Kosmynin. An information broker for adaptive distributed resource discovery service. In *1st International Conference of Web Society (WebNet'96)*, 1996.

[Kos97] A. Kosmynin. From bookmark managers to distributed indexing: An evolutionary way to the next generation of search engine. *IEEE Communications Magazine*, 35(6):146–151, June 1997.

[KSS97] Henry Kautz, Bart Selman, and Mehul Shah. Referall web: Combining social networks and collaborative filtering. *Communications of the ACM*, 40(3), March 1997.

[LB96] Nancy J. Lightner and Indranil Bose. What is wrong with the world wide web?: A diagnosis of some problems and prescriptions of some remedies. *Ergonomics*, 39(8):995–1004, 1996.

[Lew92] David D. Lewis. *Representation and Learning in Information Retrieval.* PhD thesis, Department of Computer and Information Science University of Massachusetts, 1992.

[Mem98a] ROADS Project Members. Field descriptions for iafa templates, 1998. http://www.roads.lut.ac.uk/v1/IAFA-help.

[Mem98b] ROADS Project Members. Iafa template usage statistics, 1998. http://www.roads.lut.ac.uk/TemplateStatistics.

[MS96] Y. S. Maarek and Israel Z. B. Shaul. Automatically organizing bookmarks per contents. *Computer Networks and ISDN Systems*, 28:1321–1333, 1996.