RESOURCE AND KNOWLEDGE DISCOVERY FROM THE INTERNET AND MULTIMEDIA REPOSITORIES

by

Osmar Rachid Zaïane
B.Sc., Université de Tunis, Tunisia, 1988
D.E.A. (M.Sc.), Université Paris XI, France, 1989
M.Sc., Université Laval, Québec, Canada, 1992

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
March 1999

APPROVAL

Name: Osmar Rachid Zaïane

Degree: Doctor of Philosophy

Title of thesis: Resource and Knowledge Discovery from the Internet and Multimedia Repositories

Examining Committee: Dr. Ze-Nian Li
Chair

Dr. Jiawei Han
Senior Supervisor

Dr. Hassan Aït-Kaci
Supervisor

Dr. Veronica Dahl
SFU Examiner

Dr. Laks V.S. Lakshmanan
Concordia University
External Examiner

Date Approved:

Abstract

There is a massive increase of information available on electronic networks. This profusion of resources on the World-Wide Web gave rise to considerable interest in the research community. Traditional information retrieval techniques have been applied to the document collection on the Internet, and a panoply of search engines and tools have been proposed and implemented. However, the effectiveness of these tools is not satisfactory. None of them is capable of discovering knowledge from the Internet. The Web is still evolving at an alarming rate. In a recent report on the future of database research known as the Asilomar Report, it has been predicted that in ten years from now, the majority of human information will be available on the World-Wide Web, and it has been observed that the database research community has contributed little to the Web thus far.

In this work we propose a structure, called a Virtual Web View, on top of the existing Web. Through this virtual view, the Web appears more structured, and common database technology is applied. The construction and maintenance of this structure is scalable and does not necessitate the large bandwidth current search engines technologies require. A declarative query language for information retrieval and networked tool programming is proposed that takes advantage of this structure to discover resources as well as implicit knowledge buried in the World-Wide Web.

Large collections of multimedia objects are being gathered for a myriad of applications. The use of on-line images and video streams is becoming commonplace. The World-Wide Web, for instance, is a colossal aggregate of multimedia artifacts. However, finding pertinent multimedia objects in a large collection is a difficult task. Images and videos often convey even more information than the text documents in which they are contained. Data mining from such a multimedia corpus can lead to interesting discoveries.

We propose the extraction of visual descriptors from images and video sequences for content-based visual media retrieval, and the construction of multimedia data cubes which facilitate multiple dimensional analysis of multimedia data, and the mining of multiple kinds of knowledge, including summarization, classification, and association, in image and video databases.

To my father, my mother and my wife

The secret of all victory lies in the organization of the non-obvious.

Oswald Spengler

Acknowledgments

For contributions to the form and content of this thesis, my knowledge and my sanity, my first thanks are to my wife Jane without whom this work wouldn't have been presented as it is. I am really indebted to Jane for her everlasting understanding and making the last steps of writing this thesis enjoyable.

I am very grateful to my parents and my parents-in-law for their continuous moral support and encouragement. I hope I will make them proud of my achievements, as I am proud of them.

I wish to express my deep gratitude to my supervisor and mentor Jiawei Han. I thank him for his continuous encouragement, confidence and support, and for sharing with me his knowledge and love of this field. I am sure Jiawei will always be my mentor and an example for perseverance and hard work. As an advisor, he taught me practices and showed me directions I will use in my academic career.

I am very thankful to Hassan Aït-Kaci, my supervisor and friend, for his insightful comments and advice. He was always hearted and cheering when discussing my research. His observations and suggestions for improvement were very assuring, especially at the end of this endeavour.

My gratitude and appreciation also goes to Ze-Nian Li for the frequent enthusiastic and constructive discussions. He not only introduced me to the vision and image processing area, but also made me confident and eager to pursue more research in multimedia.

My deepest thanks to Laks Lakshmanan and Veronica Dahl for serving as examiners of my thesis.

I would also like to thank the many people in our department, support staff and faculty, for always being helpful over the years. A particular acknowledgment goes to Kersti, Carole, and sumo. I thank my friends and fellow students at Simon Fraser University for making the last years a memorable, rewarding and enriching experience.

Support for this research was initially funded by the Fonds pour la Formation de Cherchers et l'Aide à la Recherche (FCAR), and later by Simon Fraser University through a Graduate Fellowship. Additional support was made by Jiawei Han's Natural Sciences and Engineering Research Council of Canada (NSERC) Research Grants OGP0037230 and OGP36727 and his grants from the Canadian Network of Centres of Excellence (IRIS:HMI-5 and TL:NCE5.2).

Contents

List of Tables

List of Figures

Whatever you do will be insignificant, but it is very important that you do it.

MAHATMA GANDHI

Chapter 1

Introduction

In this thesis, we demonstrate the inefficiency and inadequacy of the current information retrieval technology applied on the Internet. We propose a framework, called Virtual Web Views, for intelligent interactive information retrieval and knowledge discovery from global information systems, and put forward a query language, WebML, for resource discovery and data mining from the Web using the virtual web views. We illustrate how descriptors collected for virtual web view building can be exploited for content-based image retrieval, and show how to carry out on-line analytical processing and data mining on visual data from the World-Wide Web, or other multimedia repositories.

More than 50 years ago, at a time when modern computers didn't yet exist, Vannevar Bush wrote about a multimedia digital library containing human collective knowledge, and filled with "trails" linking materials of the same topic[44]. At the end of World War II, Vannevar urged scientists to build such a knowledge store and make it useful, continuously extendible and more importantly, accessible for consultation. Today, the closest to the materialization of Vannevar's dream is the World-Wide Web hypertext and multimedia document collection. However, the ease of use and accessibility of the knowledge described by Vannevar is yet to be realized. Since the 1960s, extensive research has been accomplished in the information retrieval field, and free-text search was finally adopted by many text repository systems in the late 1980s. The advent of the World-Wide Web in the 1990s helped text search become routine as millions of users use search engines daily to pinpoint resources on the Internet. However, resource discovery on the Internet is still frustrating and sometimes even useless when simple keyword searches can convey hundreds of thousands of documents as results.

The dramatic drop in the price of storage devices and the advent of the World-Wide Web, an unprecedented information disseminator, are promoting the proliferation of massive collections of multimedia resources, either text documents, or images, or other media. Never has it been easier than with the World-Wide Web to publish all manner of digital documents and make them almost instantly available to everyone. However, given the monumental size of the collection, availability does not necessarily indicate universal accessibility or even visibility of the published artifacts. It is an extremely difficult task to find pertinent documents (either text, images, or other media) in this agglomerate. Finding relevant digital documents in a large collection is known as *Resource Discovery*. In a recent report on the future of database research [29] written by prominent authorities in database research, it has been foretold that the Web and other on-line data stores will hold the majority of published human knowledge. Despite the richness of this massive knowledge collection, the report underlines the challenges still ahead for the research community to produce methods for sorting out through this collection. The authors comment on the lack of considerable contribution from the database research community in the development of striking methods for the management and effective exploitation of the resources available on the Web.

In this thesis, we are concerned with resource discovery in the World-Wide Web context and in the context of large visual media collections, as well as the discovery of implicit information from those same collections. The information is implicit in the sense that it is not specifically stored in a clear and visible manner, but rather tacitly implied, in contrast to explicit information, which is apparent and certain, example, information conveyed by hyperlinks, properties of content components, or Web access behaviours.

The content of the dissertation is composed of two distinct parts. The first part covers resource discovery from the Internet and Web Mining. The makeup of a structure for the description of on-line artifacts and a query language that takes advantage of the structure are proposed. The second part pertains to resource and knowledge discovery from multimedia repositories. It deals specifically with visual artifacts like images and videos. This division is simply organizational to help the reader. Multimedia repositories are indeed an integral part of the Internet. In the second part of the thesis, however, visual media is treated as a particular example and is taken beyond the structure presented in part one.

In this work we pose the following theses:

Thesis 1 : (Web Overload) Search engines using Web crawling technology are the state of the art in resource discovery from the World-Wide Web. However, this technology is not scalable, is overloading the networks, and is not viable in the long term. While search engines are necessary, the underlying approach for building indices by crawling the Web ought to be improved or replaced.

Thesis 2 : (Virtual Web View) The Internet artifact collection is unstructured and difficult to manage. There exists a possibility to create a view on this collection to make it appear structured and to use the database technology to partially manage it.

Thesis 3 : (Web Mining) It is possible to extract explicit and implicit knowledge from the World-Wide Web document collection as implied by the content of the documents, the inter-connections between documents in hyperspace, and the access to these documents.

Thesis 4 : (WebML as a query language) There exists an SQL-like query language that combines capabilities for resource discovery and knowledge discovery using Virtual Web Views.

Thesis 5 : (Media Content) It is possible to use image content features like colours and textures to solve object similarity search in image and video collections.

Thesis 6 : (Media Repository Mining) Mining visual media metadata can yield interesting though unrevealed information about a medium or media collections.

1.1 Motivation and Research Description

We motivate the thesis by discussing a number of open problems related to the theses presented in the previous section.

1. **Web Overload:** Due to the huge amount of data rapidly accumulated in the World-Wide Web space, “surfing” the web to find information has become cumbersome. Finding real information is often a hit-and-miss process. Catalogues and searchable directories are somehow prohibited by the very dynamic nature of the Web and its resources which make these catalogues stale and useless very rapidly. Automatic collection of resources has partially solved the information retrieval problem on the World-Wide Web. Processes known as robots, spiders or crawlers, recursively traverse the Web space by retrieving web documents and following all links in them until no documents remain to be retrieved. This automatic document retrieval method exhaustively visits all documents assuming that all documents in cyberspace are somehow interconnected with the initial web documents with which the crawling process started.

By visiting web documents, indexes can be created to allow resource discovery using search engines. Notice that to index the whole Web space, all documents have to be downloaded one by one. In other words, the entire content of the Web is downloaded. Moreover, due to the continuous growth and change of the Web space, the crawling process has to be repeated continuously in order to assert a current index. This means that the content of the Web is continuously downloaded. Given the fact that today there are more than 400 different crawlers traversing the Web at all times[156], the content of the Web is continuously downloaded many times. The network traffic generated by crawlers is extremely high. Moreover, it has been demonstrated [237] that the network traffic on many web sites is in a high percentage constituted from search engine crawlers requests for web content for indexing purposes. In addition, spiders generate unnecessary localized load on already overloaded information servers. By continuously and consecutively requesting all documents from the same server, spiders can flood servers and prevent them from serving other users. This has precipitated controversy. Based on these concerns, guidelines for implementing spider-like programs were proposed [154, 155] in order to reduce the number of localized requests in a given lapse of time. Obviously, at the rate the Web space is growing and the number of crawlers is increasing, the current web crawling technology is not viable for effective Web space indexing, even with availability of larger bandwidth.

2. **Virtual Web View:** The World-Wide Web is a collection of artifacts with complex structures. Artifacts are physical objects like documents, images, videos, sound, maps, data files, games, applications etc. Artifacts can also be virtual objects, like users, hosts, networks etc., playing a role on the World-Wide Web. These objects are distributed and stored, or represented, on a large set of heterogeneous repositories. Querying such data is very costly, if not impossible, due to the semantic ambiguities and the heterogeneity of the data sources. Processing and generalizing (summarizing) the raw data can resolve certain ambiguities. By storing the processed data in a relational table at a more general conceptual level, the cost of query processing is reduced [208]. High level queries can be applied directly on the processed data. Moreover, users may prefer to scan the general description of the information on the Internet, rather than read the details of large pieces of information (i.e. web artifacts). This may lead to cooperative query answering [124] in which the user can successively refine

the query by following the description of the selected general summaries. The idea of generalizing data at a higher conceptual level can be repeated to form layers of generalization. These layers constitute a Multiple-Layered DataBase (MLDB) [127, 276]. The philosophy behind the construction of the MLDB is information extraction. Ideally, the extraction of information from the primitive data to build the first layer of the MLDB is automated. However, in many cases, depending upon the artifacts processed, manual help from the document author, the server administrator, or other, might be necessary. The advent of the eXtensible Markup Language (XML) [180], a new standard for defining and describing artifacts on-line, is a promising accomplishment that would facilitate the automation of the information extraction from the primitive layer. The adoption of XML applications for describing documents on-line could indeed be key in the efficient and automatic extraction fundamental data from within the documents. Once constructed, the MLDB provides a global view of the current artifacts in the World-Wide Web. We call it a Virtual Web View (VWV).

Definition 1.1.1 *A* **Virtual Web View** *is a set of relations organized in layers. Relations in a given layer summarize the relations in the lower layers, and are abstracted in the relations of the layers above, starting from the lowest layer containing descriptors of artifacts in the World-Wide Web.* □

A VWV is a view on the World-Wide Web artifact collection, thus abstracting a selected set of artifacts. Many VWVs can co-exist to cover the whole World-Wide Web. Another motivation promoting the layered VWV is the possibility of querying the view at different abstraction levels. In general, higher level relations of the VWV are much smaller than their corresponding lower ones. Thus, querying higher layers is faster and less costly. Higher layers of a VWV can be cached for efficiency and for reduction of network traffic.

3. **Web Mining:** Knowledge Discovery in Databases (KDD) is a process in which implicit knowledge is discovered and extracted from large databases. Data Mining, which locates and enumerates valid patterns in large databases, is one step among other steps in the KDD process. The steps of KDD as described in [202] are: *selection* during which data sets or data samples are selected, *preprocessing* during which the data is cleaned and preprocessed to eliminate noise, *transformation* during which the data

is reduced to its useful features, *data mining* during which some specific tasks locate patterns in the data, and *interpretation* during which the patterns discovered are evaluated and consolidated into knowledge. The heterogeneous, unstructured and chaotic World-Wide Web is not a database. The Web is a set of different data sources with unstructured and interconnected artifacts that continuously change. The selection, preprocessing and transformation steps of Knowledge Discovery in the Web (KDW) have to take into account the dynamic and heterogeneous nature of the Web. Moreover, the transformation step has to consider the fact that the artifacts on the web (i.e. web pages and media) are not structured like records in a database. In addition, the hyperlink structure inherent to the Web can yield interesting information that should be taken into account. An artifact linked by many documents is obviously popular, hence probably more important or relevant than a document that is not linked to by other artifacts. However, an artifact linked by many non-important (or irrelevant) documents is less pertinent than an artifact link by one relevant document. Obviously, links are a rich knowledge source. In addition, the access patterns of users on the Internet can reveal interesting knowledge about accessed artifacts.

Definition 1.1.2 Web Mining *is the extraction of interesting and potentially useful patterns and implicit information from artifacts or activity related to the World-Wide Web.* □

Definition 1.1.3 *The* **taxonomy of Web Mining** *domains includes* Web Content Mining *which pertains to the extraction of information from artifact content,* Web Structure Mining *which educes information from artifact link structure, and* Web Usage Mining *which tracks access patterns to Web artifacts.* □

4. **WebML as a query language:** SQL is a widely accepted declarative query language for relational databases. Many optimizers have been implemented for it, making it fast and reliable.

   WebML, the language we propose, exploits the Virtual Web Views and the concept hierarchies with which the layers of the VWV are constructed. New primitives and functions allow browsing, progressive browsing and knowledge discovery. Proposing a high level SQL-like language that enhances and enriches SQL syntax for resource and knowledge discovery has straightforward advantages. First, the new language can take advantage of the powerful SQL syntax structure and SQL expressive power. Second, an interpreter can be implemented to translate the queries into SQL and take advantage of the fast and reliable SQL query optimizers. Third, the new language has a better chance for acceptance by database users and Web programmers. SQL is a powerful query language, but is also a programming language for database applications. WebML is also intended as a high level programming language for information retrieval and data mining applications on the World-Wide Web.

5. **Media Content:** The use of image and video in multimedia databases has proven extremely effective in various applications such as education, entertainment, medicine, commerce, and publishing. Multimedia data is much richer than simple textual data. However, collections of multimedia objects present many management and retrieval challenges. The most commonly used indexing methods for visual artifacts are description-based. These approaches use manually entered keywords to index images or videos. They are inadequate in terms of scalability, and are very poor in retrieval effectiveness. Moreover, querying in this context is restricted to keywords and can not take advantage of image content like colours, textures, shapes or objects represented in the images. However, automatically extracting visual features from images poses many challenges regarding scalability of the process and efficiency in the use of these features for content-based retrieval.

6. **Media Repository Mining:** Knowledge discovery in multimedia databases has not been widely analyzed or studied. Data mining is a young field but it has already produced impressive results. Multimedia is very popular and the proliferation of multimedia repositories is significant. The marriage between data mining and multimedia is very tempting and logical. The structure of multimedia objects is however complex, and the descriptors of multimedia objects are unlike any common data processed by standard data mining techniques. Multimedia mining poses interesting challenges due to the complexity of the media artifacts and the high dimensionality of artifacts descriptors.

1.2 Contributions in this Thesis

The major contributions of this thesis are summarized as follows:

- Proposal of a framework and model, Virtual Web View, for hierarchical organization and management of Web objects for resource and implicit knowledge discovery form the Internet.
- Presentation of strategies for mediating between different virtual web views with distinct or interoperable ontologies.
- Definition of WebML, a declarative query and mining language for the Web.
- Definition of DMQL, a data mining query language for mining large databases.
- Proposal of automatic descriptors extraction means for image and video summarization aimed at content-based retrieval and data mining from visual media.
- Proposal of an architecture for a data mining and OLAP system from visual media using data cubes.
- Presentation of new efficient and scalable algorithms for content-based multimedia association rules with recurrent items and spatial relationships at different image resolution levels.

The following implementations were realized in the context of the thesis:

1. Implementation on the Web of a client server version of DBMiner, a data mining and OLAP system for large relational databases.
2. Implementation of an image discovery and indexing agent, Excavator, which retrieves images and related web pages and indexes images using visual descriptors for the images and keywords from the web pages.
3. Implementation of C-BIRD, a content-based image retrieval system from image repositories. The system was implemented as a Web-based application allowing image retrieval and resource discovery form the Web.
4. Implementation of a data mining and OLAP system, MultiMediaMiner, for mining characteristic, association, and classification rules from images retrieved form the Web.

1.3 Organization of the Thesis

The thesis is structured in two parts. The first part (Part I) presents our work pertaining to resource and knowledge discovery from the Internet, and is divided into two chapters: Chapter 2 and Chapter 3. The second part (Part II) covers data mining from visual media repositories, and is divided into three chapters: Chapter 4 through Chapter 6.

Chapter 2 introduces our stratified architecture used to build structured views on artifacts distributed on the Internet. We show how a layered structure can abstract information about Internet artifacts and offers to view the resources at high conceptual levels. We call these structures Virtual Web Views. A Virtual Web View covers part of the Web and uses local concept hierarchies to represent the "world" it exhibits. This work has been published in technical reports and various conference proceedings [127, 128, 276, 277]. Different virtual web views can coexist, each with its own set of concept hierarchies. Mediating between virtual web views to solve inter-view queries is also put forth in this Chapter. This work has been submitted for publication at the Conference on Cooperative Information Systems (CoopIS'99) [278].

In Chapter 3 we present a declarative query language, WebML, that takes advantage of the multi-layered structure and concept hierarchies present in virtual web views to discover resources as well as knowledge from the Internet. The complete syntax is unveiled and some examples are given to demonstrate the expressive power of the query language. WebML and its predecessors, WebQL and NetQL, were presented in the following articles [127, 128, 276, 277].

In Chapter 4 we present an overview of content-based visual media retrieval techniques and discuss our implementation of C-BIRD, our content-based image retrieval system. The chapter provides details on visual feature extraction from images and video frames for content-based querying and object localization. Our research related to content-based image retrieval has been published in [168, 169, 167].

Chapter 5 discusses the implementation of MultiMediaMiner a system for On-line Analytical Processing and high-level knowledge discovery from multimedia descriptors. The challenges we faced and the compromises we adopted are described in detail. MultiMediaMiner has been demonstrated at the SIGMOD Conference 1998 [279].

Finally, we show in Chapter 6 how data mining can be applied on image content to discover relationships between localized features in images or video frames. The research related to data mining on image content has been submitted for publication at the SIGKDD Conference 1999 [281].

There are four appendices. In Appendix A we present a survey of information retrieval techniques used for resource discovery on the Internet and outline some of the recent approaches and attempts for data mining from the web. A more extended survey has been published as a technical report [270] and submitted to the ACM Computing Surveys Journal [271].

Appendix B and C present the Backus-Naur Form grammar of the Web Mining Language WebML and the Data Mining Query Language DMQL, respectively.

Appendix D present available means for defining metadata for objects on the Internet. Examples of the Dublin Core elements are presented using HTML META tags and XML document type definition (DTD).

Part I

Web Mining

The problems that we have today were created at a particular level of thinking.
We can not solve these problems at the same level of thinking.

Albert Einstein

Nam et ipsa scientia potestas est.

Francis Bacon

Chapter 2

Building a Virtual Web View

With the rapid expansion of the information base and the user community in the Internet, efficient and effective discovery and use of the resources in the global information network has become an important issue in the research into global information systems.

Although research and developments of database systems have been flourishing for many years, with different kinds of database systems successfully developed and delivered to the market, a global information system, such as the Internet, stores a much larger amount of information in a much more complicated and unstructured manner than any currently available database systems. Thus, the effective organization, discovery and use of the rich resources in the global information network poses great challenges to database and information system researchers. In a recent report on the future of database research known as the Asilomar Report [29], it has been predicted that in ten years from now, the majority of human information will be available on the World-Wide Web, and it has been observed that the database research community has contributed little to the Web thus far.

The first major challenge of a global information system is the diversity of information in the global information base. The current information network stores hundreds of tera-bytes of information including documents, softwares, images, sounds, commercial data, library catalogues, user directory data, weather, geography, other scientific data, and many other types of information. Since users have the full freedom to link whatever information they believe useful to the global information network, the global information base is huge, heterogeneous, in multimedia form, mostly unstructured, dynamic, incomplete and even inconsistent, which creates tremendous difficulty in systematic management and retrieval in comparison with the structured, well-organized data in most commercial database systems.

The second challenge is the diversity of user community. The Internet currently connects about 40 million workstations [185, 283], and the user community is still expanding rapidly (See Figures A.6 and A.7 in Appendix A). Users may have quite different backgrounds, interests, and purposes of usage. Also, most users may not have a good knowledge about the structure of the information system, may not be aware of the heavy cost of a particular search (e.g., a click may bring megabytes of data over half of the globe), and may easily get lost by groping in the "darkness" of the network, or be bored by taking many hops and waiting impatiently for a piece of information.

The third challenge is the volume of information to be searched and transmitted. The huge amount of unstructured data makes it unrealistic for any database systems to store and manage and for any queries to find *all* or even *most* of the answers by searching through the global network. The click-triggered massive data transmission over the network is not only costly and unbearable even for the broad bandwidth of the communication network, but also too wasteful or undesirable to many users. Search effectiveness (e.g., hit ratio) and performance (e.g., response time) will be bottlenecks for the successful applications of the global information system.

There have been many interesting studies on information indexing and searching in the global information base with many global information system servers developed. Some of these studies and systems are presented in Appendix A, including attempts made to discover resources in the World Wide Web. Crawlers, spider-based indexing techniques used by search engines, like the WWW Worm [179], RBSE database [77], Lycos [178] and others, create a substantial value to the web users, but generate an increasing Internet backbone traffic. They not only flood the network and overload the servers but also lose the structure and the context of the documents gathered. These wandering software agents on the World Wide Web have already created controversies [154, 155] as mentioned in Appendix A. Other indexing solutions, like ALIWEB [153] or Harvest [34], behave well on the network but still struggle with the difficulty to isolate information with relevant context and cannot solve most of the problems posed for systematic discovery of resources and knowledge in the global information base.

In this chapter, a different approach, called a Multiple Layered DataBase (MLDB) approach for building Virtual Web Views (VWV) is proposed to facilitate information discovery in global information systems. We advocate spider-less indexing of the Internet. Authors or web-server administrators send their own indexes or pointers to artifacts to be indexed.

When documents are changed, added or removed, the indexing process is triggered again. A multiple layered database (MLDB) is a database composed of several layers of information, with the lowest layer (i.e., *layer-0*) corresponding to the primitive information stored in the global information base and the higher ones (i.e., *layer-1* and above) storing generalized information extracted from the lower layers.

The proposal is based on the previous studies on *multiple layered databases* [208, 124] and *data mining* [203, 119] and the following observations.

With the development of data analysis, transformation and generalization techniques, it is possible to generalize and transform the diverse, primitive information in the network into reasonably structured, classified, descriptive and higher-level information. Such information can be stored into a massive, distributed but structured database which serves as the layer-1 database in the MLDB. By transforming an unstructured global information base into a relatively structured "global database", most of the database technologies developed before can be applied to manage and retrieve information at this layer.

However, the layer-1 database is usually still too large and too widely distributed for efficient browsing, retrieval, and information discovery. Further generalization should be performed on this layer at each node to form higher layer(s) which can be then merged with the corresponding layered database of other nodes at some backbone site in the network. The merged database can be replicated and propagated to other remote sites for further integration [66]. This integrated, higher-layer database may serve a diverse user community as a high-level, global information base for resource discovery, information browsing, statistical studies, etc.

The multiple layered database architecture transforms a huge, unstructured, global information base into progressively smaller, better structured, and less remote databases to which the well-developed database technology and the emerging data mining techniques may apply. By doing so, the power and advantages of current database systems can be naturally extended to global information systems, which may represent a promising direction. Moreover, data mining can be put to use in such hierarchical structure in order to perform knowledge discovery on the World-Wide Web (or the Internet). The next section surveys some techniques and approaches relevant to Knowledge Discovery on the Internet also known as *Web Mining.* We intend to integrate most of these techniques in our proposal.

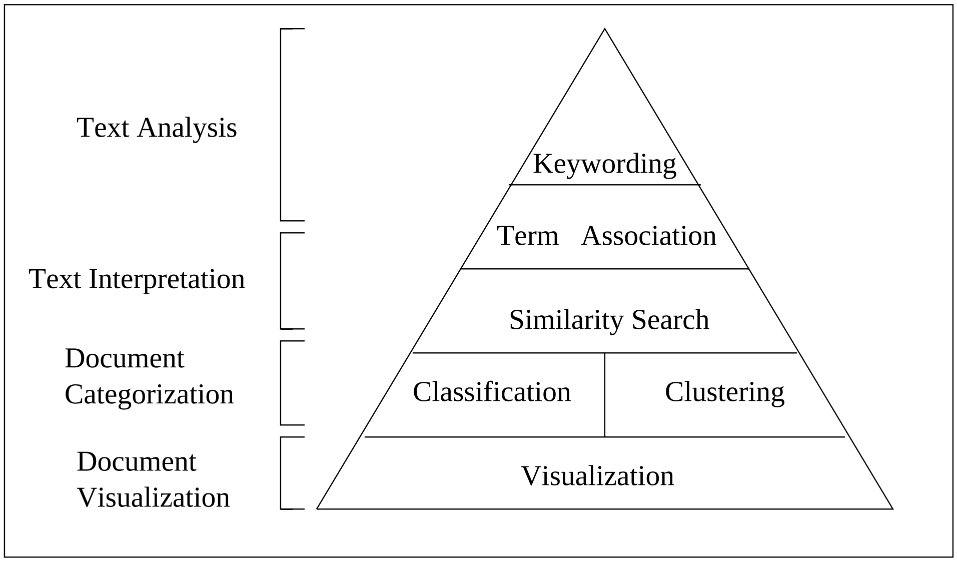


Figure 2.1: Text Mining Pyramid.

2.1 Data Mining or Knowledge Discovery on the Internet

Data mining, as defined in [202], is the process of non-trivial extraction of implicit, previously unknown and potentially useful information from data in large databases. Data mining is the principal core of the knowledge discovery process, which also includes data integration, data cleaning, relevant data selection, pattern evaluation and knowledge visualization. Traditionally, data mining has been applied to databases. The wide spread of the World-Wide Web technology has made the large document collection in the World-Wide Web a new ground for knowledge discovery research. In contrast to resource discovery that finds and retrieves resources from the Internet, knowledge discovery on the Internet aims at deducing and extracting implicit knowledge not necessarily contained in a resource. Traditional knowledge discovery functions put to use on databases, like characterization, classification, prediction, clustering, association, time series analysis, etc. can all be applied on the global information network. Not all these functionalities were attempted on the Internet but a few were applied to document repositories with some success. Text mining for instance has been attracting much interest.

One of the most important assets in any corporate organization is the collection of documents regularly amassed, like technical reports, articles, memos, presentations, patents, e-mail messages, web pages, etc. This collection of free text documents embodies the corporate cumulated expertise. When time is of the essence for success, even precise information retrieval systems pinpointing relevant documents can become insufficient. Text mining, however, can drastically improve the precision of information retrieval systems, or extract relevant knowledge from documents, alleviating the need for going through the retrieved documents manually in the search for pertinent knowledge. Text mining includes most of the knowledge discovery steps, from data cleaning to knowledge visualization. The dominant categories in text mining are text analysis, text interpretation, document categorization, and document visualization. Text analysis scrutinizes text document content to investigate syntactical correlation and semantic association between terms. Key concepts and phrases are extracted to represent the document or document sections (i.e. keywording). Text interpretation abstracts documents in concise form by paraphrasing the document content. Document categorization organizes a document collection in groups, while the document visualization consists of the representation of document concentration in groups and the group intersections. Figure 2.1 shows a pyramid of text mining themes ordered by category. The pyramid illustrates the top-down relationship between text mining functions. The different functions are:

Keywording: Extraction of relevant key phrases/words from documents. This extraction is limited to the only terms or concepts that are pertinent to the topic of the whole document (or section).

Summarization: Extraction of relevant key information from documents. The essential ideas of a document (or section) are abstracted and paraphrased in a synopsis.

Similarity search: Search for documents (or sections) containing similar concepts as a given document (or section).

Classification: Organization of a collection of documents by predefined themes. Given a set of classes and descriptions of classes, the documents are classified (or categorized). Documents can belong to one or more classes.

Clustering: Search for predominant themes in a collection of documents and categorization of all documents in the found themes.

Text mining is only one of the technologies that constitute knowledge discovery from the Internet. It primarily acts on textual content. The Internet is a highly dynamic multimedia environment involving interconnected heterogeneous repositories, programs, and interacting users. Obviously, text mining has a limited grasp on knowledge in such an environment.

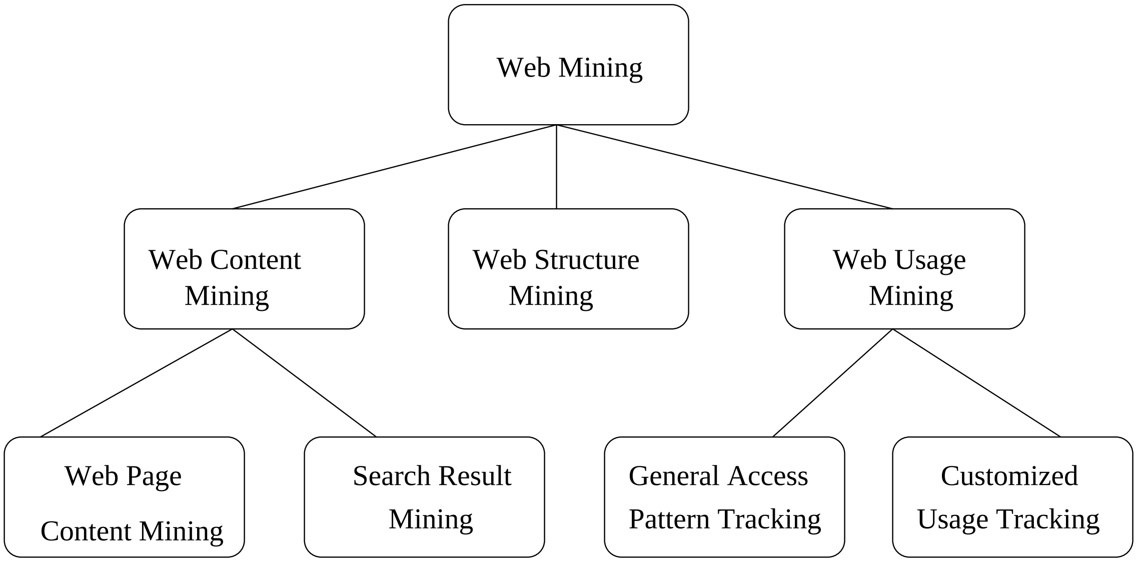


Figure 2.2: Taxonomy of Web Mining techniques.

Data mining on the Internet, commonly called webmining, needs to take advantage of the content of documents, but also of the usage of such resources available and the relationships between these resources. Web mining, the intersection between data mining and the World-Wide Web, is growing to include many technologies conventionally found in artificial intelligence, information retrieval, or other fields. Agent-based technology[81], concept-based information retrieval, information retrieval using case-based reasoning[64], and document ranking using hyperlink features and usage (like CLEVER) are often categorized under web mining. Web mining is not yet clearly defined and many topics will continue to fall into its realm.

We define Web Mining as *the extraction of interesting and potentially useful patterns and implicit information from artifacts or activity related to the World-Wide Web.*

Figure 2.2 shows a classification of domains that we believe to be akin to Web Mining. In the World-Wide Web field, there are roughly three knowledge discovery domains that pertain to web mining: Web Content Mining, Web Structure Mining, and Web Usage Mining. Web content mining is the process of extracting knowledge from the content of documents or their descriptions. Web document text mining, resource discovery based on concepts indexing or agent-based technology may also fall in this category. Web structure mining is the process of inferring knowledge from the World-Wide Web organization and links between references and referents in the Web. Finally, web usage mining, also known as Web Log Mining, is the process of extracting interesting patterns in web access logs.

2.1.1 Web Content Mining

Most of the knowledge in the World-Wide Web is buried inside documents. Current technology barely scratches the surface of this knowledge by extracting keywords from web pages. This has resulted in the dissatisfaction of users regarding search engines and even the emergence of human assisted searches[1] on the Internet. Web content mining is an automatic process that goes beyond keyword extraction. Since the content of a text document presents no machine-readable semantic, some approaches have suggested to restructure the document content in a representation that could be exploited by machines. Others consider the web structured enough to do effective web mining. Nevertheless, in either cases an intermediary representation is often relied upon and built using known structure of a limited type and set of documents (or sites) or using typographic and linguistic properties. The semi-structured nature of most documents on the Internet helps in this task. Essence[129], the technology used by the harvest system[34] relies on known structure of semi-structured documents to retrieve information. The usual approach to exploit known structure in documents is to use wrappers to map documents to some data model. Many declarative languages have been proposed to query such data models. Weblog[161] relies on Datalog-like rules to represent web documents. WebOQL[19] uses graph trees to extract knowledge and restructure web documents. WebML[127, 128, 276], presented in Chapter 3, uses relational tables to take advantage of relational database power and data mining possibilities. Techniques using lexicons for content interpretation are yet to come.

There are two groups of web content mining strategies: Those that directly mine the content of documents and those that improve on the content search of other tools like search engines.

[1] Some sites like http://www.humansearch.com, http://www.searchmill.com and http://www.searchforyou.com offer search services with human assistance.

Web Page Summarization

There has been some research work on retrieving information from structured documents, hypertext, or semi-structured documents[1, 2, 42]. However, most of the suggested approaches are limited to known groups of documents, and use custom-made wrappers to map the content of these documents to an internal representation. Perhaps the most prominent research results for knowledge discovery from heterogeneous and irregular documents, like web pages, were presented by the Ahoy!, WebOQL, and the Shopbot Project. Ahoy![2][226] specializes in discovering personal homepages. Given information about a person, Ahoy! uses Internet services like search engines and e-mail listservers to retrieve resources related to the person's data. Ahoy! uses heuristics to identify typographic or syntactic features inside the documents that could betray the document as being a personal homepage. WebOQL is a query language for web page restructuring. Using a graph tree representation of web documents, it is capable of retrieving information from on-line news sites like CNN[3] or tourist guides. The shopping agent described in [71] learns to recognize document structures of on-line catalogues and e-commerce sites, and extracts price lists and special offers. This agent is capable of compiling information retrieved from different sites and discovering interesting bargains.

The major obstacle for efficient information extraction from within documents is the absence of metadata, and the lack of a standard way to describe, manipulate and exchange data in electronic documents. The recommendations for XML 1.0 (eXtendible Markup Language) standard by the World-Wide Web Consortium in 1998, and its endorsement by many companies major players in the Web arena, is bringing relief for resource discovery.

XML provides a flexible data standard that can encode the content, semantics, and schema for a wide variety of electronic documents. XML is a universal data format that separates the data from the presentation of the document and enables documents to be self-describing using Document Type Definitions (DTD). See Section 2.3 for more details about XML.

[2] http://www.cs.washington.edu/research/ahoy

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

Search Engine Result Summarization

The heterogeneity of the World-Wide Web and the absence of structure has lead some researchers to mine subsets of known documents or data from documents known to pertain to a given topic. One such subset can be a search result of a query sent to search engines. The system presented in [183] uses a small relational table containing minimal information to provide a query language (WebSQL) for better result refining. The system accesses the documents retrieved by search engines and collects information from within the document or from the data usually provided by servers like the URL, title, content type, content length, modification date, and links. The SQL-like declarative language provides the ability to retrieve pertinent documents from within the search result. Zamir and Etzioni present in [284] a technique for clustering documents retrieved by a set of search engines. The technique relies solely on information provided in search result like titles, URLs, snippets (i.e. descriptions or first lines of the page content), etc. to induce clusters and categorize the retrieved documents in these discovered clusters. The clusters, which can present overlapping, represent a higher-level view on top of the list of retrieved documents and facilitate the sifting through the often very large search engine result list.

2.1.2 Web Structure Mining

Thanks to the interconnections between hypertext documents, the World-Wide Web can reveal more information than just the information contained in documents. For example, links pointing to a document indicate the popularity of the document, while links coming out of a document indicate the richness or perhaps the variety of topics covered in the document. This can be compared to bibliographical citations. When a paper is cited often, it ought to be important. The PageRank[40] and CLEVER[46] methods take advantage of this information conveyed by the links to find pertinent web pages.

We will present later in this chapter the virtual web views, using a multi-layered database approach, which also benefits from the structure of the Web by abstracting relevant information from web artifacts and keeping relationships between them. The virtual web views exploit the knowledge conveyed by the information network but not explicitly stated in documents. By means of counters, higher levels cumulate the number of artifacts subsumed by the concepts they hold. Counters of hyperlinks, in and out documents, retrace the structure of the web artifacts summarized.

2.1.3 Web Usage Mining

Despite the anarchy in which the World-Wide Web is growing as an entity, locally on each server providing the resources there is a simple and well structured collection of records: the web access log. Web servers record and accumulate data about user interactions whenever requests for resources are received. Analyzing the web access logs of different web sites can help understand the user behaviour and the web structure, thereby improving the design of this colossal collection of resources. There are two main tendencies in Web Usage Mining driven by the applications of the discoveries: General Access Pattern Tracking and Customized Usage Tracking. The general access pattern tracking analyzes the web logs to understand access patterns and trends. These analyses can shed light on better structure and grouping of resource providers. Many web analysis tools exist[4] but they are limited and usually unsatisfactory. We have designed a web log data mining tool, WebLogMiner, and proposed techniques for using data mining and OnLine Analytical Processing (OLAP) on treated and transformed web access files. These studies were presented in [282]. Applying data mining techniques on access logs unveils interesting access patterns that can be used to restructure sites in a more efficient grouping, pinpoint effective advertising locations, and target specific users for specific selling ads [141]. Customized usage tracking analyzes individual trends. Its purpose is to customize web sites to users. The information displayed, the depth of the site structure and the format of the resources can all be dynamically customized for each user over time based on their access patterns. One innovative study has proposed such adaptive sites: web sites that improve themselves by learning from user access patterns[200].

While it is encouraging and exciting to see the various potential applications of web log file analysis, it is important to know that the success of such applications depends on what and how much valid and reliable knowledge one can discover from the large raw log data. Current web servers store limited information about the accesses. Some scripts custom-tailored for some sites may store additional information. However, for an effective web usage mining, an important cleaning and data transformation step before analysis may be needed[282].

[4] The university of Illinois maintains a list of web access analyzers on a HyperNews page accessible at http://union.ncsa.uiuc.edu/HyperNews/get/www/log-analyzers.html

2.2 A Multiple Layered Database Model for Global Information Systems

In this section we present the multi-layered database structure underlying the virtual web views. This structure takes into account the three aspects of web mining presented above: web content mining, web structure mining, and web usage mining.

Although it is difficult to construct a data model for the primitive global information base (i.e., layer-0), advanced data models can be applied in the construction of better structured, higher-layered databases. To facilitate our discussion, we assume that the nonprimitive layered database (i.e., layer-1 and above) is constructed based on an extended-relational model with capabilities to store and handle complex data types, including set- or list- valued data, structured data, hypertext, multimedia data, etc. Multiple layered databases can also be constructed similarly using other data models, including object-oriented and extended entity-relationship models.

Definition 2.2.1 *A global* multiple layered database (MLDB) *consists of 3 major components: $\langle \mathcal{S}, H, D \rangle$, defined as follows.*

1. $\mathcal{S}$: a database schema*, which contains the meta-information about the layered database structures;*

2. $\mathcal{H}$: a set of concept hierarchies*; and*

3. $\mathcal{D}$: a set of (generalized) database relations at the nonprimitive layers of the MLDB and files in the primitive global information base. □

The first component, a database schema, outlines the overall database structure of the global MLDB. It stores general information such as structures, types, ranges, and data statistics about the relations at different layers, their relationships, and their associated attributes as well as the location where the layers reside and are mirrored. Moreover, it describes which higher-layer relation is generalized from which lower-layer relation(s) (i.e., a route map) and how the generalization is performed (i.e., generalization paths). Therefore, it presents a route map for data and metadata (i.e., schema) browsing and for assistance of resource discovery.

The second component, a set of concept hierarchies, provides a set of predefined concept hierarchies which assist the system to generalize lower layer information to high layer ones and map queries to appropriate concept layers for processing. These hierarchies are also used for query-less browsing of resources like drill-down and roll-up operations.

The third component consists of the whole global information base at the primitive information level (i.e., layer-0) and the generalized database relations at the nonprimitive layers. In other words, it contains descriptions of on-line resources summarized in each layer.

The third component is by definition dynamic. Note that the first, as well as the second component, can also dynamically change. The schema defined in the first component of the MLDB model can also be enriched with new fields, and new route maps can be defined after the system has been initially conceived. The updates are incremental and are propagated, in the case of the schema update, from lower layers to higher ones. New concept hierarchies can be defined as well, or updated. While updates to current concept hierarchies imply incremental updates in layered structure, new concept hierarchies may suggest the definition of a new set of layers or an analogue MLDB.

We first examine the database schema. Because of the diversity of information stored in the global information base, it is difficult, and even not realistic, to create relational database structures for the primitive layer information base. However, it is possible to create relational structures to store reasonably structured information generalized from primitive layer information. For example, based on the accessing patterns and accessing frequency of the global information base, layer-1 can be organized into dozens of database relations, such as *document, person, organization, images, sounds, software, map, library_catalogue, commercial_data, geographic_data, scientific_data, games*, etc. The relationships among these relations can also be constructed either explicitly by creating relationship relations as in an entity-relationship model, such as *person-organization*, or implicitly (and more desirably) by adding the linkages in the tuples of each (entity) relation during the formation of layer-1, such as *adding URL* [5] *pointers pointing to the corresponding authors ("persons") in the tuples of the relation "document" when possible.*

To simplify our discussion, we assume that the layer-1 database contains only two relations, document and person. Other relations can be constructed and generalized similarly.

Example 2.2.1 Let the database schema of layer-1 contain two relations, document and person, as follows (with the attribute type specification omitted).

[5] *Uniform Resource Locator. Reference is available by anonymous FTP from ftp.w3.org as /pub/www/doc/url-spec.txt*

1. document(*file_addr, authors, title, publication, publication_date, abstract, language, table_of_contents, category_description, keywords, index, multimedia_attached, num_pages, format, first_paragraphs, size_doc, timestamp, access_frequency, URL_links_in, URL_links_out, ...*).

2. person(*last_name, first_name, home_page_addr, position, picture_attach, phone, e-mail, office_address, education, research_interests, publications, size_of_home_page, timestamp, access_frequency, ...*).

Take the *document* relation as an example. Each tuple in the relation is an abstraction of one *document* from the information base (layer-0). The whole relation is a detailed abstraction (or descriptor) of the information in documents gathered from a site. The first attribute, *file_addr*, registers its file name and its "URL" network address. The key could have been a system generated object identifier *doc_id*, used to identify the documents which may be duplicated and have different URL addresses, such as in [233]. However, for simplicity we chose to retain the URL of a document as a key and duplicate the entries in *document* if necessary, allowing documents to evolve independently. There is a possibility to have two URLs for the same on-line document, especially with virtual domain addresses, but it is difficult to identify and thus we chose not to add another identifier other than the document URL. There are several attributes which register the information directly associated with the file, such as *size_doc* (size of the document file), *timestamp* (the last updating time), etc. There are also attributes related to the formatting information. For example, the attribute *format* indicates the format of a file: .ps, .dvi, .tex, .troff, .html, text, compressed, uuencoded, etc. One special attribute, *access_frequency*, registers how frequently the entry is being accessed. This is either access relative to the record in layer-1 or access collected from the web log file of the web server where the document resides. URL_links_in, URL_links_out, register the number of known pointers pointing to the document (i.e. popularity of the document), and the number of pointers coming out of the documents (i.e. number of URLs in the document). The popularity of a document can be weighted relatively to the importance of the initial document that point at it. If the initial document (i.e. parent document) is in the same topic or is popular itself, the counter is multiplied by a higher coefficient, however, if the initial document is not relevant or from the same site as the current document, the counter is multiplied by a low coefficient. $URL\_Links\_in \equiv \sum_{i,j} C_i$, where j is the number of distinct URLs pointing to the document, and C_i is 1 for an irrelevant parent page from the same web site, and higher otherwise. Relevance in this context can be measured by intersection of the document keyword sets (topics). The same applies for URL_Links_out: $URL\_Links\_out \equiv \sum_{i,k} C_i$, where k is the number of distinct URLs in the document, and

C_i their "importance". Other attributes register the major semantic information related to the document, such as *authors, title, publication, publication_date, abstract, language, table_of_contents, category_description, keywords, index, multimedia_attached, num_pages, first_paragraphs*, etc. □

Note that getting the Links_out list of a document is straightforward, however, the Links_in list can be difficult if we look at the links as a sparse matrix between all existing URLs. Such a matrix for the Web can not be computed in a realistic manner. In the VWV context, Links_in contains only "known links", that is links from documents in the VWV. When a document is added, its Links_out is divided into two sets: the known links and the outside VWV links. The known links are used to update the Links_in of those documents in the VWV. The Links_in of the new document is computed by checking for the URL of the new document in the Links_out lists of the VWV.

Layer-1 is a detailed abstraction (or descriptor) of the layer-0 information. The relations in layer-1 are substantially smaller than the primitive layer global information base but still rich enough to preserve most of the interesting pieces of general information for a diverse community of users to browse and query. Layer-1 is the lowest layer of information manageable by database systems. However, it is usually still too large and too widely distributed for efficient storage, management and search in the global network. Further compression and generalization can be performed to generate higher layered databases.

Example 2.2.2 Construction of an MLDB on top of the layer-1 global database.

The two layer-1 relations presented in Example 2.2.1 can be further generalized into layer-2 database which may contain two relations, doc_brief and person_brief, with the following schema,

1. doc_brief(*file_addr, authors, title, publication, publication_date, abstract, language, category_description, keywords, num_pages, format, size_doc, access_frequency, URL_links_in, URL_links_out*).

2. person_brief (*last_name, first_name, publications, affiliation, e-mail, research_interests, size_home_page, access_frequency*).

The resulting relations are usually smaller with less attributes and records. Least popular fields from layer-1 are dropped, while the remaining fields are inherited by the layer-2 relations. Relations are split according to different classification schemes, while tuples are merged relying on successive subsumptions according to the concept hierarchies used. General concept hierarchies are provided explicitly by domain experts. Other hierarchies are built automatically and stored implicitly in the database. We have proposed and implemented a technique for the construction of a concept hierarchy for keywords extracted from web pages using an enriched WordNet semantic network[263]. This approach [274, 275] is presented later in this Chapter.

Further generalization can be performed on layer-2 relations in several directions. One possible direction is to partition the *doc_brief* file into different files according to different classification schemes, such as category description (e.g., *cs_document*), access frequency (e.g., *hot_list_document*), countries, publications, etc., or their combinations. Choice of partitions can be determined by studying the referencing statistics. Another direction is to further generalize some attributes in the relation and merge identical tuples to obtain a "summary" relation (e.g., *doc_summary*) with data distribution statistics associated [119]. The third direction is to join two or more relations. For example, *doc_author_brief* can be produced by generalization on the join of *document* and *person*. Moreover, different schemes can be combined to produce even higher layered databases.

A few layer-3 relations formed by the above approaches are presented below.

1. cs_doc(*file_addr, authors, title, publication, publication_date, abstract, language, category_description, keywords, num_pages, format, size_doc, access_frequency, URL_links_in, URL_links_out*).

2. doc_summary(*affiliation, field, publication_year, first_author_list, file_addr_list, average_popularity, count*).

3. doc_author_brief(*file_addr, authors, affiliation, title, publication, pub_date, category_description, keywords, num_pages, format, size_doc, access_frequency, URL_links_in, URL_links_out*).

4. person_summary (*affiliation, research_interest, year, num_publications, count*).

The attribute *count*, is a counter that reckons the records from the lower layer generalized into the current record. *average_popularity* averages the *URL_links_in* count of the generalized records from the lower layer.

In general, the overall global MLDB structure is constructed based on the study of frequent accessing patterns. It is also plausible to construct higher layered databases for a special-interest community of users (e.g., ACM/SIGMOD, IEEE/CS) on top of a common layer of the global database. This generates partial views on the global information network, hence, the name Virtual Web View (VWV). A VWV provides a window to observe a subset of Web artifacts, and gives the illusion of a structured world.

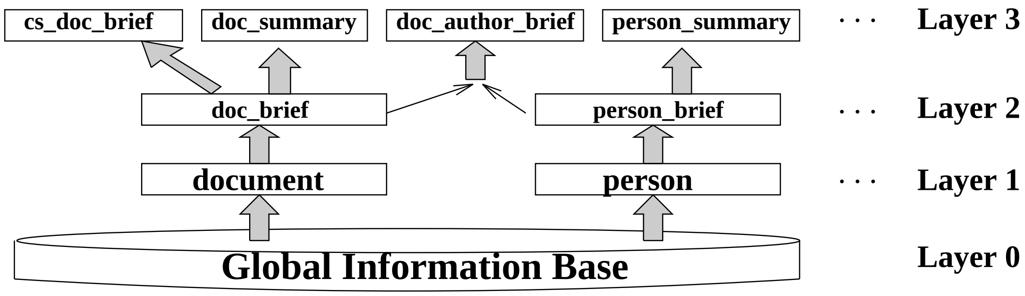


Figure 2.3: A conceptual route map of the global information base

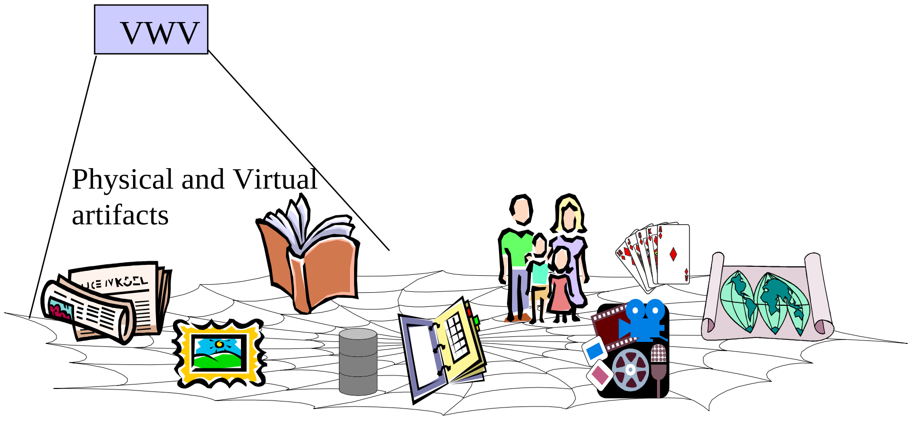


Figure 2.4: A VWV abstracts a selected set of artifacts and makes the WWW appear as structured.

This customized local higher layer acts as cache which may drastically reduce the overall network traffic [67, 15]. Some systems like Lagoon[6] "mirror" remote documents, but we believe that caching indexes (i.e. high layers containing descriptors) would be definitely more profitable.

One possible schema of a global MLDB (containing only two layer-1 relations) is presented in Figure 2.3. □

2.3 Metadata Matters

The first step, and probably the most challenging one in the construction of the layered structure of the VWV, is the transformation and generalization of the unstructured data of the primitive layer into relatively structured data, manageable and retrievable by databases. The challenge is mostly due to the common and persistent absence of information describing information in the primitive layer: Metadata.

2.3.1 The Dublin Core Metadata Initiative

Since 1995, the Dublin Core invitational workshop series has gathered librarians, digital library researchers, content experts, text-markup experts from around the world to discuss and promote better discovery standards for digital resources [259]. These experts in the library and digital library research community have stressed the importance of metadata in networked digital documents to facilitate resource discovery. With the phenomenal growth of networked resources, finding relevant information on the Internet became problematic. The lack of document semantic descriptors hinders the progress in indexing techniques. The primary goal that motivated the participants at the Dublin Core workshop series was to find a simple international consensus for describing metadata for digital documents on the Internet or other information systems. They insisted on simple and commonly understood semantics, conformity to emerging standards, extensibility, and interoperability with indexing systems. What emerged from this effort is a set of 15 element descriptors to describe the content and the representation of digital documents, as well as intellectual properties related to the documents. The elements have descriptive names intended to convey common semantic understanding. To promote and insure interoperability, some of the element

[6] Lagoon Caching Software Distribution, available from ftp://ftp.win.tue.nl/pub/infosystems/www/README.lagoon

descriptions (such as SUBJECT, TYPE, and FORMAT) are associated with a controlled vocabulary for the respective element values. Some of these enumerated lists of values are still under development in the Dublin Core workshop series. Other elements (such as DATE or LANGUAGE) follow strict ISO standards or Network Working Group recommendations (rfc for "request for comments").

The following element descriptions are taken from the RFC 2413 *Description of Dublin Core Elements* [258]:

1. *Title (Label:* **TITLE***)*
   The name given to the resource, usually by the Creator or Publisher.

2. *Author or Creator (Label:* **CREATOR***)*
   The person or organization primarily responsible for creating the intellectual content of the resource. For example, authors in the case of written documents, artists, photographers, or illustrators in the case of visual resources.

3. *Subject and Keywords (Label:* **SUBJECT***)*
   The topic of the resource. Typically, subject will be expressed as keywords or phrases that describe the subject or content of the resource. The use of controlled vocabularies and formal classification schemes is encouraged.

4. *Description (Label:* **DESCRIPTION***)*
   A textual description of the content of the resource, including abstracts in the case of document-like objects or content descriptions in the case of visual resources.

5. *Publisher (Label:* **PUBLISHER***)*
   The entity responsible for making the resource available in its present form, such as a publishing house, a university department, or a corporate entity.

6. *Other Contributor (Label:* **CONTRIBUTOR***)*
   A person or organization not specified in a CREATOR element who has made significant intellectual contributions to the resource but whose contribution is secondary to any person or organization specified in a CREATOR element (for example, editor, transcriber, and illustrator).

7. *Date (Label:* **DATE***)*
   A date associated with the creation or availability of the resource. Recommended best

practice is defined in a profile of ISO 8601 that includes (among others) dates of the forms YYYY and YYYY-MM-DD. In this scheme, for example, the date 1994-11-05 corresponds to November 5, 1994.

8. *Resource Type (Label:* **TYPE***)*
   The category of the resource, such as home page, novel, poem, working paper, technical report, essay, dictionary. For the sake of interoperability, Type should be selected from an enumerated list that is currently under development in the workshop series.

9. *Format (Label:* **FORMAT***)*
   The data format and, optionally, dimensions (e.g., size, duration) of the resource. The format is used to identify the software and possibly hardware that might be needed to display or operate the resource. For the sake of interoperability, the format should be selected from an enumerated list that is currently under development in the workshop series.

10. *Resource Identifier (Label:* **IDENTIFIER***)*
    A string or number used to uniquely identify the resource. Examples for networked resources include URLs and URNs (when implemented). Other globally-unique identifiers, such as International Standard Book Numbers (ISBN) or other formal names are also candidates for this element.

11. *Source (Label:* **SOURCE***)*
    Information about a second resource from which the present resource is derived. While it is generally recommended that elements contain information about the present resource only, this element may contain metadata for the second resource when it is considered important for discovery of the present resource.

12. *Language (Label:* **LANGUAGE***)*
    The language of the intellectual content of the resource. Recommended best practice is defined in RFC 1766.

13. *Relation (Label:* **RELATION***)*
    An identifier of a second resource and its relationship to the present resource. This element is used to express linkages among related resources. For the sake of interoperability, relationships should be selected from an enumerated list that is currently

under development in the workshop series.

14. *Coverage (Label:* **COVERAGE***)*
    The spatial or temporal characteristics of the intellectual content of the resource. Spatial coverage refers to a physical region (e.g., celestial sector) using place names or coordinates (e.g., longitude and latitude). Temporal coverage refers to what the resource is about rather than when it was created or made available (the latter belonging in the Date element). Temporal coverage is typically specified using named time periods (e.g., Neolithic) or the same date/time format ISO 8601 as recommended for the Date element.

15. *Rights Management (Label:* **RIGHTS***)*
    A rights management statement, an identifier that links to a rights management statement, or an identifier that links to a service providing information about rights management for the resource.

While the order of the elements is not important, each element is optional and may be repeated in the same resource.

The Dublin Core element set is already sanctioned by the World Wide Web Consortium (W3C) and is approved by a multitude of organizations. Many international digital library projects[7] have already adopted and are using the Dublin Core Metadata element set to describe their electronic networked resources. The promotion of the Dublin Core Metadata set of commonly understood descriptors improves the possibilities of semantic interoperability across disciplines and information systems, and, if widely ratified and used, it would greatly assist the interpretation of on-line artifacts and hence, facilitate the construction of Virtual Web Views.

2.3.2 XML The eXtensible Markup Language

Although there exist search engines for postscript documents[8] and others for other types of documents like images, most documents considered for indexing on the Internet are HTML (Hypertext Markup Language) documents. HTML is a simple language composed of a fixed

[7] A list of projects is available at http://purl.org/dc/projects

[8] "ML Papers", first released in 1997 by Andrew Ng, is a search engine that automatically extracts titles, authors and abstracts from postscript papers found on the Web. It can be accessed at http://gubbio.cs.berkeley.edu/mlpapers/.

set of tags that describe how a document should be displayed. The simplicity and the portability of HTML made it extremely popular and widely accepted standard. However, while HTML provides rich facilities for visualization of document content, it does not provide any standard-based way to manage or "comprehend" the data. Although standards like HTML are necessary for the visual part of the digital documents, they are insufficient for managing, representing and manipulating data on-line. There is a need for a better format that allows data exchange and intelligent search. These are the needs that lead to the XML (eXtensible Markup Language) recommendations by the World-Wide Web Consortium[9]. While XML resembles HTML, it complements it by describing data, such as authors and keywords, or even temperature and price. XML has also tags, but the set of tags is unlimited since developers can define their own. A document such as the following is perfectly valid in XML:

```
<THESIS>
  <TITLE> Resource and Knowledge Discovery from the Internet and Multimedia Repositories </TITLE>
  <AUTHOR>Osmar Rachid Zaïane</AUTHOR>
  <DEGREE>Doctor of Philosophy</DEGREE>
  <UNIVERSITY>Simon Fraser University</UNIVERSITY>
  <DEPARTMENT>Computing Science</DEPARTMENT>
  <COPYRIGHTYEAR>1999</COPYRIGHTYEAR>
  <DEFENCE>MARCH 1999</DEFENCE>
</THESIS>
```

The language is not about tags but is a framework that provides a uniform method for describing and exchanging structured data. This is a significant improvement on HTML since XML, which is actually a meta-language, facilitates more precise declarations of content and more meaningful search results across multiple platforms. XML is a meta-language because it gives the possibility to define tags that in turn describe content. In other words, with XML, one can define a set of tags (i.e. an XML-based markup language) with which documents or records can be described. An XML-based markup language, once defined, is an XML application, like HTML is an application of SGML (Standard Generalized Markup Language-ISO 8879). There are already many industry standard XML-based languages (or applications) such as MML (Mathematical Markup Language) for mathematical documents,

[9] http://www.w3c.org

CML (Chemical Markup Language) for chemistry, OTP (Open Trading Protocol) and OFX (Open Financial Exchange) for electronic commerce.

The interesting factor in XML, is that an XML document can describe its own format. An XML-based markup language consists of a set of element types that were given a label and a semantic. This set of element types serve to define a *type* of documents and is referred to as *Document Type Definition (DTD)*. An XML document is associated to a DTD, but a DTD can be associated to many documents. The DTD describes the format of a document by defining data tags, their order, and their nested structure. A validating XML parser would use the DTD to verify that the document contains all required tags in the specified order. DTDs allow the creation of vertical applications. The industry standards mentioned above, such as MML, CML, OTP, OFX, etc., are all using their own standard DTDs for interchangeable data. A standard DTD will eventually be proposed for web documents, to aid search engines and web mining applications in better extracting content from web pages. We present in Appendix D an XML document type definition for the Dublin Core elements. This DTD, if used, would help an XML parser interpret the information of a web document to readily build the first layer of our virtual web views.

There are many projects in progress related to XML still in working draft stage at the World-Wide Web Consortium, such as XSL (eXtensible Stylesheet Language) for formatting semantics of XML documents, XLL (eXtensible Link Language) which captures hypertext and hypermedia information, and XQL a query language for querying XML documents.

While XML allows the creation of powerful vertical applications (i.e. specialized in a given domain or domains), the development of horizontal XML application is limited by the semantic interpretation of tags defined in the different DTDs. Since the creator of XML documents gets to define the tags, tags can indeed have mnemonic names like <AUTHOR> to describe the author or authors of a document. While this may seem very interesting and powerful, this freedom to create tag label at will limits interoperability between applications. As a matter of fact, an XML parser may not be able to distinguish between <AUTHOR>, <AUTHORS>, <AUT>, <AUTEUR>, <CREATOR>, <CRT>, <WRITTENBY>, <WRITER>, etc., even though the goal of all these labels, defined in their respective DTDs, would be to describe the document's authors list. This is the reason we believe there is a need for a limited standard set of descriptors and we recommend the Dublin Core element set (see Appendix D).

Interoperability between web resource applications requires conventions not only about structure and syntax (i.e. organization of metadata and encoding grammar), which XML provides, but also requires conventions about semantics: the meaning of elements in the markup language. Providing common and consistent semantics is what current standards lack. A particular application of XML that supports a consistent structural representation of semantics, RDF, is being studied and will be presented and endorsed by the World-Wide Web Consortium. RDF stands for Resource Description Framework [162]. Its essential purpose is to support metadata interoperability using XML as syntax for interchange . It provides the infrastructure that will enable interoperability between applications that exchange metadata. In this general purpose framework, vocabularies can be declared using properties defined by specific expert communities that put constraints on values to use in these vocabularies. We believe that RDF, relying on the support of XML will help the deployment of metadata on the World-Wide Web and will make the enrichment of on-line documents with useful and practical metadata a common practice.

2.4 Construction and maintenance of MLDBs

A philosophy behind the construction of MLDB is information abstraction, which assumes that most users may not like to read the details of large pieces of information (such as complete documents) but may like to scan the general description of the information. Usually, the higher level of abstraction, the better structure the information may have. Thus, the sacrifice of the detailed level of information may lead to a better structured information base for manipulation and retrieval.

Figure 2.5 presents the general architecture of a multiple layered global information base, where the existing global information base forms layer-0, and the abstraction of layer-0 forms layer-1. Further generalization of layer-1 and their integration from different sites form layer-2, which can be replicated and propagated to each backbone site, and then be further generalized to form higher layers.

2.4.1 Construction of layer-1: From Global Information Base to Structured Database

The goal for the construction of layer-1 database is to transform and/or generalize the unstructured data of the primitive layer at each site into relatively structured data, manageable and retrievable by the database technology. Three steps are necessary for the realization

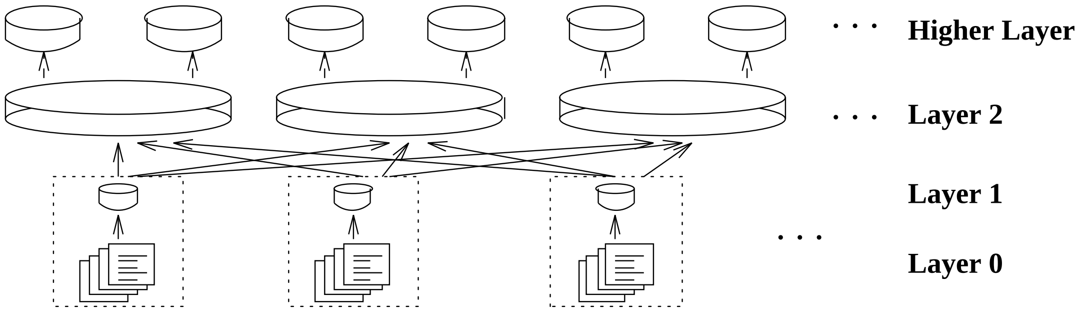


Figure 2.5: The general architecture of a global MLDB and its construction.

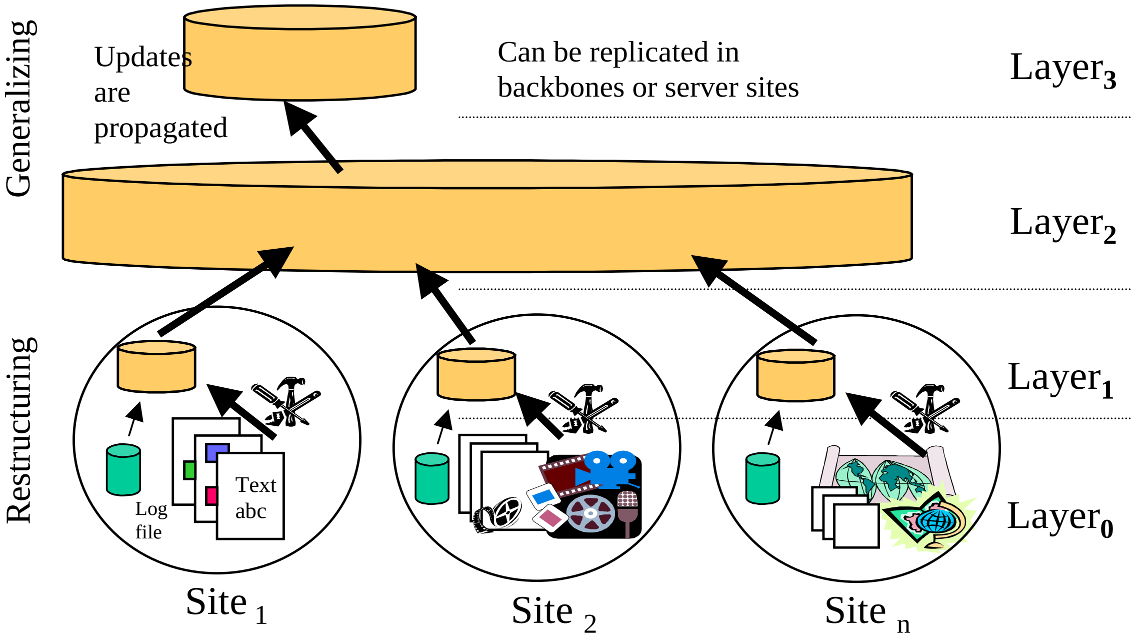


Figure 2.6: Using tools to generate and update the first layer of the MLDB structure.

of this goal: (1) standardization of the layer-1 schema, (2) development of a set of softwares which automatically perform the layer-1 construction, and (3) layer construction and database maintenance at each site.

Obviously, it is neither realistic nor desirable to enforce standards on the format or contents of the primitive layer information in the global information network. However, it is desirable to construct a rich, shared and standardized layer-1 schema because such a schema may lead to the construction of a structured information base and facilitate information management and sharing in the global information network.

While enforcing a standard for describing document content, which would considerably help in the construction of the first layer of the MLDB, is a difficult and probably, and arguably, an utopian task, some industry standards such as XML (and XML-based applications) and other recommendations such as the Dublin Core Metadata initiative and the W3C Resource Description Framework, are gaining momentum and will become in the near future a great asset that will facilitate the implementation of the first MLDB stratum in an efficient and economical way. Metadata and a uniform interpretation of descriptors of metadata are key issues in this respect. Initially, a Virtual Web View can either be restricted to a niche of resources with metadata attached, or compromise with information extracted or deduced by specialized tools and agents.

A standard layer-1 schema can be worked out by studying the accessing history of a diverse community of users and predicting the future accessing patterns by experts. Such a schema is constructed incrementally in the process of increasingly popular use of the information network, or standardized by some experts or a standardization committee. However, it is expected that such a schema may need some infrequent, incremental modifications, and the layer-1 database would be modified accordingly in an automatic and incremental way.

To serve a diverse community of users for flexible information retrieval, the layer-1 schema should be rich enough to cover most popular needs, and detailed enough to reduce excessive searches into the layer-0 information base. Notice that different VWVs can coexist to serve different community needs. Because of the diversity of information in a global information base, there often exist cases in which data have complex structures or cannot match the specified schema. We examine a few such cases.

1. Attribute values with complex structures or in multimedia forms.

   Some attributes may contain set- or list- valued data or nested structures. For example, the attribute "*authors*" contains a list of authors each having a structure: *name, affiliation, e-mail*, etc. Some attributes may be of a long text form (e.g., *abstract, first_paragraphs*) or in a multimedia form (e.g., the *picture* of a *person* could be a *photo* or a segment of *video*). Such nested structures, sets, lists, hypertext or multimedia data can be defined by extended data types, as in many extended-relational or object-oriented database systems [79, 152].

2. Missing attribute values.

   Since different users may have different conventions to connect the information to the network, it often happens that some attribute values may be missing. For example, *publication, publication_date*, and *table_of_contents* may not exist in a particular document. Descriptive elements from the Dublin Core are not mandatory but are all optional. Although missing values can be handled in a straight forward manner by introducing a null value, such as *value_unavailable*, efforts should be paid to dig up the values implicitly stored in the global information base. Index can be constructed automatically and/or selectively by searching through a document for frequently occurring terms. Keywords may need to be extracted, if not already present, from the frequently appearing technical terms in the text. The document publication information, if not existing in the *document*, can be extracted from the corresponding authors' publication record, or other relevant information. Tools like Ahoy! [226] can be used to find automatically personal home pages of authors. A person's research interest, may be extractable from his/her home page or from the researcher's frequently used keywords in his/her publications.

3. Inconsistent or variant attribute values.

   Some attributes of an entry may consist of multiple, potentially inconsistent values due to multiple entries of information. For example, a person may have several working addresses. Also, some attributes may contain a set of variant values. For example, one document may have several variant forms (such as .tex, .dvi, or .ps forms) on the network. The layer-1 schema should be flexible enough to allow variations. Multiple variations or their pointers should be stored in the corresponding attributes, with flags identifying their distinctions.

Since the layer-1 construction is a major effort, a set of softwares should be developed to automate the construction process. (Notice that some existing global information index construction softwares, like the Harvest Gatherer [34], have contributed to such practice).Layer-1 construction softwares are a collection of such tools. The eminent appearance of standards and recommendations for metadata availability such as Dublin Core, XML and RDF, will simplify these tools. The schema of the relation document in the example 2.2.1 can be easily filled with the Dublin Core element set and automatic extraction of some additional already existing attributes: doc_size, modif_date, access, links, format, etc. Some already existing tools and standards extract and represent other necessary information needed for layer-1 construction. vCard[10], for instance, is used to exchange information about people on-line. Table 2.1 shows an example how RDF vCard objects can be integrated with other metadata standards, in this case the Dublin Core metadata standard, and encoded with RDF. Tools exist to find personal homepages, e-mail addresses, and even phone numbers of people on-line.

```
<? xml version="1.0" ?>
<RDF xmlns = "http://w3.org/TR/WD-rdf-syntax#"
      xmlns:DC = "http://purl.org/DC/1.0#"
      xmlns:vCard = "http://imc.org/vCard/3.0#" >

<Description about = "http://www.cs.sfu.ca/~zaiane/thesis.html" >
<DC:Title> Resource and Knowledge Discovery from the Internet and Multimedia Repositories </DC:Title>
<DC:Creator parseType="Resource">
      <vCard:FN> Osmar Zaiane </vCard:FN>
      <vCard:N parseType="Resource">
            <vCard:Family> Zaiane </vCard:Family>
            <vCard:Given> Osmar </vCard:Given>
      </vCard:N>
      <vCard:EMAIL>
            <value> zaianecs.sfu.ca</value>
            <type resource ="http://imc.org/vCard/3.0#internet" />
      </vCard:EMAIL>
</DC:Creator>
<DC:Date> 1999-01-12 </DC:Date>
<DC:Subject> Data Mining, WWW, Knowledge Discovery, Visual Data, Content-Based Retrieval </DC:Subject>
<DC:Publisher> Simon Fraser University </DC:Publisher>
<DC:Rights> Copyright 1999 </DC:Rights>
</Description>
</RDF>
```

Table 2.1: Example of RDF object with vCard and Dublin Core elements.

XML standard applications are in the works to describe and exchange information about software (games, business applications, etc.), databases, and other entities on-line that we consider Web artifacts worth being abstracted in Layer-1. We will describe in Chapter 4

[10] *vCard* (The Electronic Business Card) is a standard to automate the exchange of personal information typically found on a traditional business card. Information can be found in RFC 2425 (MIME Content-Type for Directory Information) and RFC 2426 (vCard MIME Directory Profile), and specification for version 2.1 in http://www.imc.org/pdi/vcard-21.doc.

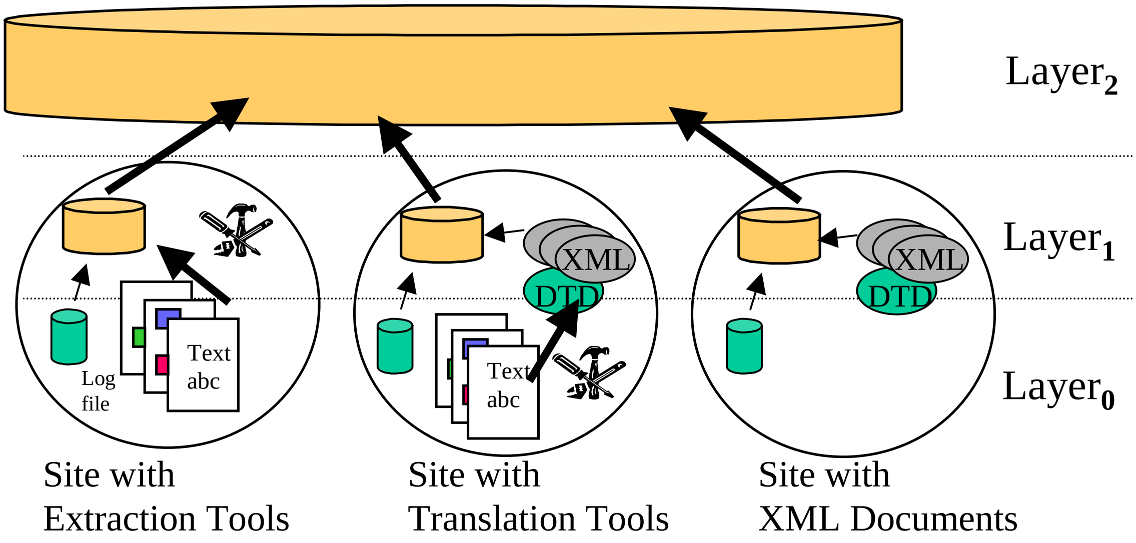


Figure 2.7: Extraction of metadata for Layer-1 construction.

tools that we developed for extraction of relevant information from visual media such as images and video. The gathering of metadata to create Layer-1 is already possible to a certain extent, but will eminently be possible for major artifacts available on the Internet thanks to metadata description standards and the availability of new tools.

A VWV can also be limited to a subset of documents accompanied with metadata and progressively augmented with new documents as metadata becomes available. This strategy might encourage authors and artifact creators to describe their documents with established standards if they wish their documents to be accessible in a VWV-like system. In our experiment presented in Section 3.5 of Chapter 3, we have restricted our VWV to a set of documents for which we had metadata available.

Figure 2.7 shows two types of layer-1 construction software: Extraction tools and translation tools. Web sites with XML-like documents that follow metadata semantic guidelines would just need an XML parser to build the first layer. However, for other sites, documents can either be translated to XML form with translation tools for the parser to process, or extraction tools are used to retrieve the relevant and available information from the documents to be directly added to the first layer. Notice that these tools can be managed and executed on site by the site administrators (or even the document authors) when even they feel necessary or possible. This would avoid unnecessarily overloading the servers to retrieve the needed information.

The layer-1 construction softwares, after being developed and tested, should be released to the information system manager in a regional or local network, which acts as a "*local software robot*" for automated layer-1 construction. Customization may need to be performed on some softwares, such as handling multilingual information, etc., before they can be successfully applied to their local information bases to generate consistent layer-1 local databases. Softwares for upgrading database structures and information transformers should also be released to local information system managers to keep their local layer-1 database upgraded and consistent with others.

2.4.2 Generalization: Formation of higher layers in MLDB

Since local layer-1 databases are all connected via Internet, they collectively form a globally distributed, huge layer-1 database. Although information retrieval can be performed directly on such a global database, performance would be poor if global-wide searches have to be initiated frequently.

Higher layered databases are constructed on top of the layer-1 database by generalization techniques. Generalization reduces the size of the global database, makes it less distributed (by replicating the smaller, higher-layer databases at, for example, network backbone sites or local server sites), while still preserving the general descriptions of the layer-1 data.

Clearly, successful generalization becomes a key to the construction of higher layered databases. Following studies on attribute-oriented induction for knowledge discovery in relational databases [119, 123], an attribute-oriented generalization method has been proposed for the construction of multiple layered databases [124]. According to this method, data in a lower layer relation are generalized, attribute by attribute, into appropriate higher layer concepts. Different lower level concepts are generalized into the same concepts at a higher level and are merged together, which reduces the size of the database.

We examine in detail the generalization techniques for the construction of higher layered databases.

Concept generalization

Nonnumeric data (such as keywords, index, etc.) is the most popularly encountered type of data in the global information base. Generalization on nonnumerical values should rely on the concept hierarchies which represent necessary background knowledge that directs

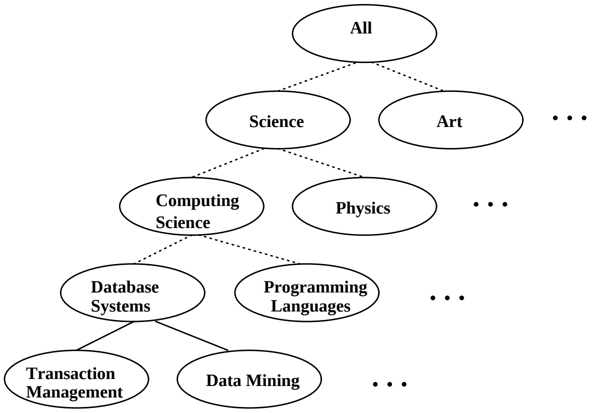


Figure 2.8: A possible concept hierarchy for *keywords*

generalization. Using a concept hierarchy, primitive data is expressed in terms of generalized concepts in a higher layer.

Concept hierarchies are provided explicitly by domain experts or stored implicitly in the database. For the global MLDB, a set of relatively stable and standard concept hierarchies should be provided as a common reference by all the local databases in their formation of higher layered databases and in their browsing and retrieval of information using different levels of concepts.

The concept hierarchies for keywords and indexes can be obtained by referencing a standard concept hierarchy catalogue which specifies the partial order of the terms frequently used in the global information base.

A portion of the concept hierarchy for *keywords* that we used in our experiments described in Section 3.5 is illustrated in Figure 2.8, and the specification of such a hierarchy and alias is in Figure 2.9. Notice that a **contains**-list specifies a concept and its immediate subconcepts; and an **alias**-list specifies a list of synonyms (aliases) of a concept, which avoids the use of complex lattices in the "hierarchy" specification. The introduction of alias-lists allows flexible queries and helps dealing with documents using different terminologies and languages. Also, the dashed lines between concepts in Figure 2.8 represent the possibility to have other layers of concepts in between.

Such a concept hierarchy is either provided by domain experts, or constructed as follows:

1. Collect the frequently used words, technical terms and search keys for classification.

2. Build up a skeleton classification hierarchy based on the technical term specification

All	**contains**:	Science, Art, ...
Science	**contains**:	Computing Science, Physics, Mathematics, ...
Computing Science	**contains**:	Theory, Database Systems, Programming Languages, ...
Computing Science	**contains**:	database systems, Programming Languages, ...
Computing Science	**alias**:	Information Science, Computer Science, Computer technologies, ...
Theory	**contains**:	Parallel Computing, Complexity, Computational Geometry, ...
Parallel Computing	**contains**:	Processors Organization, Interconnection Networks, PRAM, ...
Processor Organization	**contains**:	Hypercube, Pyramid, Grid, Spanner, X-tree, ...
Interconnection Networks	**contains**:	Gossiping, Broadcasting, ...
Interconnection Networks	**alias**:	Intercommunication Networks, ...
Gossiping	**alias**:	Gossip Problem, Telephone Problem, Rumor, ...
Database Systems	**contains**:	Data mining, transaction management, query processing, ...
Database Systems	**alias**:	Database technologies, Data management, ...
Data mining	**alias**:	Knowledge discovery, data dredging, data archaeology, ...
Transaction management	**contains**:	concurrency control, recovery, ...
Computational Geometry	**contains**:	Geometry Searching, Convex Hull, Geometry of Rectangles, Visibility, ...
...		

Figure 2.9: Specification of hierarchies and aliases extracted from an experimental concept hierarchy for computer science related documents.

standards in each field, such as *ACM Computing Review: Classification System for Computing Reviews*, etc. Notice that most of such classification standards are on-line documents.

3. Consult on-line dictionaries (such as Webster dictionary and thesaurus, etc.) for automatically attaching the remaining words to appropriate places in the hierarchy (which may need some human interaction).

4. Consult experts in the fields to make sure that the hierarchy is reasonably complete and correct.

5. Incrementally update such a hierarchy, when necessary, due to the introduction of new terminologies.

In the preliminary experiment described in Section 3.5 of Chapter 3, the used concept hierarchy (also shown in Figure 2.9) was built manually using the set of all keywords extracted from our document collection. We had also first hand experience automatically building a concept hierarchy for the MultiMediaMiner project described in Chapter 5. The hierarchy shown in Figure 5.2 was built using WordNet semantic network [263, 25], a collection of more than 95,000 English words with their relationships, commonly used in cognitive science and computational linguistics.

Algorithm 2.4.1 Creating a concept hierarchy of recognized keywords using WordNet semantic Network.

Input: (i) List of keywords $\mathcal{L}$kw; (ii) List of domain specific terms and phrases *domain*, (iii) enriched WordNet $EWordNet$.

Output: (i) List of rejected keywords $\mathcal{R}$, (ii) Concept hierarchy $\mathcal{CH}$.

Method. For all given words, accept only those that are domain specific, recognized in WordNet or their canonical form is recognized by WordNet. Organize the accepted words in a hierarchy given the parent-child relationship in WordNet. The pseudo-code for creating the keyword hierarchy is as follows:

```
begin
    $\mathcal{R} \leftarrow \emptyset$ ; $\mathcal{L} \leftarrow \emptyset$
    foreach word in $\mathcal{L}$kw do {
        if (word $\in$ $domain$) add word to $\mathcal{L}$; next word
        accept $\leftarrow$ $false$
        CanonicalForms $\leftarrow$ $MorphologicalAnalysis$(word)
        foreach form in CanonicalForms do {
            if (form $\in$ $EWordNet$) add form to $\mathcal{L}$; accept $\leftarrow$ $true$
        }
        if ($\neg$ accept) add word in $\mathcal{R}$
    }
    foreach word in $\mathcal{L}$ do {
        if (word $\notin$ $\mathcal{CH}$)
            parent $\leftarrow$ lookup(ParentOf(word),$EWordNet$)
            while parent $\notin$ $\mathcal{CH}$ do {
                descendant $\leftarrow$ parent; parent $\leftarrow$ lookup(ParentOf(descendent),$EWordNet$)
            }
            $\mathcal{D} \leftarrow$ GetChildren(parent, $\mathcal{CH}$
            add word to $\mathcal{CH}$
            draw arc from parent to word in $\mathcal{CH}$
            foreach descendent in $\mathcal{D}$ do {
                if (IsParent(word, descendent, $EWordNet$)
                    remove arc from parent to descendant in $\mathcal{CH}$
                    draw arc from word to descendent in $\mathcal{CH}$
(24)            endif
(25)        }
(26)    endif
(27) }
end                                                   □
```

Line 1 to 10 are the cleaning step retaining in $\mathcal{L}$ only recognized words. The rejected words are put in $\mathcal{R}$ (line 9) which is later consulted by an ontology expert who would either discard the words or consider them new domain related terms by adding them manually to the concept hierarchy and WordNet for future runs. *MorphologicalAnalysis* is a procedure that extracts all possible forms and declinations of a given term. Line 11 to 27 are the hierarchy tree building.

The constructed concept hierarchies are replicated to each server participating in the VWV, together with the higher layered databases, for information browsing and resource discovery. Managing very large concept hierarchies is a challenge. An efficient encoding taxonomies and managing of dynamic partial orders techniques for reasoning with taxonomies (concept hierarchies, lattices, or complex semantic networks) in computer applications have been proposed in [86, 87].

Generalization on numerical attributes is performed in a more automatic way by the examination of data distribution characteristics [5, 96, 68]. In many cases, it may not require any predefined concept hierarchies. For example, the size of document can be clustered into several groups, such as {*below 10Kb, 10Kb-100Kb, 100Kb-1Mb, 1Mb-10Mb, over 10Mb*}, according to a relatively uniform data distribution criteria or using some statistical clustering analysis tools. Appropriate names are assigned to the generalized numerical ranges, such as {*tiny-size, small-size, middle-size, large-size, huge-size*} to convey more semantic meaning.

With the availability of concept hierarchies, generalization can be performed to produce the strata of the MLDB structure.

Attribute-oriented generalization

Data generalization refers to generalizing data within an attribute in a relational tuple, such as merging generalized data within a set-valued data item, whereas relation generalization refers to generalizing a relation, which often involves merging generalized, identical tuples in a relation.

By removing nongeneralizable attributes (such as long text data, etc.) and generalizing data in other attributes into a small set of values, some different tuples may become identical at the generalized concept level and can be merged into one. A special attribute, *count*, is associated with each generalized tuple to register how many original tuples have been generalized into the current one. This process reduces the size of the relation to be stored in a generalized database but retains the general description of the data of the original database at a high concept level. Such a summarized view of data may facilitate high-level information browsing, statistical study, and data mining.

With data and relation generalization techniques available, the next important question is how to selectively perform appropriate generalizations to form useful layers of databases. In principle, there could be a large number of combinations of possible generalizations by selecting different sets of attributes to generalize and selecting the levels for the attributes to reach in the generalization. However, in practice, a few layers containing most frequently referenced attributes and patterns are sufficient to balance the implementation efficiency and practical usage.

Frequently used attributes and patterns are determined before generation of new layers of an MLDB by the analysis of the statistics of query history or by receiving instructions from users and experts. It is wise to remove rarely used attributes but retain frequently referenced ones in a higher layer. Similar guidelines apply when generalizing attributes to a more general concept level. For example, for a document, the further generalization of layer-1 *document* to layer-2 *doc_brief* can be performed by removing the less frequently inquired attributes *table_of_contents, first_paragraphs*, etc.

Notice that a new layer could be formed by performing generalization on one relation or on a join of several relations based on the selected, frequently used attributes and patterns. Generalization [119] is performed by removing a set of less-interested attributes, substituting the concepts in one or a set of attributes by their corresponding higher level concepts, performing aggregation or approximation on certain attributes, etc.

Since most joins of several relations are performed on their key and/or foreign key attributes, whereas generalization may remove or generalize the key or foreign key attributes of a data relation, it is important to distinguish between the following two classes of generalizations.

1. key-preserving generalization, in which all the key or foreign key values are preserved.

2. key-altering generalization, in which some key or foreign key values are generalized, and thus altered. The generalized keys should be marked explicitly since they usually cannot be used as join keys at generating subsequent layers.

It is crucial to identify altered keys since, if the altered keys were used to perform joins of different relations, it may generate incorrect information [124]. Notice that a join on generalized attributes, though undesirable in most cases, could be useful if the join is to link the tuples with *approximately* the same attribute values together. For example, to search documents, one may like to consider some closely related but not exactly the same subjects. Such kind of join is called an approximate join to be distinguished from the precise join.

Usually, only *precise join* is considered in the formation of new layered relations using joins because approximate join may produce huge sized joined relations and may also be misleading in the semantic interpretation at different usages. However, approximate join will still be useful for searching some weakly connected concepts in a resource discovery query.

An MLDB construction algorithm

Based on the previous discussion, the construction of an MLDB can be summarized into the following algorithm, which is similar to attribute-oriented generalization in knowledge discovery in databases [119].

Algorithm 2.4.2 Construction of an MLDB.

Input: A global information base, a set of concept hierarchies, and a set of frequently referenced attributes and frequently used query patterns.

Output: A multiple layered database (MLDB) abstracting a given subset of the WWW.

Method. A global MLDB is constructed in the following steps.

1. Determine the multiple layers of the database based on the frequently referenced attributes and frequently used query patterns.

2. Starting with the global information base (layer-0), generalize the relation step-by-step (using the given concept hierarchies and generalized schema) to form multiple layered relations (according to the layers determined in Step 1).

3. Merge identical tuples in each generalized relation and update the *count* of the generalized tuple.

4. Construct a new schema by recording the definitions of all the generalized relations, their relationships and the generalization paths.

Rationale of Algorithm 2.4.2.

Step 1 indicates that the layers of an MLDB should be determined based on the frequently referenced attributes and frequently used query patterns. This is reasonable since to ensure the elegance and efficiency of an MLDB, only a small number of layers should be constructed, which should provide maximum benefits to the frequently accessed query patterns. Obviously, the frequently referenced attributes should be preserved in higher layers, and the frequently referenced concept levels should be considered as the candidate concept levels in the construction of higher layers. Steps 2 and 3 are performed in a way similar to the attribute-oriented induction, studied previously [119, 124]. Step 4 constructs a new schema which records a route map and the generalization paths for information browsing and knowledge discovery. □

Example 2.4.1 A portion of relation *doc_brief* is presented in Table 2.2.

file_addr	authors	title	publication	pub_date	key_words	···
http://fas.sfu.ca/9/cs/research /projects/HMI-5/documents /papers/han/coop94.ps.gz	J. Han Y. Fu R. Ng	Cooperative Query Answering Using Multiple Layered Databases	Proc. 2nd Int'l Conf. Cooperative Info. Systems	May 1994	data mining, multiple layered database, ···	···
···	···	···	···	···	···	···
ftp://ftp.cs.colorado.edu /pub/cs/techreports /schwartz/FTP.Caching-PS	P.B.Danzig R.S.Hall M.F.Schwartz	A Case for Caching File Objects Inside Internetworks	Proc. SIGCOMM	Sept. 1993	caching, ftp, ···	···
···	···	···	···	···	···	···
http://sobolev.mit.edu/people /jphill/publications/shap.dvi	J.R.Phillips H.S.J. Zant	Influence of induced magnetic fields on Shapiro steps in Josephson junction arrays	Physical Review B 47	1994	magnetic fields, Josephson array, Shapiro step, ···	···
···	···	···	···	···	···	···

Table 2.2: A portion of *doc_brief* extracted from *document* at layer-1.

By extraction of only the documents related to *computing science*, a layer-3 relation *cs_doc* can be easily obtained. Also, performing attributed-oriented induction on *doc_brief* leads to another layer-3 relation *doc_summary*, a portion of which is shown in Table 2.3.

Notice that backward pointers can be stored in certain entries, such as *first_author_list* and *file_addr_list*, in the *doc_summary* table, and a click on a first author or a file_address will lead the presentation of the detailed corresponding entries stored in layer-2 or layer-1. □

affiliation	field	pub_year	count	first_author_list	file_addr_list	···
Simon Fraser Univ.	Database Systems	1994	15	Han, Kameda, Luk, ···	···	···
Univ. of Colorado	Global Network Systems	1993	10	Danzig, Hall, ···	···	···
MIT	Electromagnetic Field	1993	53	Bernstein, Phillips, ···	···	···
···	···	···	···	···	···	···

Table 2.3: A portion of *doc_summary* extracted from *doc_brief* at layer-2.

2.4.3 Distribution and maintenance of the global MLDB

Replication and distribution of the global MLDB

A global MLDB is constructed by extracting extra-layers from an existing (layer-0) global information base using generalization and transformation techniques. A higher layer database is usually much smaller than the lower layered database. However, since the layer-1 database is resulted from direct, detailed information extraction from the huge global information base, its size is still huge. It is unrealistic to have this layer replicated and distributed to other servers. A possible implementation is to store each local layer-1 database at each local network server site, but to replicate the higher layered databases, such as layer-2 and above, and propagate them to remote backbone and/or ordinary network servers. Load can be further partitioned between backbone and ordinary servers. For example, one may store a complete layer-2 database at the backbone site but only the relatively frequently referenced portions of layer-2 and/or higher layers at the corresponding sites. Also, specifically projected layers (e.g., medical database) can be stored at the closely relevant sites (e.g., hospitals and medical schools). By doing so, most information browsing and brief query answering can be handled by searching within the local network. Only detailed requests will be forwarded to the backbone servers or further to the remote sites which store the information. Only when the full document is explicitly requested by a user (with the awareness of its size), will the full layer-0 document be sent across the network to the user site. This will substantially reduce the amount of data to be transmitted across the network and thereby improve the response time.

Moreover, some higher layered databases could be defined by users for easy reference. For example, a user may define a new database at a high layer as "*all the documents related to heterogeneous databases published in major conferences or journals since 1990*". An information manager cannot construct a new database for every user's definition. Most such definitions will be treated like views, i.e., no physical databases will be created, and queries on such views will be answered by the query modification technique [79, 152]. Only if such a view is shared and frequently referenced, may it be worthwhile to create a new database for it.

Incremental updating of the global MLDB

The global information base is dynamic, with information added, removed and updated constantly at different sites. It is very costly to reconstruct the whole MLDB database. Incremental updating could be the only reasonable approach to make the information updated and consistent in the global MLDB.

In response to the updates to the original information base, the corresponding layer-1 and higher layers should be updated incrementally. Incremental update can be performed on every update or at regular times at the local site and propagate the updates to higher layers.

We only examine the incremental database update at insertion and update. Similar techniques can be easily extended to deletions. When a new file is connected to the network, a new tuple t is obtained by the layer-1 construction algorithm. The new tuple is inserted into a layer-1 relation R_1. Then t should be generalized to t' according to the route map and be inserted into its corresponding higher layer. Such an insertion will be propagated to higher layers accordingly. However, if the generalized tuple t' is equivalent to an existing tuple in this layer, it needs only to increment the count of the existing tuple, and further propagations to higher layers will be confined to count increment as well. When a tuple in a relation is updated, one can check whether the change may affect any of its high layers. If not, do nothing. Otherwise, the algorithm will be similar to the deletion of an old tuple followed by the insertion of a new one.

2.5 Reflections on Mediating Virtual Web Views

While in theory it is possible to create a unique global virtual web view that would summarize and represent the entire content of the World-Wide Web, it is neither practical nor desirable. A VWV is based on concept hierarchies and it is very difficult to find a consensus on a general ontology[11]. It is more realistic to build different VWVs specializing in different

[11] Some general ontologies are being developed in specific domains or applications such as for electronic commerce: http://www.ontology.org.

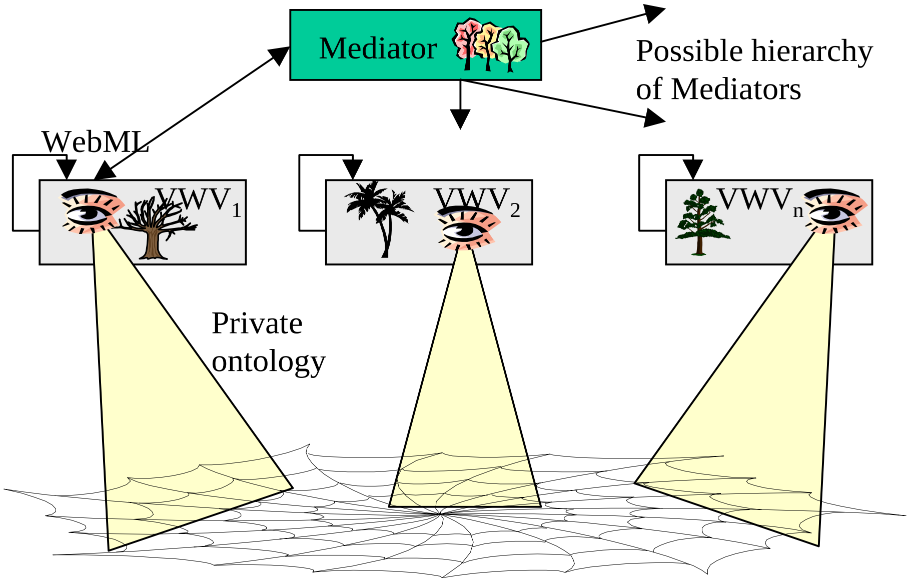


Figure 2.10: Mediating Virtual Web Views.

topics or restricted geographically, etc. VWVs can also share the same primitive data but use different ontologies (i.e. concept hierarchies). In such a context, a user of a given VWV may want to access data not visible but available through other VWVs. The user may not even know of the existence of these other VWVs. This is an information gathering problem. Each VWV is using a private ontology to generalize the different strata of the MLDB structure and thus the distributed VWVs are heterogeneous data sources. An entity which could transparently translate information requests and integrate the different answers, is needed. Such entity is typically called an *agent*. There is no general agreement on what an agent software should be or be capable of (see Section A.2.4 in Appendix A). In the context of VWVs, we would like to delegate to this information agent the task of replying to a query in an understandable way, when a given VWV is unable to provide the answer. We call such information gathering agent a Mediator since it plays the role of an intermediary between VWVs. Basically, it manages, and possibly translates, information exchange between VWVs. When a requester submits a query to the mediator, there could be many VWVs that could answer it. The mediator also plays the role of a Matchmaker choosing the source which would best answer a request. A mediator would be aware of the existence of some VWVs with which it communicates (i.e. answers their requests and accesses their data). A hierarchy of different mediators could exist where mediators delegate to each other

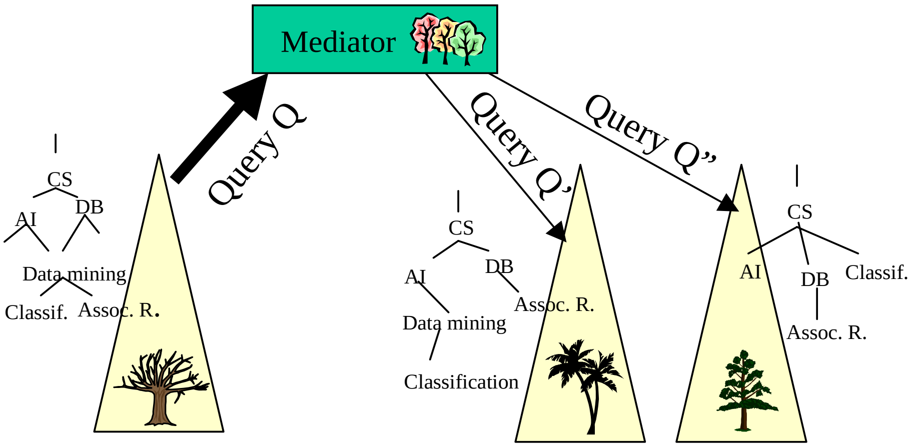


Figure 2.11: Mediation Scenarios.

requests and sub-queries (Figure 2.10). A mediator can also play the role of a Broker by creating execution plans of queries when requests have to be broken down to sub-queries and submitted to different VWVs (Figure 2.11).

Being heterogeneous, in the sense that each VWV is using a different set of concept hierarchies, the ideal model would be to have a global knowledge representation used by the mediator, and a wrapper around each VWV for interoperability with the mediator. A wrapper transforms queries from the VWV representation to the mediator's one, and converts the mediator's communications to a format understandable by the VWV. However, until a global and agreed upon ontology is available, we propose to give the mediator the role of a wrapper agent (or Elucidator) that rewords queries and answers to forms understandable by each party it communicates with. To do so, a mediator keeps a table of all VWVs it is aware of and their respective ontologies.

Without discussing the wrapper function that translates queries and answers based on different ontologies, we present some possible scenarios for mediating virtual web views:

Scenario 1: Matchmaking with no translation

1 VWV_0 Sends a query Q_0 to mediator

2 Mediator transmits Q_0 to VWV_α

3 Mediator receives answer A_α from VWV_α

4 Mediator answers VWV_0 with A_α and ontology of VWV_α

5 VWV_0 interprets A_α

In the first scenario, the mediator simply passes on the query of a requester to a VWV that might have the answer. The answer is then returned to the requester with the ontology of the VWV that answered the query. No translations are done. The requester has the responsibility to interpret the answer. The role of the Mediator in this case is just a matchmaker finding the best source to answer the request.

Scenario 2: Mediating with translation
1 VWV_0 Sends a query Q_0 to mediator
2 Mediator translates Q_0 to Q_α to fit ontology of VWV_α
3 Mediator sends Q_α to VWV_α
4 Mediator receives answer A_α from VWV_α
5 Mediator translates A_α to A_0 to fit ontology from VWV_0
6 Mediator answers VWV_0 with A_0
7 VWV_0 receives A_0

With the second scenario, the mediator has the task to translate the query before submitting it to the data source (VWV_α), and to translate the answer before returning it to the requester (VWV_0). In other words, the mediator holds a wrapper for VWV_0 and VWV_α to translate and transform communications.

In the third scenario, the mediator trades with more than one VWV for the same request. It carries out more sophisticated planning by partitioning the requester's query into sub-queries, submitting the sub-queries to different data sources, and integrating the results before returning the answer to the requester. The mediator has to coordinate between many heterogeneous data sources. It maintains a list of service providers (i.e. VWVs) and their capabilities (i.e. ontologies). The query at hand is broken down according the available service providers' capabilities, translated, and sent to the providers in an appropriate sequence. This may need careful distributed query planning and query optimization.

Scenario 3: Mediating with Planning and translation
1 VWV_0 Sends a query Q_0 to mediator
2 Mediator expresses Q_0 into $Q_1, Q_2, ..., Q_n$
3 Mediator sends $Q_1, Q_2, ..., Q_n$ to $VWV_1, VWV_2, ..., VWV_n$
4 Mediator receives answer $A_1, A_2, ..., A_n$ from $VWV_1, VWV_2, ..., VWV_n$
5 Mediator integrates $A_1, A_2, ..., A_n$ into A_0 using available ontologies

```
6    Mediator answers $VWV_0$ with $A_0$
7    $VWV_0$ receives $A_0$
```

2.6 Discussion

Virtual Web Views provide the following advantages for information discovery in global information systems.

1. Application of database technology: The Multiple Layered Database architecture transforms an unstructured global information base into a structured, global database, which makes the database technology (not just storage management and indexing techniques) applicable to resource management, information retrieval, and knowledge discovery in the global information network.

2. High-level, declarative interfaces and views: The architecture provides a high-level, declarative query interface on which various kinds of graphics user-interfaces can be constructed for browsing, retrieval, and discovery of resource and knowledge. Moreover, multiple views can be defined by different users or user communities, cross-resource linkages can be constructed at different layers, and resource search can be initiated flexibly.

3. Performance enhancement: The layered architecture confines most searches to local or less remote sites on relatively small and structured databases, which will reduce the network bandwidth consumption, substantially enhance the search efficiency, and lead to relatively precise locating of resources and quick response of user's requests.

4. A global view of database contents: By preprocessing and generalizing primitive data, a VWV may transform semantically heterogeneous, primitive level information into more homogeneous, high-level data at a high layer. It may provide a global view of the current contents in a database with summary statistics, which will assist users to browse database contents, pose progressively refined queries, and perform knowledge discovery in databases. Users could even be satisfied with the general or abstract data at a high layer and not bother to spend time and network bandwidth for more details.

5. Intelligent query answering and database browsing: A user may not always know the exact need when searching in the global information base. With a VWV, a query is

treated like an information probe, being mapped to a relatively high concept layer and answered in a hierarchical manner. This will provide users with a high-level view of the database, statistical information relevant to the answer set, and other associative and summary information at different layers. See Chapter 3 for details about WebML the query language we propose for the MLDB structure.

6. Information resource management: Incremental updating can be performed on different layers using efficient algorithms, as discussed in Section 3. With the MLDB architecture, it is relatively easy to manage the global MLDB and make it consistent and up-to-date. For example, it is easy to locate weakly-consistent replicas [66] based on their property similarity at higher layers (rather than searching through the whole global information base!). Based on accessing statistics, one can also decide whether a duplicate should be removed or be preserved for resource redirection.

2.7 Conclusions

Different from the existing global information system services, a new approach, called *virtual web views (VWV)* using *multiple layered database (MLDB)* structure, has been proposed and investigated for resource and knowledge discovery in global information systems. The approach is to construct progressively a global multiple layered database by generalization and transformation of lower layered data, store and manage multiple layered information by database technology, and perform resource and knowledge discovery by query transformation, query processing and data mining techniques. The Virtual Web View plays the role of a data warehouse for web content.

The major strength of the VWV approach is its promotion of a tight integration of database and data mining technologies with resource and knowledge discovery in global information systems. With the dynamically growing, highly unstructured, globally distributed and huge information base, the application of the mature database technology and promising data mining techniques could be an important direction to enhance the power and performance of global information systems.

Our study shows that the web data warehousing can be performed and updated incrementally by integration of information retrieval, data analysis and data mining techniques, information at all of the non-primitive layers can be managed by database technology, and resource and knowledge discovery can be performed efficiently and effectively in such a multiple layered database.

Enforcing a consistent standard for metadata on the Internet will simplify data exchange and the effective information extraction from on-line documents. XML and the Dublin Core initiative are new standards endorsed by many organizations. With these standards web data warehousing can start with a niche of documents and progressively add new artifacts when these standards are more widely used.

Discovery consists of seeing what everybody has seen and thinking what nobody has thought.
Albert von Szent-Gyorgyi

More grows in the garden than the gardener has sown.
Unknown

Chapter 3

Querying the Web for Resources and Knowledge

More than half a century ago, in a paper in which he describes the “Memex”, a system for storing and organizing multimedia information, Vannevar Bush invited researchers to join the effort in building an information system for holding the human knowledge, and making it easily accessible[44]. He writes: “A record, if it is to be useful... must be continuously extended, it must be stored, and above all it must be consulted.” A massive aggregation of documents is now stored on the Internet. Some consider it the biggest database ever built. The World-Wide Web is holding a colossal collection of resources, from structured records, images and programs to semi-structured files and free text documents. The availability of information is not questionable. We are actually overwhelmed by this excess of information. Accessibility as described by Vannevar Bush however is still unsolved. For many decades, information retrieval from document repositories has drawn much attention in the research community [218, 142]. Information retrieval has been a fertile research field. Many techniques have been proposed and implemented in successful and less prevailing applications. With the advent of the World-Wide Web, the appearance of a panoply of services and accumulation of a colossal aggregate of resources, information retrieval techniques have been adapted to the Internet, bringing forth indexing models and search engines. Today several search engines are used daily by millions of users. However, the effectiveness of these tools is not satisfactory and is even irritating (See Appendix A). The annoying results of current search engine technologies have invited researchers to tackle new challenges. Better indexing approaches, specialized information gathering agents, filtering and clustering methods, etc. have since been proposed.

A new research trend in the field of information retrieval from the World-Wide Web is web querying [276, 161, 183, 149, 19] and the design of query languages for semi-structured data [205, 1, 20]. The approach for querying structured and semi-structured documents involves the construction of tailored wrappers that map document features into instances in internal data models (i.e. graphs or tables). The introduction of new types of documents usually necessitates the construction of new custom-made wrappers to handle them. Due to the semi-structured nature of web pages written in HTML, the migration of semi-structured data query languages like UnQL[41] and Lorel[1] to the World-Wide Web domain is evident. W3QL[149], WebLog[161], WebSQL[183] and WebOQL[19] are all intended for information gathering from the World-Wide Web. While WebLog and WebOQL aim at restructuring web documents using Datalog-like rules or graph tree representations, WebSQL and W3QL are languages for finding relevant documents retrieved by several search engines in parallel. However, none of these approaches takes advantage of the structure of the global information network as a whole. Moreover, none of these languages performs data mining from the Web. A language like WebSQL is built on top of already existing search engines which lack precision and recall. Indeed WebSQL has the same strategy as Metacrawler which submits a request simultaneously to several search indexes[224]. W3QS, the system using W3QL, also uses existing search engines. A web document structuring language like WebOQL or WebLog is capable of retrieving information from on-line news sites like CNN, tourist guides, or conference lists, but is limited to a subset of the web defined in the queries. Their powerful expressions, however, can extract interesting and useful information from within a given set of web pages. We intend to use this power to build our system's data model. We propose a web query language, WebML that permits resource discovery as well as knowledge discovery from a subset of the Internet or the Internet as a whole. WebML is an SQL-like declarative language for web mining. We have introduced new primitives that we believe make the language simple enough for casual users. These primitives allow powerful interactive querying with an OLAP (OnLine Analytical Processing)-like interaction (i.e. drill-down, roll-up, slice, dice, etc.). The language takes advantage of a Multi-Layered DataBase (MLDB) model[128, 276], presented in Chapter 2, in which each layer is obtained by successive transformations and generalizations performed on lower layers, the first layer being the primitive data from the Internet. The higher strata are stored in relational tables and take advantage of the relational database technology. Their construction is based on a propagation algorithm and assumes the presence or availability of descriptive metadata, either provided by document authors through tags and XML-based descriptions, or extracted by tools like WebLog, WebOQL, and some Web agents.

In the remainder of the Chapter, we review some relevant query languages before introducing WebML. Since WebML is an SQL-like declarative language and its syntax in derived from it, we start briefly reviewing SQL and QBE as examples of relational database query languages. We also examine some examples of World-Wide Web query and restructuring languages and present a data mining language DMQL that we proposed for data mining from relational databases. DMQL is relevant in this context since WebML heavily borrows semantic and syntactic constructs from DMQL for its data mining capabilities. WebML is later introduced with examples for resources discovery as well as knowledge discovery from the Internet. Finally, an implementation experiment is described.

3.1 Relevant Query Languages for Data and Information Retrieval

Query languages are means to access data, generally stored in databases, or repositories of some sort. They can be for strict retrieval of data, or manipulation of the data stored and to be stored. Database query languages have been investigated for many years and there are numerous languages specializing in different domains and structures based on the data they access or manipulate, or based on the database model used.

A query is a statement defining some constraints on the data to be retrieved or manipulated. Languages expressing queries can either be visual (like QBE), or in the form of a programming language. In the following section, we review some specific query languages in the relational database domain, in the World-Wide Web field, and in data mining.

3.1.1 Relational Database Languages

There are many relational database query languages, some more influential than others. All are based on strong mathematical foundations. SQL [16] is, without a doubt, the relational query language standard today. However, there are other commercial languages, like QBE, QUEL, Datalog, etc. [228]. While SQL uses a combination of relational algebra and relational-calculus constructs, QBE is based on domain relational calculus, QUEL is based on the tuple relational calculus, and Datalog is based on the logic-programming language Prolog [228]. In this sub-section, more emphasis will be given to SQL and QBE as examples of relational query languages.

Structured Query Language (SQL)

SQL is an acronym of Structured Query Language and was originally called SEQUEL-2 (SQL is still pronounced “sequel”), derived from SEQUEL which was introduced as the query language for the relational database management system *System/R* in the 1970s. The language deals exclusively with relations, also called tables, and its underlying data model is the relational model presented by Codd in 1970 [58].

SQL is divided into two distinctive parts: the DDL (Data Definition Language) for defining the data structure, and the DML (Data Manipulation Language) for retrieving and updating the data. However, we will emphasize more the DML part, since it is more related to WebML, the language we define later, and we refer to SQL as the data manipulation language of SQL.

The basic structure of SQL consists of the **SELECT-FROM-WHERE** clauses. The **SELECT** clause determines the attributes and aggregates to display. It corresponds to the project operation of the relational algebra (Π). The **FROM** clause describes the data sets from which to retrieve the data. This data set can be a relation or a set of relations. This clause corresponds to the cartesian-product operation of the relational algebra (×). The **WHERE** clause, which corresponds to the selection predicate of the relational algebra (σ), specifies the constraints upon the data to retrieve.

The following SQL example retrieves titles and on-line addresses of all documents published after 1995 and where “zaiane” appears in the author:

```
SELECT d.title,d.url
FROM Document d
WHERE d.author LIKE “%zaiane%” AND
        d.date.year > 1995
```

Used in an interactive SQL environment, this query would return a result displayed in a tabular form. Indeed, the result of this query is a relation with two columns: one for the title, and one for the URL. The number of rows depends upon the records in "*document*" that satisfy the conditions stated in the **WHERE** clause.

SQL supports aggregate functions such as **avg**, **min**, **max**, **sum**, and **count** for the calculation of the average, minimum, maximum, the total sum, and the count of attributes, and has additional clauses such as **GROUP BY**, **HAVING**, and **ORDER BY** for, respectively, grouping rows by attribute values, filtering groups under some conditions, and ordering the output. One powerful peculiarity of SQL is the possibility to "nest" queries together. A *nested-query* is a query that contains another query (or nested-query) in its **WHERE** clause.

The following SQL example retrieves titles and on-line addresses of all documents published after 1995 and authored by someone from SFU who also authored a publication by ACM, grouped by the year of publication.

```
SELECT d.title,d.url
FROM Document d
WHERE d.date.year > 1995 AND
        d.author IN (SELECT d2.author
                     FROM document d2, Person p
                     WHERE d2.author = p.name AND
                           p.institution = "SFU" AND
                           d2.publisher = "ACM")
GROUP BY d.date.year
```

It is important to note that SQL can be used in an interactive way, but can also be embedded into programs. It has been designed for use within general-purpose programming languages, such as C/C^{++}, Pascal, Fortran, Cobol, Perl, etc. It is, arguably, considered a programming language for data manipulation in database systems.

There are many extensions and "flavours" of SQL. Some extensions were added to handle complex data types like set-valued attributes, multimedia (images, maps, sounds, videos, etc.), and even multi-dimensional data cubes.

Query By Example (QBE)

QBE is the database query language of the QBE database system developed at IBM T.J. Watson Research Center in the 1970s. The peculiarity of QBE is that unlike SQL and other languages, which require that the user knows the database schema, QBE provides the user with the schema in an empty skeleton table form. The query consists of filling in the columns of the skeleton tables with the desired conditions and commands.

For example, retrieving titles and on-line addresses of all documents published after 1995 and where "zaiane" appears in the author, requires filling out the form as follows:

Document	*title*	*author*	*Date*	*URL*
	P. _Title	zaiane	> 1995	P. _URL

QBE has the same expressive power as SQL. However, its two-dimensional syntax makes it particularly user friendly. Many Web-based database query interfaces have adopted this approach using Web-based forms, even though the underlying database system uses SQL.

3.1.2 Querying the World-Wide Web

The emergence of the World-Wide Web and its increasing popularity for dissemination of information has made it a new target for query language design. Given the chaotic structure of the Web and its documents, designing a query language for it is challenging. Nevertheless, many attempts have been made: some query languages have been proposed, borrowing from relational query languages and semi-structured data query languages. These languages aim at retrieving data (or resources) from the Web given constraints on content and structure, and at modeling and restructuring the Web by constructing new resources that bring together otherwise scattered information. Examples of these languages are: W3QL [149], WebLog [161], UnQL [41], WebSQL [183], Lorel [1], WebOQL [19], STRUDEL [95], etc. In the following, we will present four of them: WebSQL, W3QL, WebLog and WebOQL.

WebSQL

WebSQL [183] was developed at the University of Toronto. It is an SQL-like language. However, it does not use relations, but rather virtual relations, and counts on existing search engines to reach out for documents. It is considered a first generation Web query language [100] since it combines the content-based queries of search engines (text patterns appearing within documents) with structure-based queries (graph patterns describing link structure). For WebSQL, the Web consists of two virtual relations: *Document(url, title, text, type, length, modif)* where *url* is the address of the document, *title* its title, *text* the document itself, *type* is the MIME type of the document, *length* is the size of the documents, and *modif* is the last modification date, and *Anchor(base, href, label)* where *base* is the URL of a document containing the link in question, *href* is the address pointed by the link, and *label* the link description. This relational abstraction of the Web allows using a relational query language to submit queries. The syntax of WebSQL is much like the SELECT-FROM-WHERE clauses of SQL with some additional keywords and primitives. The concise syntax is as follows:

```
<WebSQL>  ::=
  SELECT <attribute_list>
  FROM <domain> SUCH THAT <domain_conditions> {, <domain> SUCH THAT <domain_conditions>}
  WHERE <conditions>
```

One interesting characteristic of WebSQL is the use of primitives to express hypertext links in the **WHERE** clause and the *domain_conditions*. For instance, internal links, links inside a document, are denoted $\mapsto$. Local links, links to other documents in the same web server, are denoted $\rightarrow$. Finally, global links are denoted $\Rightarrow$. These symbols can be alternated and repeated in regular expression with | and $*$.

As an example, to find the documents mentioning "Data Mining" and linking through paths of length two or less to documents containing a Java applet, the query is as follows:

```
  SELECT x.url, x.title, y.url, y.title
  FROM document x SUCH THAT x MENTIONS "Data Mining",
          document y SUCH THAT x = |→|→→|⇒|⇒⇒ y,
          anchor z SUCH THAT z.base = y
  WHERE z.label CONTAINS "applet";
```

This query is translated into search requests sent to a list of search engines. The output of these engines is merged and further parsed before displaying the final result.

W3QL

W3QL, the query language of W3QS system [149], is also a first generation web query language treating documents as atomic objects containing text patterns and pointers to other atomic objects (i.e. documents). It also uses search engines to access already built indexes. The links to these search engines and their respective query forms are stored in external files and can be updated on the fly. The major difference between W3QL and WebSQL is the possibility to use external programs and unix commands directly within W3QL queries. External programs (written by users) can be used to specify content conditions that cannot be expressed with the query language in order, for example, to parse web pages. These written programs are dynamic extensions of the language. They can be run in the SELECT or the WHERE clause. W3QL also allows the definition of views that can be updated and maintained automatically at regular intervals. The concise grammar of W3QL is as follows:

```
<W3QL>      ::=
   SELECT [CONTINUOUSLY] < select_clause>
   FROM <path>
   WHERE <node> IN <file>| FILL <node> AS IN <file>| RUN <unix_prog> IF <node> UNKNOWN
   [USING <algorithm>]
   [EVALUATED EVERY <time_unit>]
```

As an example, to find the HTML documents containing the string "Market Basket" and titled "Association Rules", the following query is submitted:

```
   SELECT cp n2/* result;
   FROM n1, l1, n2;
   WHERE n1 IN Indexes.url FILL n1.form AS IN Indexes.fil WITH keyword="Market Basket";
   SQLCOND (n2.format=HTML) AND (n2.title="Association Rules");
```

Indexes.url and *Indexes.fil* are two files containing, respectively, the list of search engines to try the query against (search with keyword="Market Basket") and the forms and syntax of these search engines. *SQLCOND* is a program that determines the format of a file (using the unix *file* command) and attributes some recognized patterns to fields. In this example, *n2.title* is identified from the HTML tags <TITLE> ... </TITLE>. The query is submitted to different search engines and the results are analyzed (with SQLCOND) and copied with the unix *cp* command to a file named *result.*

WebLog

Developed at the University of Concordia in Montréal, WebLog[161] is considered a second generation web query language for its ability to manipulate components of documents, and even generate new compositions as result of a query. It uses deductive rules instead of SQL-like syntax like the previous languages. WebLog queries are regarded as programs. The language is inspired from SchemaLog [160] and takes advantage of the expressive power of Datalog and Prolog. With WebLog, a web document is treated as "groups of related information" separated by delimiters. These delimiters are user specified, and are usually HTML tags defining the granularity of the groups, such as paragraphs <P> or lines in lists <LI>, etc. Each of the groups of information is dealt with as an object with attributes such as content (referred to as *occurs*), links (referred to as *hlink*), etc. The language uses built-in predicates to manipulate links and strings, and user defined predicates. It also uses predicates to define string similarity such as synonymy, etc. to help in keyword searching.

To illustrate the sophisticated capabilities of WebLog, we give two examples of queries, one for information retrieval and one for restructuring.

The following Weblog program finds all documents in the SFU site that have information related to "Data Mining" and are linked from Zaïane's web page:

```
sfu_pages(http://www.cs.sfu.ca/~zaiane) ⟵
sfu_pages(U) ⟵ sfu_pages(V), V[hlink↦L], href(L,U), substring(U,http://www.cs.sfu.ca/).
interesting_urls(U) ⟵ sfu_pages(U), U[occurs ↦ T], synonym(T, 'Data Mining').
```

Note that the predicate *sfu_pages* starts with the URL http://www.cs.sfu.ca/~zaiane. This assumes that the user has partial knowledge about where the information he or she is looking for is located. WebLog does not use already built indexes like soliciting existing search engines. It crawls the Web recursively with programmed predicates such as *sfu_pages* above.

Knowing that Zaïane's sports page contains references to competitions in swimming and triathlon around the world, the following WebLog program compiles in a new page called *result.html*, a list of triathlon competitions in British Columbia:

```
traverse(L) ⟵ http://www.cs.sfu.ca/~zaiane/sports.html[occurs↦'Meets', hlink↦L].
traverse(L) ⟵ traverse(M), href(M,U), U[hlink↦L], U[occurs↦'British Columbia'].
```

```
result.html[L:title→‘Competitions in BC’, hlink↦L, occurs↦‘Competition date:’.D.‘General Infor-
mation:’.I] ⟵ traverse(L), href(L,I), I[occurs↦‘triathlon’], I[occurs↦S], substring(S,D), isa(D,date).
```

This program assumes that the sports page contains a link to meets which are ordered geographically. *traverse* traverses all hyperlinks in grouped information containing the keyword “meets”, and stops at pages where the keyword “British Columbia” appears. The result in *result.html* collects information with dates from pages containing the string “triathlon”. We assume these dates as competition dates.

WebLog is powerful and very flexible. However, programming with WebLog assumes preliminary knowledge about the potential location of the information (i.e. initial URL) and the rough structure of the documents. It could be very useful, when the structure and semantic of a web site is known, to extract and summarize some interesting data from the site for the construction, for example, of the first layer of the MLDB Web “warehouse” (see Chapter 2).

WebOQL

WebOQL[19] was developed at the University of Toronto. It is a language to query and restructure hypertexts, structured documents (i.e semi-structured data), or record-based data. Its data model, hypertrees and webs, is capable of abstracting instances from any of these types of data collections. WebOQL manipulates hypertrees and webs, and as a result of a query, generates hypertrees and webs. In the WebOQL model, webs are related hypertrees collected together, while hypertrees are ordered arc-labeled trees with internal and external arcs. Internal arcs are used to stand for structured objects and external arcs are used to represent references to objects (usually hyperlinks). Both types of arcs are labeled with records (see Figure 3.1 for an example). In [19], hypertrees and webs are compared to an internal representation of a compiler. All data collections are translated into this model before processing of queries can proceed.

WebOQL is a functional language, but its syntax has been fitted to the SELECT-FROM-WHERE SQL form and resembles the object-oriented query language OQL. Some operators have been introduced to manipulate trees and arcs. Since WebOQL acts on sub-trees (parts of documents, web sites, or even databases records) and can generate sub-trees, it is considered a second generation web query language. To illustrate the capabilities of the language,

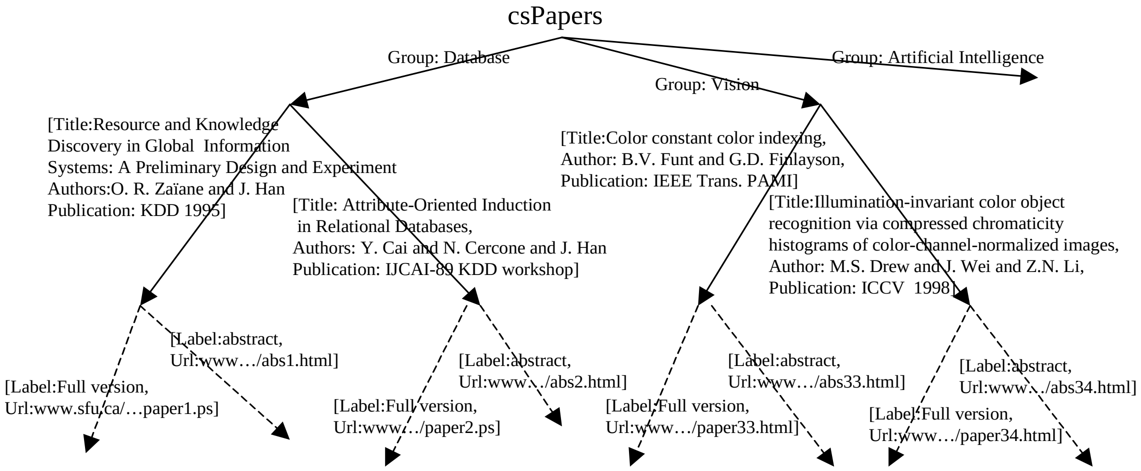

csPapers
Group: Database
Group: Vision
Group: Artificial Intelligence
[Title:Resource and Knowledge Discovery in Global Information Systems: A Preliminary Design and Experiment Authors:O. R. Zaïane and J. Han Publication: KDD 1995]
[Title: Attribute-Oriented Induction in Relational Databases, Authors: Y. Cai and N. Cercone and J. Han Publication: IJCAI-89 KDD workshop]
[Title:Color constant color indexing, Author: B.V. Funt and G.D. Finlayson, Publication: IEEE Trans. PAMI]
[Title:Illumination-invariant color object recognition via compressed chromaticity histograms of color-channel-normalized images, Author: M.S. Drew and J. Wei and Z.N. Li, Publication: ICCV 1998]
[Label:Full version, Url:www.sfu.ca/…paper1.ps]
[Label:abstract, Url:www…/abs1.html]
[Label:Full version, Url:www…/paper2.ps]
[Label:abstract, Url:www…/abs2.html]
[Label:Full version, Url:www…/paper33.html]
[Label:abstract, Url:www…/abs33.html]
[Label:Full version, Url:www…/paper34.html]
[Label:abstract, Url:www…/abs34.html]

Figure 3.1: Hypothetical example of a WebOQL hypertree.

we present two queries for examining and restructuring the web.

Given the hypertree presented in Figure 3.1[1], the following query retrieves the title and URL address of full version papers authored by "Jiawei Han":

SELECT $[y.Title, y\prime.Url]$
FROM x IN csPapers, y IN $x\prime$
WHERE $y.authors$ ~ "Jiawei Han"

The prime operator returns the first sub-tree of its argument. Since x iterates over the groups (top level of the hypertree), the primed variable $x\prime$ iterates over the title (sub-trees of the groups). Priming y reaches the external arcs for the URL addresses. Obviously, the structure of the hypertree has to be known in order to write the query.

By using the AS in the SELECT clause, one can map the hypertree into another to generate a new Web document. The following query performs restructuring.

SELECT $[y.Title, y.Author]$ AS $x.Group$
FROM x IN csPapers, y IN $x\prime$

The examples we gave are straightforward examples from the hypertree in Figure 3.1. WebOQL queries can be arbitrarily complex with nested subqueries.

[1] hypothetical hypertree similar to the one given in [19]

3.1.3 Data Mining Query Language

Believing that the success of the relational databases is partially credited to the standardization of the relational query language (SQL), we have been working on the design of a data mining query language [126] in the hope that it might perhaps become a motivation for others to examine and work on an approved language. The design of a data mining language is a challenging assignment given that data mining covers a wide spectrum of task, from data summarization to mining association rules, data classification, or finding some specific patterns. In the development of a general data mining system, our requirement was a versatile data mining language that covers different data mining methods and activities. We have tentatively designed the Data Mining Query Language, DMQL, which by no means can be claimed complete, but may serve as an interesting example for further discussion. The language has been partially implemented and is used in DBMiner data mining system [125, 120] for the discovery of several kinds of knowledge in relational databases.

Figures 3.2, 3.3, 3.4 and 3.5 show some snapshots of the web interface we implemented for DBMiner system. While the PC implementation is a stand-alone application, the web version uses a client-server architecture. Users select and submit DMQL queries, such as in Figure 3.3, and have the possibility to interactively manipulate the output, graphics, tables, or rules, by drilling down, rolling up, slicing, dicing, or changing the different thresholds. The system can be tried at http://db.cs.sfu.ca/DBMiner.

The design of DMQL adheres to the following principles:

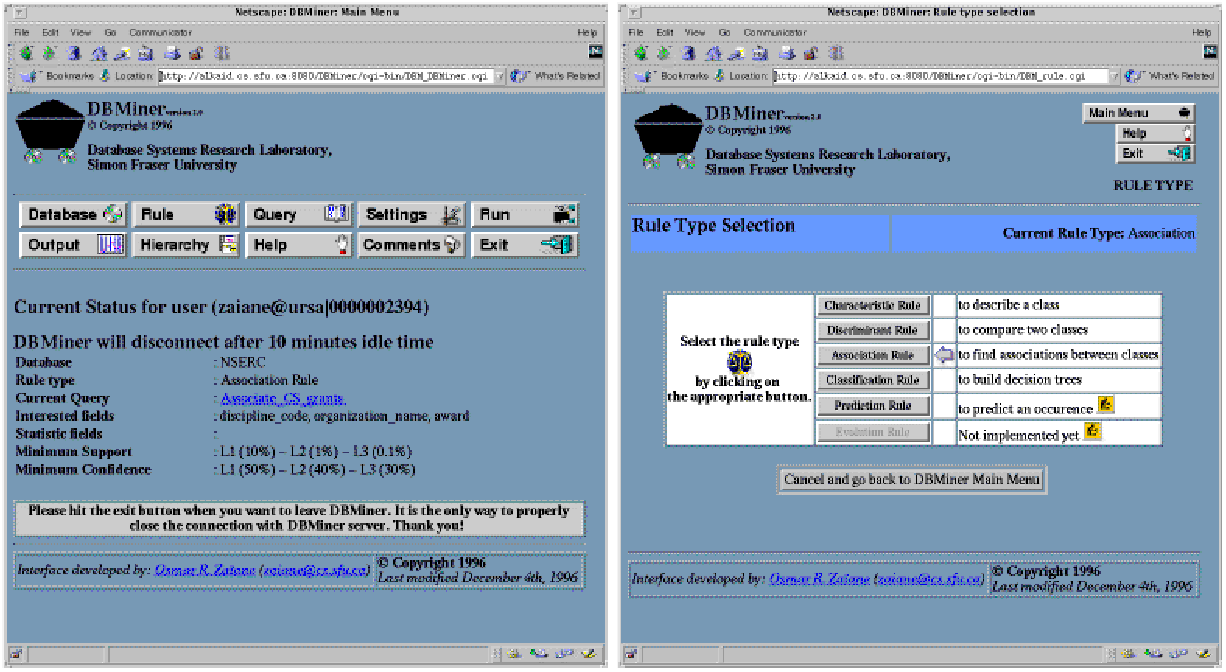
1. The set of data relevant to a data mining task should be specified in a data mining request;

2. The kind of knowledge to be discovered should be specified in a data mining request;

3. Background knowledge (such as conceptual hierarchy information, etc.) could be generally available for data mining process;

4. Data mining results should be able to be expressed in terms of generalized or multiple-level concepts;

5. Various kinds of thresholds should be able to be specified (interactively if possible) to filter out less interesting knowledge.


Netscape: DBMiner: Main Menu
File Edit View Go Communicator Help
Bookmarks Location: http://alkaid.cs.sfu.ca:8080/DBMiner/cgi-bin/DBM_DBMiner.cgi What's Related
DBMiner version 1.0
© Copyright 1996
Database Systems Research Laboratory,
Simon Fraser University
Database
Rule
Query
Settings
Run
Output
Hierarchy
Help
Comments
Exit

Current Status for user (zaiane@ursa|0000002394)

DBMiner will disconnect after 10 minutes idle time

Database	: NSERC
Rule type	: Association Rule
Current Query	: Associate_CS_grants
Interested fields	: discipline_code, organization_name, award
Statistic fields	:
Minimum Support	: L1 (10%) – L2 (1%) – L3 (0.1%)
Minimum Confidence	: L1 (50%) – L2 (40%) – L3 (30%)

Please hit the exit button when you want to leave DBMiner. It is the only way to properly close the connection with DBMiner server. Thank you!

Interface developed by: Osmar R. Zaiane (zaiane@cs.sfu.ca) © Copyright 1996 Last modified December 4th, 1996

Netscape: DBMiner: Rule type selection
File Edit View Go Communicator Help
Bookmarks Location: http://alkaid.cs.sfu.ca:8080/DBMiner/cgi-bin/DBM_rule.cgi What's Related
DBMiner version 1.0
© Copyright 1996
Database Systems Research Laboratory,
Simon Fraser University
Main Menu
Help
Exit
RULE TYPE

Rule Type Selection

Current Rule Type: Association

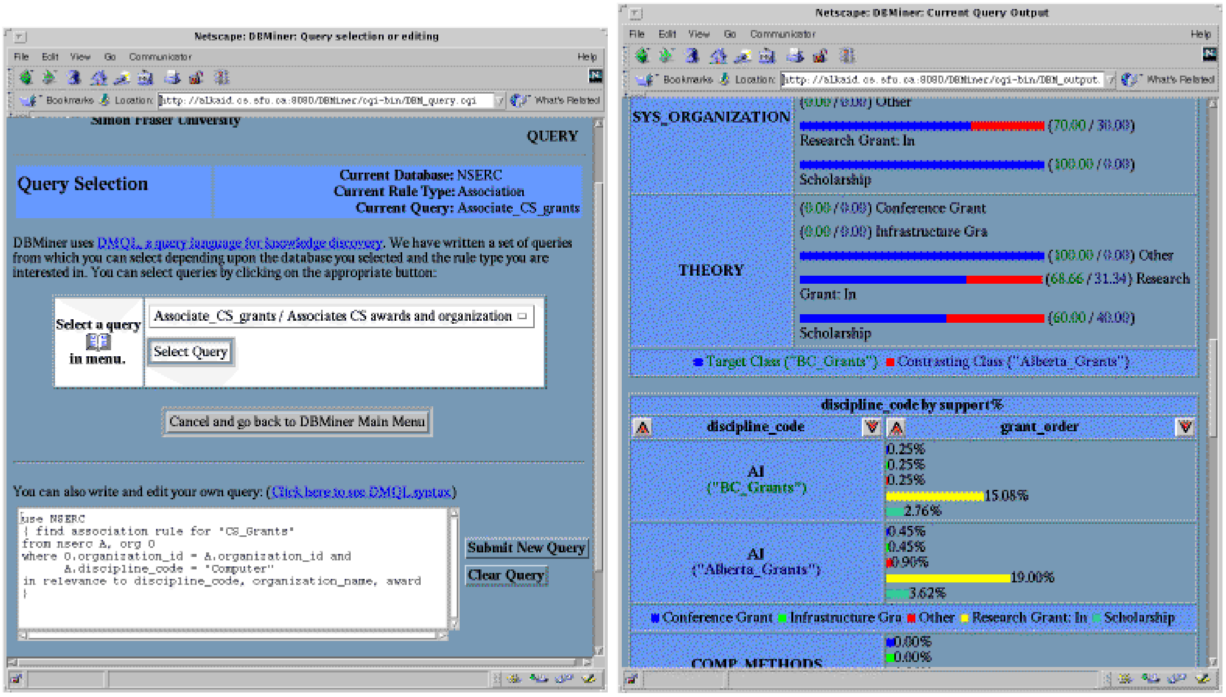
Select the rule type by clicking on the appropriate button.

Characteristic Rule		to describe a class
Discriminant Rule		to compare two classes
Association Rule	⇦	to find associations between classes
Classification Rule		to build decision trees
Prediction Rule		to predict an occurence
Evolution Rule		Not implemented yet

Cancel and go back to DBMiner Main Menu

Interface developed by: Osmar R. Zaiane (zaiane@cs.sfu.ca) © Copyright 1996 Last modified December 4th, 1996

Figure 3.2: Snapshots from DBMiner Web Interface (left: main menu - right: rule selection).


Netscape: DBMiner: Query selection or editing
File Edit View Go Communicator Help
Bookmarks Location: http://alkaid.cs.sfu.ca:8080/DBMiner/cgi-bin/DBM_query.cgi What's Related
Simon Fraser University
QUERY

Query Selection

Current Database: NSERC
Current Rule Type: Association
Current Query: Associate_CS_grants

DBMiner uses DMQL, a query language for knowledge discovery. We have written a set of queries from which you can select depending upon the database you selected and the rule type you are interested in. You can select queries by clicking on the appropriate button:

Select a query in menu.

Associate_CS_grants / Associates CS awards and organization

Select Query

Cancel and go back to DBMiner Main Menu

You can also write and edit your own query: (Click here to see DMQL syntax)

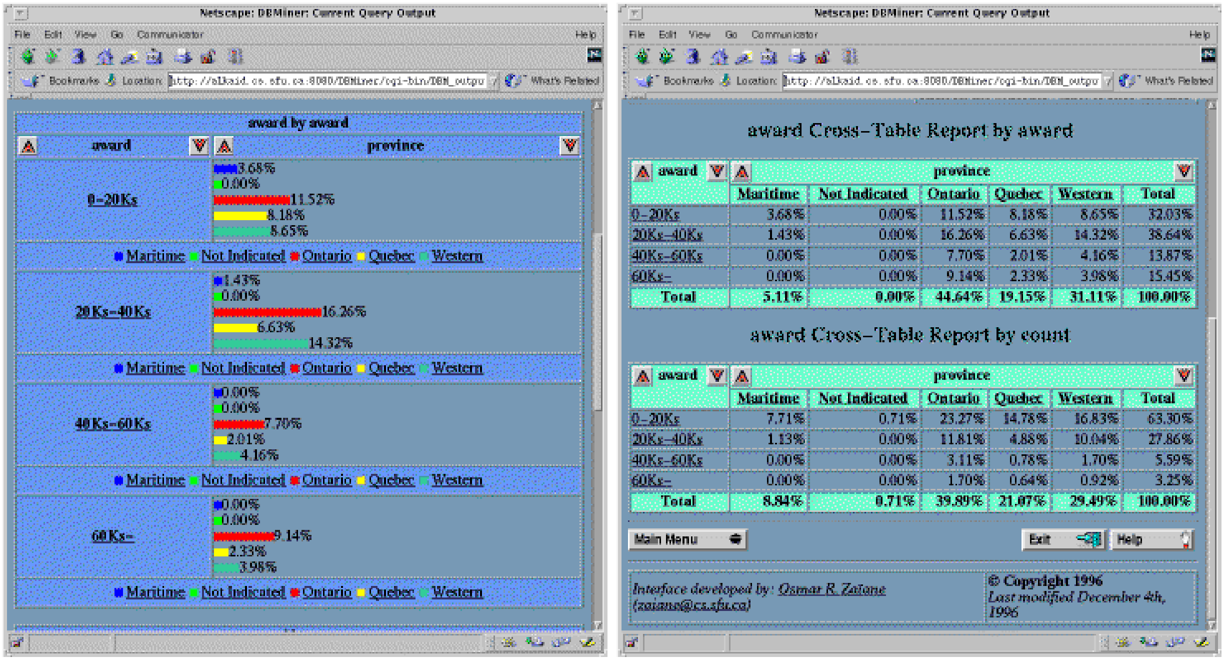
```
use NSERC
{ find association rule for "CS_Grants"
from nserc A, org O
where O.organization_id = A.organization_id and
      A.discipline_code = "Computer"
in relevance to discipline_code, organization_name, award
}
```

Submit New Query

Clear Query

Netscape: DBMiner: Current Query Output
File Edit View Go Communicator Help
Bookmarks Location: http://alkaid.cs.sfu.ca:8080/DBMiner/cgi-bin/DBM_output. What's Related
SYS_ORGANIZATION
(0.00 / 0.00) Other
(70.00 / 30.00) Research Grant: In
(100.00 / 0.00) Scholarship
THEORY
(0.00 / 0.00) Conference Grant
(0.00 / 0.00) Infrastructure Gra
(100.00 / 0.00) Other
(68.66 / 31.34) Research Grant: In
(60.00 / 40.00) Scholarship
■ Target Class ("BC_Grants") ■ Contrasting Class ("Alberta_Grants")
discipline_code by support%
discipline_code
grant_order
AI ("BC_Grants")
0.25%
0.25%
0.25%
15.08%
2.76%
AI ("Alberta_Grants")
0.45%
0.45%
0.90%
19.00%
3.62%
■ Conference Grant ■ Infrastructure Gra ■ Other ■ Research Grant: In ■ Scholarship
COMP_METHODS
0.00%
0.00%

Figure 3.3: DBMiner Web Interface (left: query selection - right: comparator).


Netscape: DBMiner: Current Query Output
File Edit View Go Communicator Help
Bookmarks Location: http://alkaid.cs.sfu.ca:8080/DBMiner/cgi-bin/DBM_outpu
What's Related
award by award
award
province
0–20Ks
3.68%
0.00%
11.52%
8.18%
8.65%
■ Maritime ■ Not Indicated ■ Ontario ■ Quebec ■ Western
20Ks–40Ks
1.43%
0.00%
16.26%
6.63%
14.32%
■ Maritime ■ Not Indicated ■ Ontario ■ Quebec ■ Western
40Ks–60Ks
0.00%
0.00%
7.70%
2.01%
4.16%
■ Maritime ■ Not Indicated ■ Ontario ■ Quebec ■ Western
60Ks–
0.00%
0.00%
9.14%
2.33%
3.98%
■ Maritime ■ Not Indicated ■ Ontario ■ Quebec ■ Western
Main Menu
Exit
Help

award Cross–Table Report by award

award	province					
	Maritime	Not Indicated	Ontario	Quebec	Western	Total
0–20Ks	3.68%	0.00%	11.52%	8.18%	8.65%	32.03%
20Ks–40Ks	1.43%	0.00%	16.26%	6.63%	14.32%	38.64%
40Ks–60Ks	0.00%	0.00%	7.70%	2.01%	4.16%	13.87%
60Ks–	0.00%	0.00%	9.14%	2.33%	3.98%	15.45%
Total	5.11%	0.00%	44.64%	19.15%	31.11%	100.00%

award Cross–Table Report by count

award	province					
	Maritime	Not Indicated	Ontario	Quebec	Western	Total
0–20Ks	7.71%	0.71%	23.27%	14.78%	16.83%	63.30%
20Ks–40Ks	1.13%	0.00%	11.81%	4.88%	10.04%	27.86%
40Ks–60Ks	0.00%	0.00%	3.11%	0.78%	1.70%	5.59%
60Ks–	0.00%	0.00%	1.70%	0.64%	0.92%	3.25%
Total	8.84%	0.71%	39.89%	21.07%	29.49%	100.00%

Interface developed by: Osmar R. Zaïane (zaiane@cs.sfu.ca)

© Copyright 1996
Last modified December 4th, 1996

Figure 3.4: DBMiner Web Interface (barcharts and cross tables for summarization).

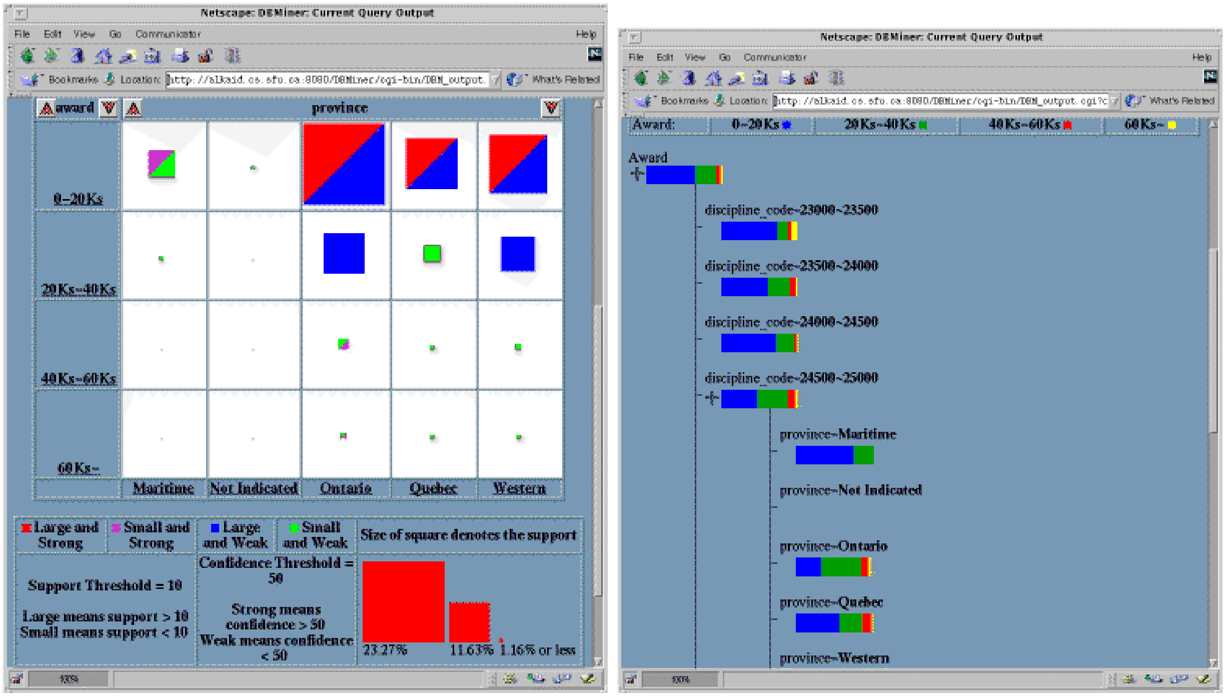

Netscape: DBMiner: Current Query Output
File Edit View Go Communicator Help
Bookmarks Location: http://alkaid.cs.sfu.ca:8080/DBMiner/cgi-bin/DBM_output.
What's Related
award
province
0–20Ks
20Ks–40Ks
40Ks–60Ks
60Ks–
Maritime
Not Indicated
Ontario
Quebec
Western
■ Large and Strong
■ Small and Strong
■ Large and Weak
■ Small and Weak
Size of square denotes the support
Support Threshold = 10
Large means support > 10
Small means support < 10
Confidence Threshold = 50
Strong means confidence > 50
Weak means confidence < 50
23.27%
11.63%
1.16% or less
100%
Netscape: DBMiner: Current Query Output
File Edit View Go Communicator Help
Bookmarks Location: http://alkaid.cs.sfu.ca:8080/DBMiner/cgi-bin/DBM_output.cgi?c
What's Related
Award:
0–20Ks ■
20Ks–40Ks ■
40Ks–60Ks ■
60Ks– ■
Award
discipline_code–23000–23500
discipline_code–23500–24000
discipline_code–24000–24500
discipline_code–24500–25000
province–Maritime
province–Not Indicated
province–Ontario
province–Quebec
province–Western
100%

Figure 3.5: DBMiner Web Interface (left: association rules - right: classification tree).

DMQL consists of the specifications of four major primitives in data mining: (1) the set of data in relevance to a data mining process, (2) the kind of knowledge to be discovered, (3) the background knowledge, and (4) the justification of how the discovered knowledge could be interesting (i.e., thresholds).

The first primitive, the set of relevant data, is specified in a way similar to that of a relational query, which is to be used to retrieve the set of relevant data from the database.

The second primitive, the kind of knowledge to be discovered, includes generalized relations, characteristic rules, discriminant rules, classification rules, association rules, etc.

The third primitive, the background knowledge, is a set of concept hierarchies or generalization operators which provide corresponding higher level concepts and assist generalization processes.

The fourth primitive, the significance of the knowledge to be discovered, can be specified as a set of different mining thresholds depending on the kinds of rules to be mined.

DMQL adopts an SQL-like syntax to facilitate high level data mining and natural integration with relational query language, SQL. The DMQL language is concisely defined in an extended Backus-Naur Form (BNF) grammar [140], where "[ ]" represents 0 or one occurrence, "{ }" represents 0 or more occurrences, and uppercase words represent keywords, as shown below:

```
<DMQL> ::=
   USE DATABASE <database_name>
   {USE HIERARCHY <hierarchy_name> FOR <attribute>}
   <rule_spec>
   RELATED TO <attr_or_agg_list>
   FROM <relation(s)>
   [WHERE <condition>]
   [ORDER BY <order_list>]
   {WITH [<kinds_of>]  THRESHOLD = <threshold_value> [FOR <attribute(s)>]}
```

Appendix C presents a more elaborated syntax of the DMQL language. Moreover, DMQL allows the specification and manipulation of concept hierarchies. We will present these capabilities later.

In <DMQL>, "USE DATABASE <database_name>" directs the mining task to a specific database "<database_name>", and the optional statement, "USE HIERARCHY <hierarchy> FOR <attribute>", assigns <hierarchy> to a particular attribute <attribute> (otherwise, a default hierarchy is used). The statement, <rule_spec>, is the specification of the kind of rules to be discovered. The following kinds of rules are considered in DMQL:

1. Mining characteristic rules.

   <rule_spec> ::= SUMMARIZE <attributes> WITH RESPECT TO <attributes> [AS <rule_name>]

2. Mining discriminant rules.

   <rule_spec> ::= COMPARE <class_1> WHERE <condition_1>
   IN CONTRAST TO <class_2> WHERE <condition_2>}
   {IN CONTRAST TO <class_i > WHERE <condition_i >}
   [AS <rule_name>]

3. Mining classification rules.

   <rule_spec> ::= MINE CLASSIFICATION [AS <rule_name>] WITH RESPECT TO <attributes>

4. Mining association rules.

   <rule_spec> ::= MINE ASSOCIATION [AS <rule_name>] WITH RESPECT TO <attributes>

5. Mining prediction rules.

   <rule_spec> ::= MINE PREDICTION [AS <rule_name>] WITH RESPECT TO <attributes>

The WITH RESPECT TO statement selects a list of relevant attributes and/or aggregations for generalization "<att_or_agg_list>". The FROM and WHERE clauses, form an SQL query to collect the set of relevant data. The ORDER BY clause simply specifies the order of rows to be printed. The "WITH-THRESHOLD" statement specifies various kinds of thresholds, noise, support, confidence, etc. depending upon the type of rules to be discovered.

While it is possible to use DMQL for interactive mining, the language was not designed for this purpose, even though a user interface allowing this interaction has been implemented for the Web and the PC versions, but without a robust parser. The major goal in the design of DMQL is to provide a user with primitives for data mining programming. In the same way SQL is embedded into programming languages, DMQL is also meant to be embedded in languages such as C/C^{++}. A function library translates DMQL into SQL before acting on the database. It is the same goal that we are aiming for *WebML*, the language for resource and knowledge discovery from the Web that we will present in the next section.

The following DMQL example query finds multi-level strong association rules in a database containing information about documents and their authors. Patterns related to documents sizes, access frequency and the institution of the authors are found for documents in computer science published after 1990.

```
USE DATABASE WebDocuments
MINE ASSOCIATION
WITH RESPECT TO D.size, D.access_frequency, A.institution
FROM document D, author A
WHERE D.date > 1990 AND
        D.Topic = “Computer Science” AND
        A.author = D.author
```

The portion of the query for finding the relevant set of data is first transformed into a standard SQL query which retrieves all the data items in the “computer science” topic and dated after 1990. WITH RESPECT TO becomes a SELECT in this case. The topic is taken at a high concept level; “computer science” covers “database”, “artificial intelligence”, “software engineering”, etc. The three attributes *size*, *access_frequency* and *institution* are generalized in the relations *document* and *author* along the concept hierarchies for the respective interested attributes. Then the multi-level association rule algorithm is applied on this data set, giving the possibility to the user to drill-down and roll-up interactively along any dimension.

The background knowledge that DMQL exploits is in the form of concept hierarchies (or lattices). As presented in the concise grammar earlier, USE HIERARCHY allows to specify hierarchies to use. DMQL also offers the possibility to create and manipulate hierarchies. DEFINE HIERARCHY is used to either generate hierarchies at the schema level based on the database attribute relationships, or generate hierarchies by grouping sets. For example, the following commands generate a sub-hierarchy defining the locations subsumed by *Western Canada* and by *Canada*:

```
DEFINE HIERARCHY FOR Location
{British Columbia, Alberta, Manitoba, Saskatchewan} < {Western-Canada}
```

```
DEFINE HIERARCHY FOR Location
{Western-Canada, Central-Canada, Maritimes} < {Canada}
```

DELETE and INSERT commands allow maintenance of hierarchies. For example, the following command adds the node "Territories" to the set subsumed by *Canada*:

```
INSERT {Territories} UNDER Canada
TO HIERARCHY FOR Location
```

The DMQL version currently implemented in DBMiner differs slightly from the syntax and keywords presented in this section. After the publication of [126], DMQL has been refined by other members of the DBMiner implementation team.

We present a more elaborate syntax in Backus-Naur Form (BNF) in Appendix C.

3.2 Multi-Layered Database Model - The Short Story

The goal for the construction of the MLDB is to transform and/or generalize the unstructured data of the primitive layer at each site into relatively structured data, manageable and retrievable by the database technology. Our motivation is not web page restructuring, as with WebLog and WebOQL, but rather web page content and web page inter-relations abstraction. However, answers to WebML queries can be used to generate new Web documents, and thus, can be considered Web restructuring.

Specialized tools, similar to Essence[129] are executed locally on information provider sites to extract pertinent data from documents. WebLog and WebOQL-like query languages, or networked information retrieval tools like Ahoy![226], or tools that take advantage of metadata if present, can also be exploited to gather the needed information from within documents. This information is stored in the first layer and is generalized in higher levels. The layer-1 is distributed and could reside locally on each information provider site. It is only the higher levels that are gathered in a centralized location and mirrored for better performance.

Extracting information from structured bibtex files or postscript papers is fairly smooth. However, most web pages don't easily convey the needed information. The extensible markup language (XML) developed by the World-Wide Web Consortium, and the Dublin

Core initiative, will help in this direction. Many web publishing tools are adopting XML and will help promote widespread improved structured web documents. The Dublin Core Metadata workshop has stressed the importance of metadata (i.e. document descriptors) in networked documents to facilitate resource discovery [259, 258] and the 15 metadata element set seems to gain popularity. Already, extensions to the HTML specifications include some tags allowing the description of keywords and content summary inside the HTML document (see Appendix D). Because of their use by search engines in their ranking of documents, more web document authors are now willing to manually add these descriptions in their web pages.

3.3 Web Mining Language

Similar to other extended-relational database systems, a Virtual Web View (VWV) system treats the requests for information browsing and resource discovery like relational queries. However, since concepts in a VWV are generalized at different layers, search conditions in a query may not match exactly the concept level of the currently inquired or available layer of the database. For example, to find documents related to a particular topic, such as "attribute-oriented induction", a query may put this term as a search key. However, the current layer may only contain terms corresponding to a higher concept level, such as "induction techniques", or "data mining methods". In this case, it is unlikely to find in the current layer an exact match with the provided search key, but is likely to find a more general concept that subsumes the search key. On the other hand, a search key in a query may be at a more general concept level than those at the current layer. For example, a search key "sports", though conceptually covers the term "baseball", does not match it in the database. Therefore, a key-oriented search in a VWV leads us to introduce four additional relational operations to extend the semantics of traditional selection and join. These operators, *coverage, subsumption, synonymy*, and *approximation*, have their correspondent built-in language primitive in WebML defined respectively as COVERS, COVERED BY, LIKE and CLOSE TO. Other primitives could be defined by users and written as external programs accessing the concept hierarchy (See Appendix B for syntax).

Definition 3.3.1 In the global MLDB system, four additional intrinsic relationships: *coverage, covered_by, synonym*, and *approximation*, are defined as follows.

1. coverage ($\supset$): A concept A covers another concept B, denoted as $A \supset B$, if A or A's synonym is an ancestor of B or B's synonym in the same concept hierarchy.

2. covered_by ($\subset$): A concept A is covered by another concept B, denoted as $A \subset B$, if A or A's synonym is a descendant of B or B's synonym in the same concept hierarchy.

3. synonym ($\simeq$): A concept A is a synonym of another concept B, denoted as $A \simeq B$, if A and B are in the same alias list in the same concept hierarchy.

4. approximation ($\sim$): A concept A is an approximate of another concept B, denoted as $A \sim B$, if A or A's synonym is a sibling of B or B's synonym in the same concept hierarchy. □

Based on these relationships, additional selection and join operations can be defined as follows.

Definition 3.3.2 Let σ be a selection performed on the i-th attribute (column) of relation R using the selection constant c. Four addition selection operations are defined as follows,

1. coverage-selection: if the selection predicate is $c \supset \$i$, i.e., the selection operation is $\sigma_{c \supset \$i} R$,

2. covered_by-selection: if the selection predicate is $c \subset \$i$, i.e., the selection operation is $\sigma_{c \subset \$i} R$,

3. synonym-selection: if the selection predicate is $c \simeq \$i$, i.e., the selection operation is $\sigma_{c \simeq \$i} R$, and

4. approximation-selection: if the selection predicate is $c \sim \$i$, i.e., the selection operation is $\sigma_{c \sim \$i} R$. □

Similarly, one can define four corresponding join operations in the global MLDB systems by replacing the query constant c in the selection predicate of the definition with the j-th column of a relation S.

3.3.1 A query language for information discovery in the global MLDB

With the construction of the global MLDB, a query language, WebML, can be defined for resource and knowledge discovery using a syntax similar to the relational language SQL

[79, 152]. Four newly introduced operators have their correspondent language primitives in WebML, as shown in Table 3.1.

WebML primitive	operation	Name of the operation
COVERS	$\supset$	coverage
COVERED BY	$\subset$	covered_by
LIKE	$\simeq$	synonym
CLOSE TO	$\sim$	approximation

Table 3.1: New WebML primitives for additional relational operations.

```
<WebML> ::= <MINE HEADER> FROM relation_list
        [ RELATED TO name_list ] [ IN location_list ]
        WHERE where_clause
        [ORDER BY attribute_name_list]
        [RANK BY] {inward | outward | access}
<MINE HEADER>::=
        {{ SELECT | LIST } { attributes_name_list | ∗ }
        | <DESCRIBE HEADER> | <CLASSIFY HEADER>}
<DESCRIBE HEADER>::= MINE DESCRIPTION
        IN RELEVANCE TO { attributes_name_list | ∗ }
<CLASSIFY HEADER>::= MINE CLASSIFICATION
        ACCORDING TO attributes_name_list
        IN RELEVANCE TO { attributes_name_list | ∗ }
```

Table 3.2: The top level syntax of WebML.

WebML borrows heavily from a data mining query language DMQL [126] which we define in Section 3.1.3. The top-level WebML query syntax is presented in Table 3.2. A more formal grammar of WebML in BNF is presented in Appendix B. At the position for the keyword select in SQL, an alternative keyword list can be used when the search is to browse the summaries at a high layer, mine description can be used when the search is to discover and describe the general characteristics of the data, mine classification is used to find classifications of web objects according to some attributes, whereas select remains to be a keyword indicating to find more detailed information. Two optional phrases, "related-to *name_list*" and "in *location_list*", are introduced in WebML for quickly locating the related subject fields and/or geographical regions (e.g., Canada, Europe, etc.). They are semantically equivalent to some phrases in the where-clause, such as "*keyword* covered-by *field_names*" and/or "*location* covered-by *geo_areas*", etc. But their inclusion not only makes the query more readable, but also helps the system locate the corresponding high layer relation if there exists one. The phrase "according-to *attributes_name_list* in-relevance-to attributes_name_list" is only used for classification with mine classification. It indicates the attributes upon which to classify web objects. The where-clause is similar to that in SQL except that new operators may be used.

While this query language is simple, users do not have to learn it and write queries. A Java-based or HTML-based user interface can easily be developed on top of WebML to avoid heavy instruction queries, and to provide a means for interaction based on field-filling and button-clicking. This is one of our future projects.

3.3.2 WebML Operational Semantics

WebML queries are applied and pertain only to a given VWV which uses an MLDB structure. According to Definition 2.2.1 in Chapter 2, a VWV has three major components: $\langle \mathcal{S}, H, D \rangle$. $\mathcal{S}$ contains the schema of the virtual view, $\mathcal{H}$ contains a set of concept hierarchies which are a set of partial orders of the form $(P, a, \leq)$ where a is an attribute defined in the schema, P is a set of values in the domain *dom(a)* and $\leq$ is a reflexive, transitive and anti-symmetric binary relation representing subsumption of values in P, and $\mathcal{D}$ is a set of relations containing descriptors and abstractions of artifacts on the Web. Relations in $\mathcal{D}$ are organized in levels where relations in each level abstract the relations in the lower levels. The relationships among the relations at different levels of $\mathcal{D}$ are outlined in a route map in $\mathcal{S}$. Besides the route map and the conventional schema definitions of the relations in $\mathcal{D}$, $\mathcal{S}$ contains a set of generalization paths each of which shows how a higher layered relation is generalized from one or a set of lower layered relations; formally, $r = \Pi_A(r_1 \bowtie r_2 \bowtie ... \bowtie r_n)$ where Π and $\bowtie$ are respectively the projection and join operators in the relational algebra, r is the relation at level L, $r_1 ... r_n$ are relations at level l such that $l < L$ and A is the attribute set of r. $A = \{a_1 ... a_k\}$ where a_i is an attribute from one of $r_1 ... r_n$ and the value of a_i is either its value at level l or an upper bound in its partial order $(P, a_i, \leq)$. A value x of attribute a_i is an upper bound of y in $P =$*dom*(a_i) if $y \leq x$.

WebML queries pertain to the relations in $\mathcal{D}$ and utilize the partial orders in $\mathcal{H}$ as well as the generalization paths in $\mathcal{S}$. Note that the result of a WebML query is always a relation.

The result relation can be at any level of the MLDB structure, and can be "intercepted" by a data mining process for further computation. Processing WebML queries is made straightforward by translating them into corresponding SQL queries and mapping values in query conditions into upper or lower bounds in the pertinent partial orders. Since WebML takes advantage of concept hierarchies using the four additional intrinsic relationships: *coverage, subsumption, synonymy*, and *approximation* presented in Definition 3.3.1, these primitives are converted into disjunctions of concepts as follows:

1. *COVERS* ($\supset x$): The coverage is replaced by a disjunction of ancestors of x, $Q \supset x$ such that $\forall\ q \in Q, x \leq q$ in partial order $(P, a, \leq)$ where a is the attribute of concept x. Q is the set of least upper bounds of x, noted $\dashv \{x\}$.

2. *COVERED BY* ($\subset x$): The subsumption is replaced by a disjunction of descendants of x, $Q \subset x$ such that $\forall\ q \in Q, q \leq x$ in partial order $(P, a, \leq)$ where a is the attribute of concept x. Q is the set of all lower bounds of x, noted $\models \{x\}$.

3. *LIKE* ($\simeq x$): The synonymy is replaced by a disjunction of synonym concepts Q from the alias list of concept x in the partial order $(P, a, \leq)$ where a is the attribute of concept x such that $\forall\ q \in Q, x \leq q \wedge q \leq x$. Q is the set of all synonyms of concept x, noted $\simeq \{x\}$.

4. *CLOSE TO* ($\sim x$): The approximation is replaced by a disjunction of sibling concepts of x, $Q \sim x$ such that $\forall\ q \in Q, q \leq u \wedge x \leq u$ and u is least upper bound of x and q in partial order $(P, a, \leq)$ where a is the attribute of concept x, $u = \sqcup\{x, q\}$. Q is the set of all direct siblings of x, noted $\sqcap \sqcup \{x\}$.

Processing WebML Queries

WebML queries are translated into SELECT-FROM-WHERE SQL queries with identical structure, except for additional conditions in the WHERE clause from the RELATED TO and IN WebML clauses. RELATED TO F, adds conditions on the subject field and can be substituted by "$\wedge$ $a$coveredby$F$" in the WHERE clause, where a is the attribute of F. IN L, adds conditions on the geographical location (i.e. Internet domain) and can also be substituted in the WHERE clause by "$\wedge$ $l$coveredby$L$" where l is the attribute location or web address. LIST, like SELECT, is translated into an SQL SELECT statement with the exception that LIST aims at the highest MLDB layer possible while SELECT directs its results to the lowest layer. MINE DESCRIPTION and MINE CLASSIFICATION queries are also translated into SELECT statements with the attribute list from the IN RELEVANCE TO clause. In both cases, the information is retrieved at the lowest layer (Layer-1) and either collected in a data cube for OLAP purposed in the case of MINE DESCRIPTION, or a decision tree is constructed for the retrieved data, based on class labels from the ACCORDING TO clause in the case of MINE CLASSIFICATION.

The major challenge of processing WebML queries in a VWV is to find the appropriate relations in the appropriate layer of the MLDB structure to execute the equivalent SQL queries. While the FROM clause indicates the lowest level relation containing descriptors of a given artifact on the Internet (i.e. document, person, image, game, etc.), the RELATED TO and IN clauses add conditions to help pinpoint the appropriate MLDB layer to execute the query. If the field or location specified in the RELATED TO or IN clause is known in the route map in $\mathcal{S}$, the relevant generalized relation is selected as source for the query. Moreover, the highest level in the partial orders of the different concepts used in the query is used to indicate the MLDB layer to use. For a query W, C is the set of all concepts and terms used in the query W and H the set of all partial orders of concepts in C, c is the highest concept used in hierarchy P if $c \in C \wedge \forall\ q \in C, q \leq c$ with $(P, a, \leq)$ where a is attribute of c. $\forall\ P \in H, P$ has only one highest concept level in C. The set of all highest concept levels in C and the route map in $\mathcal{S}$ identify the MLDB layer to use as source of the query W.

3.4 WebML Examples

As mentioned earlier, the MLDB structure provides ground for resource discovery on the Internet (i.e. pinpointing relevant documents) as well as knowledge discovery (i.e. implicit knowledge extraction). Following are examples of queries for resource discovery and for data mining from the Web which illustrate the semantics of WebML.

Example 3.4.1 (*Query for Resource Discovery*) The query, *list the documents published in Europe and related to "data mining"*, is presented as follows.

```
LIST        *
FROM        document    IN    Europe
RELATED TO              computing science
```

```
WHERE   ONE OF keywords COVERED BY "data mining"
```

Notice that the keyword LIST indicates that the query is to briefly browse the information, and therefore, it searches the relations using the where-clause as a constraint. Using SELECT instead of LIST would locate a set of URL addresses of the required documents, together with the important attributes of the documents. The keyword LIST, however, allows to display document attributes at a high conceptual level and provides and OLAP-like interaction. "FROM *document*" does not indicate to find the document relation at layer-0 or layer-1, but indicates to find the top-most layer of the *document* relation which fits the query. Therefore, "*document*" is a clue to the system to find the appropriate relation at a high layer. We adopt this convention since it is the system's responsibility to find the best match, and it is unreasonable to ask users to remember all the relation names at different layers. Moreover, the RELATED TO clause can help the system locate the appropriate top layer relation in case the relations are split by topic. To execute this query, the VWV system uses the phrase "FROM *document*" and "related-to *computing science*" to locate the top layer relation, *cs_document* for example. The phrase, "ONE OF *keywords* COVERED BY *'data mining'*" means that there exists an entry in the set *keywords* which is subsumed under '*data mining*'. Moreover, the phrase "IN *Europe*" confines the search to be within *Europe* which will be mapped into concrete countries using a concept hierarchy for Internet domains. In this case, a relatively large set of answers will be returned. An interactive process to deepen the search will usually be initiated by users after browsing the answer set. □

Example 3.4.2 (*Query for Resource Discovery*) To locate the documents related to data mining topics and linked from Osmar's webpage, and then rank them by *importance*, a simple WebML query is presented as follows.

```
SELECT  *
FROM    document
WHERE EXACT "http://www.cs.sfu.ca/˜zaiane" IN links_in
        AND ONE OF keywords COVERED BY "data mining"
        AND "Ted Thomas" IN authors
RANK BY INWARD, ACCESS
```

Notice that “SELECT ∗” means to print all the important attributes in a relation at a high layer, and moreover, “*‘Ted Thomas’* IN *authors*” means that *‘Ted Thomas’* is in the set of *authors*, whereas “ONE OF *keywords* COVERED BY *‘data mining’*” means that there exists an entry in the set *keywords* which is subsumed under ‘*data mining*’. “RELATED TO *computing science*” is not necessary in this particular query since ‘*data mining*’ is subsumed under ‘*computing science*’, however, this clause can alleviate ambiguity in case the search term is subsumed under more than one ancestor. Moreover, the RELATED TO clause can help the system locate the appropriate top layer relation in case the relations are split by topic. The EXACT keyword specifies that the URL should be used as given in the comparisons, and not as prefix of potential URLs as we shall see in the next example. To execute this query, the VWV system uses the phrase “FROM *document*” and “RELATED TO *computing science*” to locate the top layer relation *cs_document* for example, and then uses the two search keys in the where-clause as well as the links-in set to locate a set of URL addresses of the required documents, together with the brief descriptions: *authors, title, publication, publication date, keywords*, etc. The documents would be ranked by the number of hyperlinks linking to them from other resources and how often these documents are accessed. This translates the subjective term *importance* stated in the request. Users have the choice to either find more detailed layer-1 descriptions (i.e. drill-down), or directly access the documents by clicking at different buttons (drill-through). □

Example 3.4.3 (*Query for Resource Discovery*) To locate the documents about “Intelligent Agents” published at Simon Fraser University (SFU) and that link to Osmar’s web pages in at least two depth link paths, the following query could be written:

```
SELECT ∗
FROM document IN “www.sfu.ca”
RELATED TO “computer science”
WHERE “http://www.cs.sfu.ca/~zaiane” IN links_out (− > OR − > − >)
          AND ONE OF keywords LIKE “Intelligent Agents”
```

In this query the EXACT keyword was not used. This means that the URL given in the query could be used as a prefix to any address in the links_out set. Moreover, → and →→ are used to check recursively the links_out sets at a depth 2 for other links that verify the condition. Only local links are used since the URL and the SFU domain match; global links are not necessary. The ‘IN “www.sfu.ca” ’ phrase is used to limit the retrieval to the SFU domain, extracted from the Internet domain hierarchy. The LIKE keyword is used to match “Intelligent Agents” to other synonyms defined in the concept hierarchy.

This query returns a list of URL addresses together with important attributes of the documents that match. □

Example 3.4.4 (*Query for Knowledge Discovery*) To inquire about European universities *productive* in publishing on-line *popular* documents related to database systems since 1990, a WebML query is presented as follows:

```
SELECT affiliation
FROM document IN “Europe”
WHERE affiliation COVERED BY “university”
          AND ONE OF keywords COVERED BY “database systems”
          AND publication-year > 1990
          AND count = “high”
          AND h(links-in) = “high”
```

In this query, “productive” is measured as those that published a *high* number of papers. The term “high” is a generalization of the numeric value of access-frequency along its concept hierarchy. While constructing the layers of the MLDB structure, concept hierarchies for numerical attributes are automatically built and labeled later by users. The label “high” would, for example, correspond to a count greater than 20, in other words affiliation with more than 20 published papers. “Popular” is measured by the high number of hyperlinks coming from other resources towards these papers. *links-in* in this case is not just counted (cardinality of links-in set), but is surveyed by a heuristic function h, provided by the user, which calculates the popularity based on the importance of the links. For example, a local link would get the weight 0.5, a link from a resource related to the condition in the where-clause would get 2, while other global links get 1.

What is interesting to note is that the execution of this query does not return a list of document references, but rather a list of universities (publishing popular documents about databases), which is implicit information (or knowledge) extracted from a conglomerate of documents. □

Example 3.4.5 (*Query for Knowledge Discovery*) Suppose the query is to "*describe the general characteristics in relevance to authors' affiliations, publications, etc. for those documents which are popular on the Internet and are on "data mining"*. A knowledge discovery query to answer this request, characterized by the keyword "MINE DESCRIPTION" is shown below:

```
MINE DESCRIPTION
IN RELEVANCE TO authors.affiliation, publication, pub_date
FROM      document RELATED TO Computing Science
WHERE     ONE OF keywords LIKE "data mining"
          AND access_frequency = "high"
```

The discovery query will be first executed as a retrieval to collect from *cs_document* the data which are relevant to "*authors.affiliation, publication, pub_date*" and satisfy the where-clause. Then the attribute-oriented induction is performed on the collected data, which generalizes "*publication*" into groups, such as major AI journals, major database conferences, and so on, and generalizes publication date to year, etc. The generalized results are collected in a data cube and can be interactively manipulated by the user using OLAP operations. □

Example 3.4.6 (*Query for Knowledge Discovery*) To classify according to update time and popularity the documents published on-line in sites in the Canadian and commercial Internet domain after 1993 and about information retrieval from the Internet, a WebML query can be presented as follows:

```
MINE CLASSIFICATION
ACCORDING TO timestamp, access_frequency
IN RELEVANCE TO *
FROM document IN Canada, Commercial
WHERE     ONE OF keywords COVERED BY "information retrieval"
          AND ONE OF keywords LIKE "Internet"
          AND publication_year > 1993
```

The phrase MINE CLASSIFICATION requests a classification tree from the system. The query first collects the relevant set of data from the VWV relations, executes a data classification algorithm to classify documents according to their access frequency and their last modification date, then presents each class and its associated characteristics in a tree. The user can navigate the tree representation and drill through to the documents if needed. □

3.5 Preliminary Experiment

We have conducted an experiment as a proof of concept for the Virtual Web View using a subset of artifacts found on the Internet. The intent of this experiment was not to implement WebML, but to show the capabilities and flexibility of the language, given an MLDB structure.

In our experiment [276] we wanted to demonstrate the strength of our model for information discovery. We assumed that the layer-1 construction softwares exist and built layer-1 manually. Our experiment is based on Marc Vanheyningen's Unified Computer Science Technical Reports Index (UCSTRI)[250] and is confined to computer science documents only. In other words, our experimental VWV is specialized in computer science technical reports published on-line. It is the best represented subset of the Internet since computer scientists are those who put most papers and technical reports on-line today. UCSTRI master index was created by merging indexes of different FTP sites. These indexes, though not fully satisfactory to our usage, contain rich semantic information like keywords, abstracts, etc. We used the master index as primitive data to create our MLDB structure by selecting 1224 entries from four arbitrarily chosen FTP sites (University of California Berkeley, Indiana University, INRIA France and Simon Fraser University). Since an important number of documents did not have keywords attached to them, we manually deduced them or used the title and, if available, the first several sentences of the abstract to do so. Using a subset of the Internet simplifies the concept hierarchies to be built. The results can be easily extrapolated to the whole Internet to prove the feasibility of our model. The aim of using Vanheyningen's master index as primitive data for our experiment is to be able to compare the query results with what the conventional UCSTRI search engine available on the Internet can provide. The first layer of our VWV was built based on the information provided by the four FTP sites we chose. The layer-1 of our simplified MLDB structure contains just one relation:

document(*file_addr, authors, affiliation, title, publication, publication_date, abstract, keywords, URL_links, num_pages, form, size_doc, timestamp, local_ID, note*).

The 1224 tuples relation constitutes our mini database on top of which we constructed a concept hierarchy for keywords. Part of the concept hierarchy is illustrated in Fig. 2.9. Note that our concept hierarchy was built manually for this experiment, but we could have used an already built taxonomy. To build the hierarchy, we collected all unique keywords and grouped them by synonyms. All synonym representatives were classified in a tree hierarchy. This hierarchy was used to deduce general topics for generalization of layer-1 tuples. Once generalized to layer-2, our relation looks as follows:

doc_summary(*affiliation, field, publication_year, count, first_authors_list, file_addr_list*).

The field field contains a high level concept which embraces all lower concepts under it. The field count is a counter for the documents that correspond to affiliation, field and pub_year.

Table 3.3 shows a portion of the tuples in doc_summary.

affiliation	field	pub_year	count	first_author_list	file_addr_list	···
Simon Fraser Univ.	Natural Language	1993	6	Popowich, Dahl, ···	···	···
Simon Fraser Univ.	Parallel Programming	1993	5	Liestman, Shermer, ···	···	···
Indiana University	Machine Learning	1994	5	Leake, Fox, ···	···	···
···	···	···	···	···	···	···

Table 3.3: A portion of *doc_summary.*

Notice that backward pointers can be stored in certain entries, such as *first_author_list* and *file_addr_list*, in the *doc_summary* table, and a click on a first author or a file_address will lead to the presentation of the detailed corresponding entries stored in layer-1.

WebMiner, the experimental prototype, allows a progressive search leading to a suboptimal hit ratio. The same simple query submitted to the search engine UCSTRI and to our VWV returns two different answers revealing a better hit ratio with our model. A query like:

```
select      *
from        document
related-to  Parallel Computing
where       one of keywords closeto "Gossiping"
```

gives, using UCSTRI, 50 references in the 4 targeted FTP sites. Only 13 of which are indeed related to parallel computing. The same query submitted to our model, return 21 references all related to parallel computing but with reference to gossiping or broadcasting (i.e., siblings in the concept hierarchy). WebMiner not only reduces the noise by giving just documents related to the appropriate field, but also improves the hit ratio by checking synonyms and siblings in the concept hierarchies.

A WebML parser was not implemented for this experiment. The queries were translated (manually) to SQL and *C* program routines. For UCSTRI, we simply entered the search keys in the appropriate fields of the search engine. Since both UCSTRI and WebMiner were essentially using the same index (the VWV is built on top of the index), it was possible to compare the results.

While the master index of UCSTRI is used solely for resource discovery, WebMiner allows queries like:

```
describe   affiliation, publication_date.year
from       document
where      one of keywords like "Computational Geometry"
```

This query returns the brief description of all universities or organizations that published documents about Computational Geometry with the date of publication as shown in Table 3.4. This query clearly does not target the documents themselves but the information about them. Note that this information is not explicitly published anywhere on the Internet but the generalization in layers of the VWV makes it fully revealed. The question mark in the last entry is due to the fact that the publication date is not indicated on the documents served at INRIA's FTP site.

affiliation	pub_year	count	count %
Simon Fraser University	1990	1	8.3%
Simon Fraser University	1991	2	16.6%
Univ. of California Berkeley	1988	1	8.3%
Univ. of California Berkeley	1990	3	25.0%
Univ. of California Berkeley	1991	1	8.3%
INRIA France	?	4	33.33%

Table 3.4: Affiliations that published about Computational geometry.

For a query like:

```
describe   affiliation
from       doc_summary
where      affiliation belong_to "university" and field = "Machine Learning"
           and publication_year > 1990 and count > 2
```

a simple search in the table doc_summary produces the list of the universities which serve at least 2 documents about machine learning published after 1990 shown in Table 3.5. Such a query is not processible with the conventional search engines on the World-Wide Web.

affiliation	count	count %
Indiana University	13	68.4%
Univ. of California Berkeley	6	31.6%

Table 3.5: Affiliations that published more than 2 documents about Machine Learning after 1990.

It is clear that the generalization of the VWV allows WebMiner to mine the Internet by simply querying the metadata summarized in the different layers without accessing the artifacts themselves, once the VWV is constructed.

While our preliminary experiment was confined to computer science documents only by using Marc Vanheyningen's Unified computer science master index [250] as primitive data to create our VWV, it illustrates the feasibility of our model. Using a subset of the Internet also simplifies the concept hierarchies to be built. For a first step, before the automated tools are constructed, we believe that this subset is rich enough to give us interesting results. The layer-1 was built into a unique Sybase relation and the upper layers were constructed using a semi-automatically generated concept hierarchies.

3.6 Conclusion and Future Work

Search engines currently available on the Internet are keyword-driven, and the answers presented are lists of presumably relevant documents. The VWV and WebML allow us to apprehend and solve the resource discovery issues by presenting lists of relevant documents to users, but also allowing the users to progressively and interactively browse detailed information leading to a targeted set of pertinent documents. The resource discovery led by progressively detailed information browsing suits the users who do not have a clear mind on what are the exact resources they need. WebML queries are treated like information probes, being mapped to a relatively high concept layer and answered in a hierarchical manner. Moreover, the knowledge discovery power of WebML is unique. It helps find interesting high level information about the global information base. It provides users with a high-level view of the database, statistical information relevant to the answer set, and other associative and summary information at different layers. In addition, the VWV model can take advantage of other web page restructuring query languages, such as WebLog and WebOQL, and available networked agents, such as Ahoy!, to retrieve pertinent descriptors from web documents and build the first layer of the MLDB structure.

We see WebML as a programming language for Web mining, to be embedded in other traditional programming languages, more than an interactive query language, much like SQL is today. An interpreter for WebML would translate queries to SQL to take advantage of the powerful and sophisticated SQL optimizers available.

Experiments run locally on a collection of on-line documents were very promising. We plan to extend these experiments and include full operational web sites. The design and implementation of a point-and-click user interface is also projected. The interface would alleviate the need for writing queries directly in WebML, and it will also allow interactive OLAP on a “hyperspace” data cube.

We have studied data mining from web access logs[282, 141] and we plan to design a query language for web log analysis and data mining. Like DMQL [126], the language will be specialized for data mining and will integrate web content mining, web structure mining and web usage mining by merging the capabilities of both WebML and DMQL.

Part II

Multimedia Mining

The future belongs to those who believe in the beauty of their dreams.

Eleanor Roosevelt

Nothing is more difficult, and therefore more precious, than to be able to decide.

Napoleon Bonaparte

Chapter 4

Content-Based Visual Media Retrieval

The Virtual Web View (VWV) is a stratified structure that attempts to make part of the Internet artifacts appear structured enough for resource and knowledge discovery by abstracting their features along different layers (see Chapter 2). The first layer generated (i.e. layer-1) is organized into dozens of database relations, such as *document, person, organization, images, sounds, software, map, library_catalogue, commercial_data, geographic_data, scientific_data, games*, etc., where each of these relations contains descriptors of artifacts from a particular class. These artifacts from the Internet can either be physical, like text documents and images, or virtual like people and networks. One of the relations that is of a particular interest in this chapter is the relation *image*, which contains descriptors of still images and videos found in the World-Wide Web. Like in the Example 2.2.1 given in Chapter 2, here is an example of a database schema for the relation *image*:

Example 4.0.1 Let the database schema of layer-1 contain the relation, image as follows (with the attribute type specification omitted).

1. image(*image_addr, author, title, publication, publication_date, category_description, keywords, size, width, height, duration, format, parent_pages, colour_histogram, texture_histogram, colour_layout, texture_layout, chromaticity_vector, movement_vector, locale_vector, timestamp, access_frequency, ...*).

Each tuple in the relation is an abstraction of one *image* or *video* from the information base (layer-0). The whole relation is a detailed abstraction (or descriptor) of the information from images gathered from a site. The first attribute, *file_addr*, registers its file name and its "URL" network address. There are several attributes which register the information directly associated with the file, such as *size* (size of the image file), *timestamp* (the last updating time), etc. There are also attributes related to the formatting information. For example, the attribute *format* indicates the format of a file: .gif, .jpeg, .bmp, .avi, .mov, .png, etc., *height* and *width* indicate the height and width of the image or the video frame, while *duration* indicates the duration of the video in time. One special attribute, *access_frequency*, registers how frequently the entry is being accessed. Similar to *URL_links_in* in the relation *documents*, which indicates the popularity of a document, *parent_pages* for the relation *image* contains the list of web pages that contain the image or video in question. The number of URLs in this list would indicate the popularity of the image on the Web. Other attributes register the major semantic information related to the image or video, such as *authors, title, publication, publication_date, category_description, keywords*, etc., while attributes like *colour_histogram, texture_histogram, colour_layout, texture_layout, chromaticity_vector, movement_vector, locale_vector* indicate features related to the content of the image. □

While the relation *image* in layer-1 is treated and generalized in the same manner as other relations from layer-1 to higher layers in the MLDB as described in Chapter 2, in addition to the resource and knowledge discovery that the VWV allows with the images, the relation *image* can convey interesting retrieval properties thanks to the attributes that hold features related to the content of the images. In this Chapter we perceive how image descriptors are automatically extracted and study the potential offered by the attributes such as *colour_histogram, colour_layout, texture_layout, movement_vector* and others, to perform content-based retrieval from multimedia repositories. Content-based image retrieval is a retrieval founded on image content features like colours and textures, rather than just identifiers or "external" descriptors like keywords . We will show that it is possible to use image content features like colours and textures to solve object similarity searches in image and video collections, find images relevant to content-based queries, and even perform resource discovery on the basis of images contained in the resources (i.e. web documents). Content-based search and similarity search are often considered a sort of data mining, or at least processes and advances leading to data mining from multimedia databases. Such data mining is explored in the next chapters.

4.1 Related Work in Multimedia Resource Discovery

Image and video indexing and retrieval has always been an interesting research field that drew the attention of many researchers[199, 99, 21, 102, 12, 231]. The advent of the World-Wide Web brought a new challenge to the computer vision and artificial intelligence community. Research in computer vision, artificial intelligence, databases, etc. is taken to a larger scale to address the problem of information retrieval from large repositories of images. Traditional pattern recognition and image analysis algorithms in the vision and Artificial Intelligence fields dealt with small sets of still images and did not scale well. The large collections of images and video frames on the Internet require scalability as well as speed and efficiency. Some interesting image and video retrieval systems are beginning to appear on the World-Wide Web scene[231, 187, 102, 21, 169].

There are two main families of image and video indexing and retrieval systems: those based on the content of the images (content-based) like colour, texture, shape, objects, etc., and those based on the description of the images (description-based) like keywords, size, caption, etc. Description-based image retrieval systems suffer poor precision usually due to the term extraction process. Automatically assigning keywords to images is a tricky task. Content-based image retrieval systems [99, 21, 102, 231, 187] use visual features to index images. These systems differ mainly in the way they extract the visual features and index images, and the way they are queried. Some systems are queried by providing an image sample. These systems search for similar images in the database by comparing the feature vector (or signature) extracted from the sample with the available feature vectors. The image retrieval system Image Surfer[1] provided by Yahoo, for example, is based on this type of search. Other systems are queried by specifying or sketching image features like colour, shape or texture, which are translated into a feature vector to be matched with the known feature vectors in the database. QBIC [99] (Query By Image Content)[2] and WebSeek [231], for example, provide both sample-based queries and the image feature specification queries. WebSeer [101] on the other hand, combines image descriptions, like keywords, and image content, like specifying the number of faces in an image, and uses image content to distinguish between photographs and figures. However, the visual features extracted are very limited. C-BIRD [265, 169, 167] (Content-Based Image Retrieval from Digital Libraries), the

[1] Image Surfer available at http://ipix.yahoo.com/

[2] QBIC available at: http://wwwqbic.almaden.ibm.com

system that we developed, also exploits both content-based and description-based retrieval techniques, and combines conjunctions and disjunctions of image features and descriptions in the queries. It is also the only known image retrieval system that retrieves images from a large image repository based on objects (or models) they contain.

Another major difference between image retrieval systems is in the domain they index. While QBIC and Virage[21] solely index image databases (i.e. images from an internal database presented via a World-Wide Web front-end), using COIR (Content-Oriented Image Retrieval) [132], C-BIRD and AMORE (Advanced Multimedia Oriented Retrieval Engine) [187] index images from the World-Wide Web. Indexes of images on the Internet can also be used to pinpoint web pages containing particular images. C-BIRD for instance, using a graph of web pages and images they contain, can easily discover resources (web pages) given an image descriptor.

Image content-based systems give a relatively satisfactory result with regard to the visual clues, however, their precision and recall are still not optimized. It is important to note that image retrieval is usually based on similarity search rather than "exact" search.

Effective strategies for image retrieval will benefit from exploiting both content-based and description-based retrieval techniques, and will combine conjunctions and disjunctions of image features and descriptions in the queries, as well as providing users with adequate and efficient user interfaces for both querying and browsing.

We have been developing the C-BIRD (Content-Based Image Retrieval from Digital-libraries) system which combines automatically generated keywords and visual descriptors like colour, texture, shape, and feature localization, to index images and videos in the World-Wide Web.

Swain and Ballard's work on colour object recognition by means of a fast matching of colour histograms [241] began an interest in the use of simple colour–based features for image and video database retrieval. In this method, a database of coarse histograms indexed by three colour values is built up. A very simple and fast histogram matching strategy can often identify the correct match for a new image, or a near–match, by using an L1 metric[3] of histogram differences [241]. It was soon realized that, along with confounding factors such as object pose, noise, occlusion, shadows, clutter, specularities, and pixel saturation, a major problem arose because of the effect of changing illumination on images of colour

[3] L1 normalization normalizes X_i into $\frac{\mathrm{X_i}}{\sum_{\mathrm{k=1}}^{\mathrm{n}} \mathrm{X_k}}$ while L2 normalization normalizes X_i into $\frac{\mathrm{X_i}}{\sqrt{\sum_{\mathrm{k=1}}^{\mathrm{n}} \mathrm{X_k^2}}}$

objects [110].

Several colour object recognition schemes exist that purport to take illumination change into account in an invariant fashion.

In [72], the problem of illumination change is addressed by extending the original Swain and Ballard method to include illumination invariance in a natural and simpler way than heretofore. First, it is argued that a normalization on each colour channel of the images is really all that is required to deal properly with illumination invariance. Second, with an aim of reducing the dimensionality of the feature space involved, a full–colour (3D) representation is replaced by 2D chromaticity histograms. It is shown that the essential illumination–invariant colour information is maintained across this data reduction. The normalization step has the effect of undoing a changing–illumination induced shift in pixel colour–space position and in chromaticity space.

Histograms in chromaticity space are indexed by two values, and are treated as though they were images. In order to greatly improve the efficiency in searching large image and video databases, the chromaticity histogram-images are compressed and then indexed into a database with a small feature vector based on the compressed histogram. In the current implementation, the chromaticity histogram-image is first reduced by using a wavelet scaling function. Then, the Discrete Cosine Transform (DCT) is applied, followed by a zonal coding [246] of the DCT image. This results in an effective low–pass filtering of the chromaticity histogram. The resulting indexing scheme is very efficient in that it uses a DCT-chromaticity vector of only 36 values. We have also applied this technique to video segmentation [256]. Since it is much more frequent that illumination changes due to camera motions and object motions in video, the colour-channel-normalization method is shown to clearly outperform other cut-detection methods[133, 256, 12] (in video segmentation) that only rely on colour histograms.

Most existing techniques for content-based image retrieval rely on global image features such as colour and texture. These global methods are based on simple statistics extracted from the entire images. They are easy to obtain and to store in a database, and their matching takes little time. Inevitably, the global methods lack the power of locating specific objects and identifying their details (size, position, orientation, etc.). Some extensions to the global method include search by colour layout [99], by sketch [12, 187], and by colour regions according to their spatial arrangements [231].

In the remainder of this chapter, we describe the content-based image retrieval features in C-BIRD, and give an account of our implementation effort to retrieve images from the World-Wide Web[4]. Some retrieval results are also given. At the end, we summarize our conclusions and future possible enhancements.

4.2 A Content-Based Visual Media Retrieval System

Conceptually, C-BIRD is constructed on top of the MLDB structure (see Chapter 2) and uses one of the artifacts descriptor relations in the first bed of the structure. In reality, it has been designed independently from the Virtual Web Views described in Chapter 2. Note that C-BIRD does not exclude the VWV, and vice-versa. The multimedia repository considered is indeed the World-Wide Web. Images and videos are retrieved from the Internet, then processed to extract their visual content and descriptive features which are stored in a set of relations. This set of relations represent the relation *image* in layer-1 as previously mentioned. Given this data extracted from the images and video frames, C-BIRD is capable of different types of searches for resource discovery in visual media repositories such as the World-Wide Web. Search by conjunctions and disjunctions of keywords, as well as a combination of all the following searches are also possible.

- Similarity Search: a sample image is given and the system looks for similar images based on the colours present in the sample image. There are two possible similarity searches:

    1. search by colour histogram: similarity with colour histogram in a sample image;
    2. search by illumination invariance: similarity with colour chromaticity in a normalized sample image.

- Template-Based Search: some visual clues are given about the desired images to be retrieved, such as colour and texture patterns. The system matches these visual patterns with the visual descriptors collected from the multimedia repository. There are four types of these searches - two for colours (presence and layout), and two for texture density and orientation (presence and layout):

[4] The development of C-BIRD system is a team effort led by Dr. Ze-Nian Li. In particular, Bing Yan, Zinovi Tauber, Dr. Mark Drew and Dr. Jie Wei have also made significant contributions with original concepts, algorithms and implementations.

1. search by colour percentage: specification of up to 5 colours and percentages;
2. search by colour layout: specification of the layout of colours in a 1 × 1. 2 × 2, 4 × 4 or 8 × 8 grid;
3. search by edge density and orientation;
4. search by edge layout: specification of edge density and orientation in a 1 × 1. 2 × 2, 4 × 4 or 8 × 8 grid.

- Model-Based Search: an object model is given as a guide and the system looks for images with the specified object regardless of its position, its orientation (in 2 dimensions), and its size in the image.

Videos are treated as images after key frames are extracted from the video streams. A cut-detection algorithm [256] is used to detect drastic changes in the scene and determines where the scene changes occurs. Independent video segments are then identified. From each video segment a set of video frames are selected to represent the segment. We chose to select only one frame (the first frame of the video segment) to represent the segment. In other words, each video stream ends up being represented by a set of frames, one for each distinct segment. Most cut-detection methods used are colour-based. The algorithm simply detects drastic changes in the colour histogram from one frame to the other. This could mean for example that the camera has moved, a new scene is shown, or an object has appeared in the picture, etc. This is a very easy and efficient approach, however it may detect cuts where they shouldn't be, for example when there is an illumination change (i.e. flash, cloud covering the sun, etc.). The technique we opted to use acts on chromaticity rather than colour histogram comparisons [256, 257]. The results shown in [256, 167] demonstrate the efficiency of this method.

4.2.1 Similarity Search for Images and Video Frames

Given an initial image, called sample image, looking for similar images is a common need in many applications. Similarity, however, is very relative. Two similar images could be images of the same scene, for example two pictures of actions on the beach, or two images containing a boat, even if one is sailing and the other is in the harbour. Similar images could also be defined as containing objects semantically related, for example a picture of a school and a picture of a pupil, or two pictures of different animals. These definitions of similarity are however complex to "comprehend" by current computer applications[5]. In the context of the C-BIRD system, we opted for a more simple definition of similarity. Two images are considered similar if their colours are similar; they are similar if their colour histograms (or chromaticity vectors) are close enough given a threshold.

For simplicity, we have used the RGB colour space where colours are represented by triplets (r, g, b), r being a value of red, g a value of green, and b an value of blue, all ranging from 0 to 255 giving more than 16 million colours (the use of HSV or LUV colour space is also possible). We have discretized the colour space to reduce the number of distinct colours given the fact that human beings using a system like C-BIRD can not easily distinguish between a large number of colours. We have chosen to quantize the colours to 512, then 256, and finally to 64 distinct colours (i.e. four values for each dimension r, g, and b).

Search by Colour Histogram

The colour histogram of an image is obtained by counting the number of times each discrete colour of the discretized colour space occurs in the image array. Similar colours in the continuous RGB space which were quantized together in the discretized space count for the same discrete colour. In our case, we have 64 colours in the histogram (512 and 256 in our previous implementations of C-BIRD). Each bin in the histogram counts the number of pixels in a particular discrete colour. Since the sum of the numbers of all bins in the histogram is equal to the number of pixels in the picture, histograms can only be compared when images have the same sizes. In order to be able to compare images of different sizes, we normalized the histograms by converting the number of pixels in bins to percentages. In other words, the bins of a normalized histogram indicate the percentage of the pixels of an image in a particular discrete colour, and the histogram becomes a probability density making, effectively, each image have the same number of pixels.

Swain and Ballard developed in [241] a very useful histogram metric used for histogram comparisons. Given two histograms: H_M of the model image, and H_I of the image the model is compared to, the intersection μ between H_M and H_I is defined as in 2.1 with n being the number of discrete colours.

$$\mu \equiv \sum_{k=1}^{n} min\{H_M(k), H_I(k)\} \tag{2.1}$$

[5] There is no known functional image interpretation algorithm yet.

The intersection value μ is the number of pixels in the picture model that have corresponding pixels of the same colour in the image compared to the model. Swain and Ballard normalize intersection (or match) values by the number of pixels in the model histogram (H_M), and thus matches are between 0 and 1.

$$\Upsilon \equiv \frac{\sum_{\mathrm{k=1}}^{\mathrm{n}} \min\{\mathrm{H_M(k)}, \mathrm{H_I(k)}\}}{\sum_{\mathrm{k=1}}^{\mathrm{n}} \mathrm{H_M(k)}} \tag{2.2}$$

We have adopted Swain and Ballard's definition of histogram intersection, but since our histograms are already normalized (i.e. the volume under each histogram is equal to unity), the calculation of Υ in Equation 2.2 is not necessary. Indeed all images are normalized to the same size by using probabilities in the histograms. In C-BIRD we compare two images by calculating μ and contrasting it to a threshold θ. The match is successful if μ exceeds θ, and the image I is declared similar to the image model M. When looking for the best matches in an image repository, the system proceeds by intersecting the colour histograms. The highest value of μ, or in other words the smallest distance value $(1 - \mu)$ indicates the image that matches best.

The time for calculating histogram intersection is proportional to the number of distinct discrete colours n, and so is very fast.

Search by Illumination Invariance

Similar pictures taken over varying light conditions may not be considered similar to the colour histogram intersection method presented above, since with different illuminations, colours are not constant. Drew, Wei and Li introduced in [72] an indexing method that is efficient and invariant under illumination change by correcting the chromaticity of the illuminant in an image.

The chromaticity (r, g) for each pixel is defined by [264] as:

$$r = R/(R+G+B) \quad , \quad g = G/(R+G+B) \tag{2.3}$$

It is shown in [72] that normalizing a chromaticity histogram and reducing its size by a wavelet transformation, and then further reducing it by going to the frequency domain and discarding higher–frequency DCT coefficients, can yield a simple yet efficient colour indexing method that is invariant under illuminant change. We have used the colour-channel-normalization method proposed by Drew, Wei and Li in [72] to derive and implement a similarity search by Illumination Invariance.

Given an image of size $m \times n$, each of the RGB channels is treated as a long vector of length $m \cdot n$. It is shown in [72] that by employing an L2 normalization on each of the three RGB vectors, the effect of any illumination change is approximately compensated. The L2 normalization of a colour C_i is defined as $\frac{\mathrm{C_i}}{\sqrt{\sum_{\mathrm{k=1}}^{\mathrm{n \cdot m}} \mathrm{C_k^2}}}$. The usage of chromaticity provides two additional advantages: (a) the colour space is reduced from 3D to 2D, hence less computations, (b) the chromaticity value is guaranteed to be in the range of [0, 1]. The chromaticity histogram obtained this way is compressed with a symmetrical wavelet low–pass filter [174]. Applying this scaling function to the chromaticity histogram h results in a new histogram $\boldsymbol{H}$. The Discrete Cosine Transform (DCT) of $\boldsymbol{H}$ is denoted $\widehat{\boldsymbol{H}}$. Because the DCT is linear, it is possible to index the image on $\widehat{\boldsymbol{H}}$. Since the lower frequencies in the DCT capture most of the energy of an image, after applying the DCT we can retain just the lower frequency coefficients to index the image with fairly good accuracy. For our prototype implementation we chose to retain 36 coefficients[6].

Populating the database, then, consists of calculating off-line the 36 values $\boldsymbol{H}_d$, viewed as indexes for each image. For the image query, first the 36 values for the query image are computed, thus obtaining $\boldsymbol{H}'_d$; then for every image in the database, the L2 distance $[\sum(\boldsymbol{H}'_d - \boldsymbol{H}_d)^2]^{1/2}$ is calculated. An image minimizing the distance is taken to be a match for the query image. Note that in this method only reduced, DCT transformed, quantized histogram–images are used — no inverse transforms are necessary and the indexing process is entirely carried out in the compressed domain. See [167] for more details.

4.2.2 Template-Based Search

Colour and texture play an important role in content-based image retrieval. Dominant colours and textures usually leave an impression on the user, who often uses these content characteristics to later retrieve the image if needed again. Querying an image repository basically consists of submitting the visual characteristics needed, like colours and textures. The submission of these characteristics can be done by means of templates, which are simply containers where desired colours and textures can be specified. We have distinguished four different templates for search by colour percentage, search by colour layout, search by edge density and orientation, and search by texture layout. Note that a combined search is

[6] By experiment, it is found [72, 257] that using only 36 coefficients worked well, these being those in the first 36 numbers in the upper left corner of the DCT coefficient matrix.

possible by combining the templates.

Search by Colour Percentage

This template is used to define the colours to be present in the image to retrieve. A certain number of colours to be present can be specified with their approximate percentage in the image to retrieve. The specified colours and their percentages are matched to the colour histograms in the database. The matching is not to the exact percentage, but given a threshold Θ, percentages are converted to ranges (percentage $\pm\Theta$).

Search by Colour Layout

This template is used to define localized colours in a 1×1, 2×2, 4×4 or 8×8 grid. The presence of the colour and its approximate location in the image is indicated by defining the desired colour in the appropriate cell of the grid. The 1×1 grid, for example, is used to specify a dominant colour in the images to retrieve. Note that not all cells of the grid have to be filled. In that case, for an empty cell, all possible colours are a match.

Search by Edge Density and Orientation

Two of the known texture measures are *coarseness* and *directionality* [242]. Recent studies [207, 172] also suggest that they are among the few most effective perceptual dimensions in discriminating texture patterns. The directionality is especially useful in handling rotations. The texture features considered are *edge density* (i.e. coarseness), which gives an estimation whether the image (or an area in the image) is highly textured, and *edge orientation* (i.e. directionality). The edge orientations supported are: horizontal, vertical, oblique right, and oblique left, in other words: 0°, 90°, 45°, and 135°. These were chosen just as a proof of concept. Any angle could be chosen too. These orientations are like quantized colours in a 360 colour space. Indeed the search by edge density and orientation template is used to define the edge orientation to be present in the image and its density, exactly like the presence of a colour and its percentage. Again, the matching is not to the exact orientation and density. The four orientations we chose (0°, 90°, 45°, and 135°) divide the 360° circle, and density has been defined by three values: high, medium and low.

Search by Texture Layout

This template is used to define localized edge orientations and their density in a 1×1, 2×2, 4×4 or 8×8 grid. The presence of the edge orientation and its approximate location in the image is indicated by defining the desired orientation in the appropriate cell of the grid. The desired edge density accompanies the edge orientation. Like the colour layout grids, not all cells of the grid have to be filled.

4.2.3 Model-Based Search

In all the content-based image retrieval systems we have studied, there is no known system capable of retrieving images containing a given object model. Model-based search consists of retrieving images containing a given object model regardless of the size, position, or orientation (in 2 D) of the object in the image. We have proposed two approaches to search by object model, one using windowing matching [169], in which the object is matched in a sliding window at different resolution levels, and the other using feature localization [167], in which images are fragmented into rough segments called *locales* and a three step algorithm matches the features in the object with the locales.

The idea behind windowing matching for object search is a recursive search of the object in windows at different resolution levels, starting with a window equal to the complete image. If the object is not identified in the first round (i.e. the window is the whole image) the window is divided into nine overlapping windows, each being 25% of the window (see Figure 4.1). The recursive process at different scales takes care of the different possible sizes the object might have, while the windows deal with the different possible positions the object might have in the image. The different possible orientations are considered within individual windows by first testing the presence of object's MFCs and verifying the vectors connecting the MFC centroids. More details about windowing matching procedure for search by object model using a multi-resolution level recognition kernel can be found in [169].

Object Search with Feature Localization

Image segmentation is a process to segment an entire image into disjoint regions. A region consists of a set of pixels that share certain properties, e.g., similar colour (or gray-level intensity), similar texture, etc. As in [23], if R is a region,

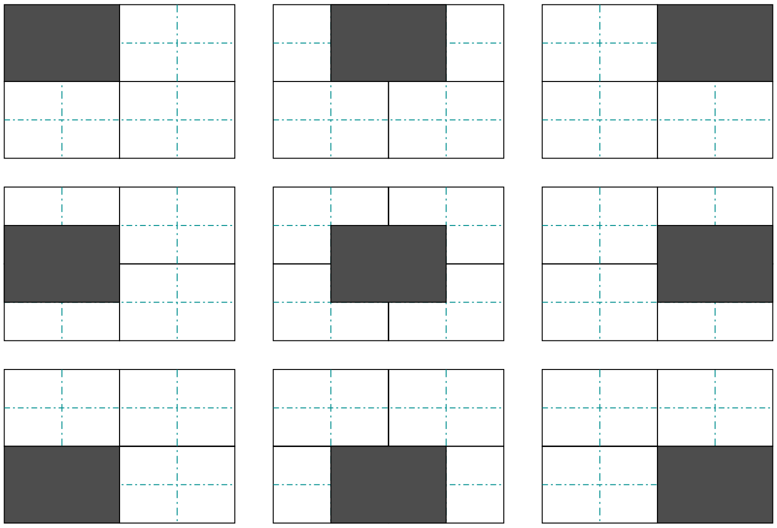


Figure 4.1: Nine overlapping search windows.

1. R is *connected*, if and only if all pixels in R are connected [7],

2. $R_i \cap R_j = \phi, \quad i \neq j$,

3. $\cup_{k=1}^{m} R_k = I$, the entire image.

Although regions do not have to be connected, most available region-based and/or edge-based segmentation methods would yield connected regions, and it is error-prone to merge some of them into non-connected regions. In short, the traditional segmentation algorithms assume (1) regions are mostly *connected*, (2) regions are *disjoint*, (3) segmentation is *complete* in that any pixel will be assigned to some region, and the union of all regions is the entire image.

Such a segmentation algorithm would yield more than a dozen purple regions, one for each character, for the title of the first book shown in Figure 4.9. It would also yield (unexpectedly) many white regions, since all the white blobs inside the letters 'A', 'P', 'R' 'O' will unfortunately be identified as regions unless some really effective algorithm can identify them as belonging to a non-connected region together with the two white boxes. The above example, albeit simple and not at all unusual, indicates that the traditional image segmentation does not yield useful grouping and representation for object recognition.

[7] Either 4-connected or 8-connected. See [23] for more details.

A more useful and attainable process is feature localization that will identify features by their locality and proximity. A new concept *locale* is hence defined in [167] by Li, Zaïane and Tauber.

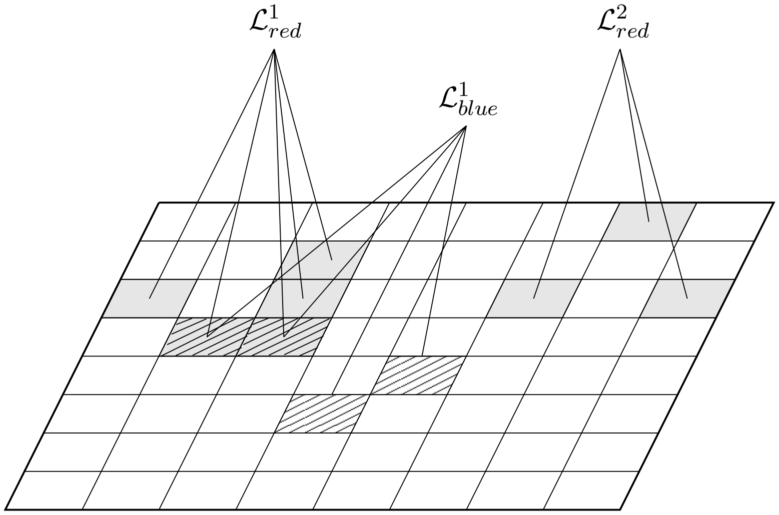


Figure 4.2: An image of 8 × 8 tiles, and locales for colours red and blue.

Definition 4.2.1 *A* locale $\mathcal{L}_x$ *is a local enclosure (or locality) of feature* x.
$\mathcal{L}_x$ *has the following descriptors:*

- *envelope* L_x *— a set of tiles to represent the locality of* $\mathcal{L}_x$*.*
- *geometric parameters —* mass $M(\mathcal{L}_x)$*,* centroid $C(\mathcal{L}_x)$*,* variance $\sigma^2(\mathcal{L}_x)$*, and shape parameters for the locale, etc.* □

A tile is a square area in an image. Its size is arbitrarily chosen as 16 × 16, but could be bigger or smaller. The tile is the building-unit for envelopes. A tile is 'red' if a sufficient number of pixels within the tile are red. It follows that a tile can be both 'red' and 'blue' if some of its pixels are red and some are blue. While a pixel is the unit for image segmentation, a tile is the unit for feature localization. Thus, feature localization is a kind of rough segmentation where overlap is possible and completeness is not necessary.

Tiles are grouped into an envelope, if they are geometrically close. The closeness will be measured by variance to be defined below.

Figure 4.2 shows a square model image that has 8×8 tiles, two locales for the colour red, and one locale for the colour blue. The envelope L^1_{red} in Figure 4.2, for example, consists of 5 tiles.

Since $Area(L_x)$ equals to the maximum number of pixels in L_x that have the feature x, $Area(L_x)$ can also be viewed as the 'magnitude' of $\mathcal{L}_x$.

$M(\mathcal{L}_x)$ is the number of pixels in L_x that actually have feature x, e.g., the number of pixels that are red. $M(\mathcal{L}_x)$ is usually less than the Area of L_x, although it could be equal to it. $C(\mathcal{L}_x)$ is simply the centroid of the mass. $\sigma^2(\mathcal{L}_x)$ is the variance of the Cartesian distance from pixels in L_x to the centroid, it measures the eccentricity of $\mathcal{L}_x$. Note, M, C, σ^2, etc. are measured in unit of pixels, not in tiles. This guarantees the granularity. Hence the feature localization is not merely a low-resolution variation of image segmentation.

The procedure for generating the locales basically uses *merge*. First, simple statistics (M, C, σ^2) are gathered within each tile. Afterwards, a method similar to "pyramid-linking" [134] is used to merge the tiles into locales. In terms of the parent-child relation, the overlapped pyramid is used.

Working bottom-up, all tiles having feature x are linked to their parent and merged into L_x, if the merged envelope will have $\sigma^2(\mathcal{L}_x) < \tau$, where τ is a threshold normalized against the size of $\mathcal{L}_x$. Otherwise, they will be linked to two different parents belonging to different envelopes L^i_x and L^j_x. During the merge, $M(\mathcal{L}_x)$, $C(\mathcal{L}_x)$, and $\sigma^2(\mathcal{L}_x)$ are updated accordingly.

From the above definition, it is important to note that the following can often be true:

1. $(\exists x)\mathcal{L}_x$ is *not connected*,
2. $(\exists x)(\exists y)\mathcal{L}_x \cap \mathcal{L}_y \neq \phi, \quad x \neq y$,
3. $\cup_x \mathcal{L}_x \neq I$, the entire image.

Namely, (1) pixels inside a locale for some features are not necessarily connected, (2) locales are not always disjoint, their envelopes can be overlapped, (3) not all pixels in an image must be assigned to some locale in the feature localization process.

Locale is not simply a variant of non-connected region; the main difference between locale and non-connected region is illustrated by the above property (2). In the proposed feature localization, it is the approximate location that is identified, not the precise membership as which pixel belongs to which region. The difference is not a philosophical one. If indeed only some simple process is to be applied, e.g., template matching, then the precise membership of the region is important. In the domain of content-based image retrieval, where a very large amount of image and video data are processed, such simple and precise matches are not feasible. Instead, a more heuristic (evidential) process is going to be adopted which usually involves multiple features and their spatial relationships. For this purpose, it should be evident that the 'blobby' locales are easier to extract, and more appropriate than regions formed by (connected) pixels.

As illustrated in Figure 4.9, perceptually what is important are the two purple words "Active Perception" on a white background which in turn appears at the upper portion of a pink cover. The colour, shape, location, spatial relationship, etc. provide a rich set of cues for the recognition of this book cover.

Property (3) indicates that, unlike the image segmentation, the feature localization is incomplete. Colour localization, for example, concentrates on dominant colours and does not take notice of small noise spots (rare colours or isolated few pixels). When only the locales of the few prominent colours are identified, the union of them is not the whole image.

We present a three-step matching algorithm for searching by object models in image and video databases, i.e., (1) colour hypothesis, (2) texture support, (3) shape verification. It is generally accepted that colour is fairly invariant to scaling and rotation, and it can be well-preserved in relatively low resolution images to save computing time. After colour localization, a hypothesis of the existence of an object at a certain location, size and orientation can be made. If there is a sufficient similarity in their texture between the object model and the image at the vicinity of the hypothesized enclosure, then a shape verification procedure based on the Generalized Hough Transform will be invoked.

The Generalized Hough Transform (GHT) [22] is adopted to represent the shape of the object. Briefly, each edge point in the object model is represented by a vector $\mathbf{r}_i$ connecting the edge point to a chosen reference point for the object. All $\mathbf{r}_i$s are stored in an R-table which serves as an object model. The R-table is indexed by the edge orientation ϕ_i of the edge point.

The major advantage of the GHT (and its variants) over other shape representations [194] is its insensitivity to noise and occlusion [171, 118]. It can also be applied hierarchically to describe the object (or a portion of the object) at multiple resolutions. It is known that the discriminative power of the GHT diminishes when the aim is to recognize the object at all possible scales, orientations, and locations. However, in our algorithm, GHT is only applied after a certain hypothesis of a possible object size, orientation, and location is made.

For both colour and shape, there is an issue of *similarity.* It is dealt with effectively using the MFC and MFO vectors and the geometric parameters of the locales. First, if the model object appears in the image with exactly the same size and orientation, then the mass M, variance σ^2 of each locale, the length ρ_i and orientation α_i of each MFC or MFO vectors, and the angles ϕ_j between the pairs of the MFC or MFO vectors are all identical, whether they are extracted from the model or from the object in the image. Second, if the object in the image has a different size and/or orientation, then M and ρ_i should be scaled according to the size ratio, α_i should be incremented by a rotational angle, whereas ϕ_j would remain the same. Certain tolerance for error (using thresholds) is implemented to support the similarity. In summary, the matching algorithm is condensed in Algorithm 4.2.1 as described in [167]:

Algorithm 4.2.1 Find the images that contain a given object model.

Input: (i) Object model; (ii) Locale descriptors of all images in database.

Output: Set of images that match the object model.

Method. After analyzing the object model, progressively eliminate images that can not be potential candidates: Step 1: extract descriptors for model; step 2: keep only images that match the model colours; step 3: keep only images that match texture of the model. step 4: keep only the images that match shape. The algorithm is outlined as follows:

```
begin
        /* ANALYSE MODEL */
              /* Image 'tiling' */
(1)     Within each $8 \times 8$ tile of the image model (with a reduced-resolution)
(2)           Gather $M$, $C$, $\sigma^2$ for each MFC associated with the object model;

        /* Colour localization */
(3)     Use overlapped pyramid linking to group tiles into locale $\mathcal{L}$'s for each MFC;

        /* MATCHING IMAGES */
```

```
(4)   For all images do {
          /* Colour hypothesis */
(5)       Starting at each $C(\mathcal{L}_{first-MFC})$,
(6)           if (number-of-MFC-vectors $\geq 1$) and they are ‘similar’ to
              the MFC-vectors in the model
(7)               Make hypothesis of matched object size, orientation, and enclosure;
                  Proceed to check texture do {
                      /* Texture support */
(8)                   At the vicinity of the hypothesized enclosure
                      if (number-of-MFO-vectors $\geq 1$) and they are consistent with
                      the hypothesized object size, orientation
(9)                       Proceed to check shape using the GHT do {
                          /* Shape verification*/
(10)                      Within the enclosure of the hypothesized object
(11)                          All edge pixels use R-table of the (rotated/scaled)
                              object model to vote;
(12)                          if number-of-votes near the reference point $> \tau_0$
(13)                              Confirm the detection of the object;
(14)                      }
(15)              }
(16)  }
end
```

□

This algorithm is basically a progressive refinement that goes over all images but avoids evaluating all visual features for all the images when not necessary. The complexity of this algorithm may vary greatly depending upon the availability of indexes for the visual features tested.

4.3 Implementation

Our image retrieval system C-BIRD has been implemented on both Unix and PC platforms. On both platforms, we used the same search engine and pre-processor written in C^{++}.

One version of the user interface is implemented in Perl and HTML as a Web application, another version is implemented as a java applet. Another user interface was also developed in C^{++} for the PC platform [265]. Figure 4.3 shows the general architecture for C-BIRD implementation. The system is accessible from http://jupiter.cs.sfu.ca/cbird/cbird.cgi, and http://jupiter.cs.sfu.ca/cbird/java/ (IE 4.0 or Netscape 4.0).

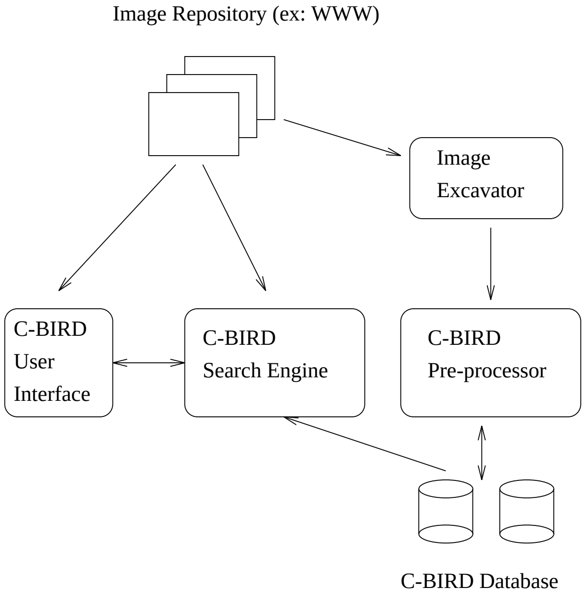


Figure 4.3: C-BIRD general architecture.

C-BIRD system rests on four major components:

- Extraction of images (Image Excavator);
- Processing of images to extract image features and storing precomputed data in a database (Pre-Processor);
- Querying (User Interface);
- Matching query with image features in the database (Search Engine).

The Image Excavator extracts images and video frames from a multimedia repository. This repository can be the WWW space, in such case, the process crawls the Web searching for still images and video streams, or a set of images and videos on disk or CD-ROM. Once images are extracted from the repository, they are given as input to the image analyzer

(C-BIRD pre-processor) that extracts visual content features like colour and edge characteristics. These visual features, along with the context feature like image URL, parent URL[8], keywords, etc., extracted with the Image Excavator, are stored in a database[9]. The collection of images and the extraction of image features are processed off-line before queries are submitted. When a query is submitted, accessing the original data in the image repository is not necessary. Only the precomputed data stored in the database is used for image feature matching. This makes C-BIRD more scalable and allows fast query responses for a large number of users and a huge set of images. When queries are submitted, only two processes are in action: the user interface interacting with users, and the search engine accessing and matching precomputed data. The user interface communicates with the search engine with a set of primitives. This allows having different user interface implementations. The search engine accesses the database of the image visual and contextual features. If necessary, both the user interface and the search engine can access the images using their URL. We have implemented eight types of searches in C-BIRD as described in Section 4.2:

1. search by conjunctions and disjunctions of keywords;
2. search by colour histogram: similarity with colour histogram in a sample image;
3. search by illumination invariance: similarity with colour chromaticity in a normalized sample image;
4. search by colour percentage: specification of up to 5 colours and percentages;
5. search by colour layout: specification of the layout of colours in a 1×1, 2×2, 4×4 or 8×8 grid;
6. search by edge density and orientation;
7. search by edge layout: specification of edge density and orientation in a 1×1, 2×2, 4×4 or 8×8 grid;
8. search by object model: specification of an object to look for in images.

A combination of these searches is also possible. The left of Figure 4.3 shows the user interface using Netscape to browse the image repository or the image set resulting from a

[8] Parent URL is the URL of the web page containing the image

[9] This database is equivalent to the relation *image* in the first layer of the MLDB presented in Chapter 2.

query. While browsing, users can submit a query by image similarity. The right of Figure 4.3 shows a user interface to specify colour layout for a given query.

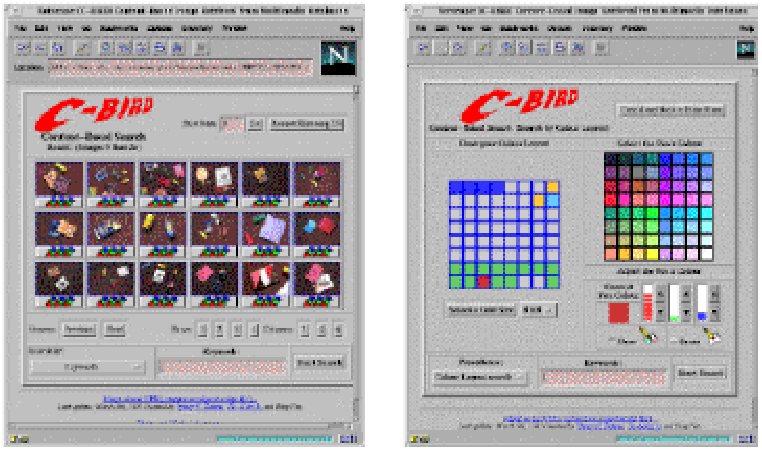


Figure 4.4: C-BIRD Web user interface.

4.3.1 Retrieving the images from the World-Wide Web

The advantage of using the images available on the WWW is two-fold. First, the WWW provides us with a huge image repository which is a superb opportunity to test the efficiency and scalability of our implementation. Second, by using the images available on the WWW we can build an index for the WWW and contribute in the construction of the MLDB structure of the VWV. This not only allows finding and retrieving images but also finding resources containing or referring to given images. Moreover, indexing images by sites can give interesting site content summaries by displaying thumbnail-sized images from a given Web site. Images from web pages are surprisingly representative of the associated textual content. Thus, browsing thumbnail-sized images from a site can give a broad idea about the content of the site.

To retrieve images from the WWW, we built a web spider (Excavator) that crawls the Web and downloads HTML pages and images. While images are analyzed by the pre-processor to extract content features, HTML pages are parsed to extract links to images and other HTML pages as well as descriptive information about images. When parsing a Web page, the Excavator extracts HTML IMG and EMBED tags and identifies image and video URLs. Subsequently, these images are downloaded and passed to the pre-processor. Images are disregarded if they are dynamic (i.e. generated by a CGI), too small, or contain less than a certain number of colours (example: 3). Thumbnails are generated for the remaining images.

Web pages on the WWW contain not only images, but also contextual information "describing" the images that can be extracted from text neighbouring the images. The descriptive text can be used to educe keywords related to images. Being semi-structured, sections and components of an HTML pages can disclose valuable information about an image contained in the page. HTML structure components provide hints for keywords extractions like: image file name and path if it contains a word or recognizable words, ALT field in the IMG tag, HTML page title, HTML page headers, parent HTML page title, hyperlink to the image from parent HTML page, and neighbouring text before and after the image. The new META tag placed in the HEAD element of the HTML page, if available, can provide valuable keywords extracted from the description and keywords sections. The set of all words collected this way, is reduced by eliminating "empty" words like articles (i.e. the, a, this, etc.) or common verbs (i.e. is, do, have, was, etc.), or aggregating words from the same canonical form (e.g.: clearing, cleared, clears, clear) as presented in [273]. There are 400 frequently found words in English, defined in [261], that can be considered of low semantic information content and thus, can be eliminated (stopwords). The automatically generated keyword list can later be manually enriched.

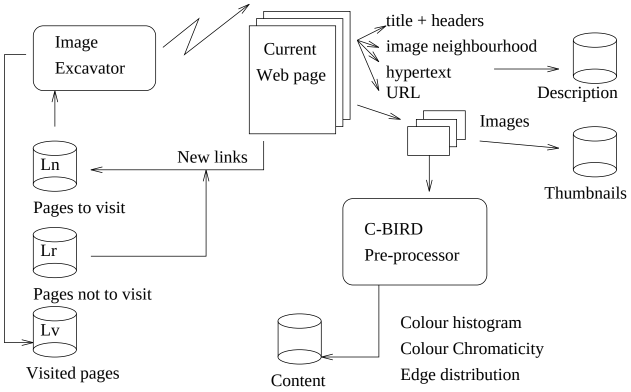


Figure 4.5: Excavator: The Web crawling process for image extraction.

Figure 4.5 illustrates the process of extracting images from the WWW. From an initial set of pages, the Excavator recursively fetches pages, parses them, and adds their hyperlinks to the list of pages to visit. Links are extracted from hyperlink references, frames tags, image maps, and client pull references in HTML pages. The Excavator uses 3 lists: a list of pages to visit (Ln), a list of pages not to visit or restrictions (Lr), and a list of visited pages (Lv). The 2 first lists, Ln and Lr, allow the restriction of image extraction from a given site, a given path or directory, or even a given page. In return, this can give interesting site content summaries by displaying thumbnail-sized images from a given Web site. Images from web pages are surprisingly representative of the associated textual content. Thus, browsing thumbnail-sized images from a site can give a broad idea about the content of the site.

Lr limits the exploration by the Excavator. The Excavator adheres to the Standard for Robot Exclusion[10] initiated by Martijn Koster, which allows webmasters, by means of a robot.txt file, to specify areas and directories of the Web site that should not be visited by spiders. Since the goal of the Excavator is not to index Web pages but to retrieve and index images, the Excavator follows the restrictions specified in the robot.txt file when adding a new URL to Ln. These restrictions, added to Lr, are not considered when downloading an image. Robot META tags present in some HTML pages are also not considered for the same reasons.

When parsing a Web page, the Excavator extracts HTML IMG and EMBED tags and identifies image and video URLs. Subsequently, these images are downloaded and passed to the pre-processor. Images are disregarded if they are dynamic (i.e. generated by a CGI), too small or contain less than a certain number of colours (example: 3). Thumbnails are generated for the remaining images. The Excavator retrieves textual information from the parsed Web pages to automatically generate keywords associated with images contained in the pages, and to be used in queries in conjunction with content-based features. When an image URL is identified, the path leading to the image is used to retrieve other images possibly stored in the same directory. On a web site, images are usually stored in the same sub-directory (ex: gifs, images,...).

Because the URL of the page containing an image is stored with the image meta-data, given an image, it is very easy to find the Web pages in which it is located. This allowed us to build an image-based index on top of the course material web site for a multimedia course at Simon Fraser University. Students remembering passages of the course notes by the images contained in them and not their textual content, can search for the images with

[10] The standard for robot exclusion is available at: http://info.webcrawler.com/mak/projects/robots/exclusion.html

C-BIRD then link to the pages using the parent URL. Figure 4.6 shows an example of an output from the Image Excavator after parsing a web page. The output shows images and their automatically generated keywords.

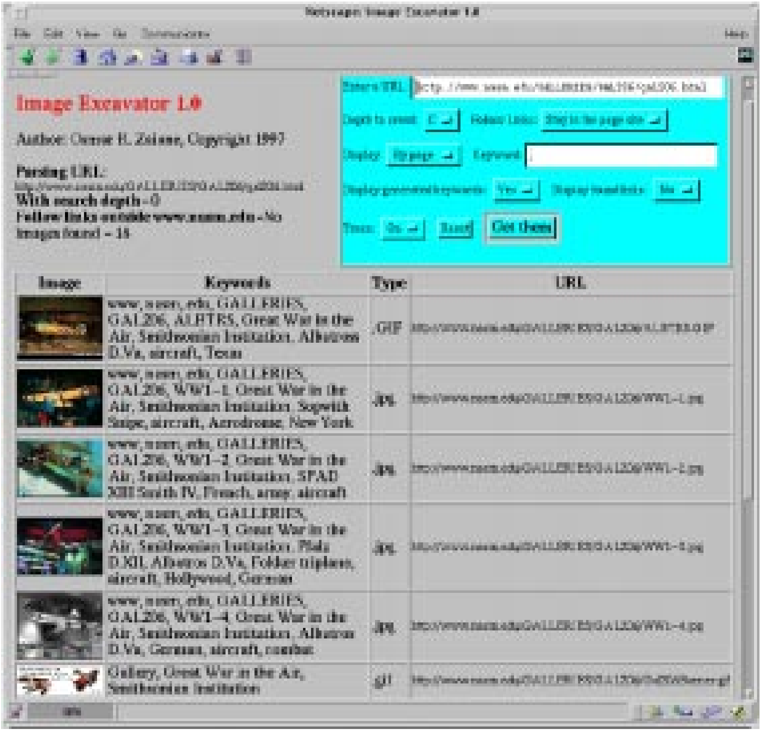


Image Excavator 1.0

Author: Osmar R. Zaïane, Copyright 1997

Parsing URL:
http://www.nasm.edu/GALLERIES/GAL206/gal206.html
With search depth - 0
Follow links outside www.nasm.edu - No
Images found - 18

Enter URL: http://www.nasm.edu/GALLERIES/GAL206/gal206.html

Depth to crawl: 0 Follow Links: Stay in the page site

Display: By page Keyword:

Display generated keywords: Yes Display found links: No

Trace: On Reset Get them

Image	Keywords	Type	URL
	www, nasm, edu, GALLERIES, GAL206, ALBTRS, Great War in the Air, Smithsonian Institution, Albatross D.Va, aircraft, Texas	.GIF	http://www.nasm.edu/GALLERIES/GAL206/ALBTRS.GIF
	www, nasm, edu, GALLERIES, GAL206, WW1-1, Great War in the Air, Smithsonian Institution, Sopwith Snipe, aircraft, Aerodrome, New York	.jpg	http://www.nasm.edu/GALLERIES/GAL206/WW1-1.jpg
	www, nasm, edu, GALLERIES, GAL206, WW1-2, Great War in the Air, Smithsonian Institution, SPAD XIII Smith IV, French, army, aircraft	.jpg	http://www.nasm.edu/GALLERIES/GAL206/WW1-2.jpg
	www, nasm, edu, GALLERIES, GAL206, WW1-3, Great War in the Air, Smithsonian Institution, Pfalz D.XII, Albatros D.Va, Fokker triplane, aircraft, Hollywood, German	.jpg	http://www.nasm.edu/GALLERIES/GAL206/WW1-3.jpg
	www, nasm, edu, GALLERIES, GAL206, WW1-4, Great War in the Air, Smithsonian Institution, Albatros D.Va, German, aircraft, combat	.jpg	http://www.nasm.edu/GALLERIES/GAL206/WW1-4.jpg
	Gallery, Great War in the Air, Smithsonian Institution	.gif	[illegible]

Figure 4.6: Output from the Image Excavator

4.3.2 C-BIRD database

The database used by C-BIRD is an addition to the image repository and contains mainly meta-data extracted by the pre-processor and the Image Excavator. As explained above, only features collected in this database at pre-processing time are used by the search engine for image or image feature matching. During run time, minimal processing is done. For each image collected, the database contains some description information, a feature descriptor, and a layout descriptor, as well as a set of multi-resolution sub-images (i.e. search windows) feature descriptors. Neither the original image nor the sub-images are directly stored in the database; only their feature descriptors are stored.

The description information encompasses fields like: image file name, image URL, image type (i.e. gif, jpeg, bmp,...), list of all known web pages referring to the image (i.e. parent URLs), a list of keywords, and a thumbnail used by C-BIRD user interface for image and video browsing.

The feature descriptor is a set of vectors for each visual characteristic. The main vectors are: a chromaticity vector containing 36 values, a colour vector containing the colour histogram quantized to 64 colours ($4 \times 4 \times 4$ for $R \times G \times B$), locale vector, MFC vector, and MFO vector. The MFC and MFO contain 5 colour centroids and 4 edge orientation centroids for the 5 most frequent colours and 4 most frequent orientations. These centroids are used to derive the MFC and MFO vectors in the recognition kernel.

The layout descriptor contains, a colour layout vector, and an edge layout vector. These vectors allow matching with user defined layouts. Regardless of their original size, all images are assigned an 8×8 grid. The most frequent colours for each of the 64 cells are stored in the colour layout vector and the number of edges for each orientation in each of the cells is stored in the edge layout vector. The latter is used for both search by edge density and search by edge orientation layout.

All these attributes and vectors constitute a subset of the relation *image* of the VWV shown in Example 4.0.1.

Since the recognition kernel searches for objects in each search window at a given resolution level, each sub-division (i.e. search window) is represented with a feature descriptor like the full image at the highest resolution level. These feature descriptors for the sub-images are stored with the image meta-data.

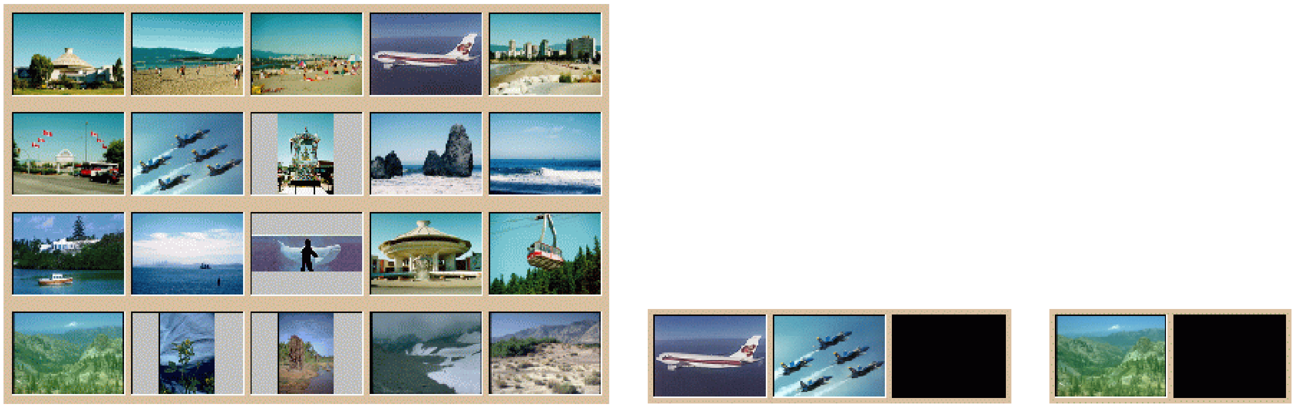
We use our illuminance invariant method to detect cuts in videos, and segment a video clip into frame sequences. The starting time and duration of the image sequence are stored with the meta-data. While the thumbnail is generated from the first frame, colour and texture features are extracted from all frames.

The database was originally implemented with miniSQL database and structured files on the Unix platform. The version running on Windows NT is using an SQL server.

4.3.3 Content-based retrieval results

C-BIRD on both platforms, has a simple and friendly user interface that allows querying by simple mouse clicks, browsing, and composing conjunctions of complicated queries. The current test database has over 1,300 images. The meta-data is stored in a SQL server running on a Pentium-II 333 MHz with 128 MB RAM. Search times are in the order of 0.1

to 2 seconds, depending upon the type of search, except for the search by object, which may take up to 10 seconds to make comparisons in all sub-windows in the different resolutions and do all the necessary rotations. Notice that the search by object model begins by selecting only images that may potentially contain the object by shortlisting the images that contain the colours present in the object.



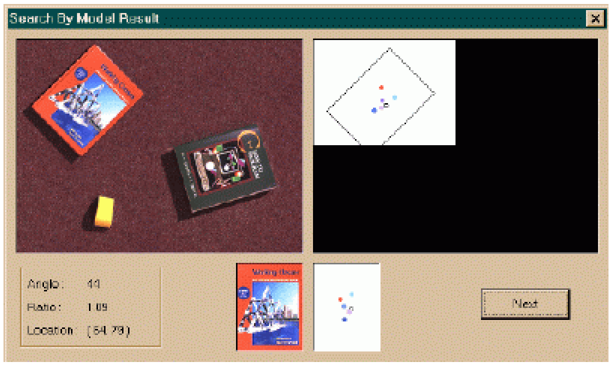
(a) (b) (c)

Figure 4.7: Conjunction of searches.

Figure 4.7 demonstrates the use of conjunction of different searches, content-based and description-based. Figure 4.7 (a) is the top-20 matches of a query based on the colour layout where the top cells were blue (i.e. for "blue sky"). Figure 4.7 (b) is the result for a combination of content-based and description-based query, with the same colour layout specified as for Figure 4.7 (a) and an additional keyword "airplane". Figure 4.7 (c) is the result of the query "blue sky and green grassland" specified with a colour layout grid with the top cells blue, the bottom cells green and a medium edge density.

Search by object model

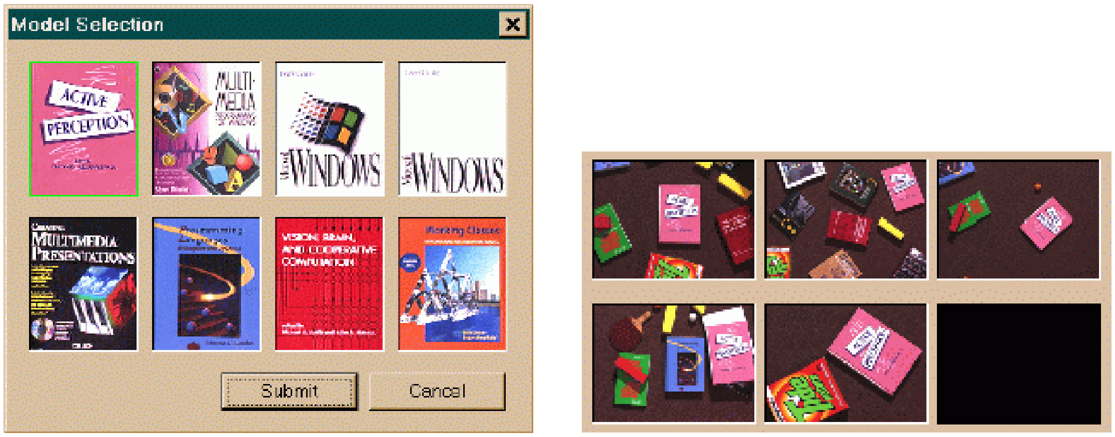
Figure 4.8 shows detailed results of a search with multilevel resolution window matching. The model book image and the centroids of the 5 MFCs are shown at the bottom of the figure. Among the 5 MFCs the colour orange-red is the first MFC. As can be seen from the original image at the left, the sought book is at the upper-left quarter. The correct match occurs in the first of the nine search windows. The graphical display of the search window, the located book and locations of the colour centroids are shown at the right of the figure. The book orientation (44°) and scale-ratio (1.09) are calculated using the weighted-average



Angle: 44
Ratio: 1.09
Location: (64,79)

Figure 4.8: Search by Object Model with MultiLevel Resolution Window Matching.

of the orientations and the lengths of the MFC vectors (vectors connecting the centroid of orange-red and the centroids of the other MFCs), respectively. Accordingly, the position (centre of the book) is determined to be at (64, 79) which corresponds to the resolution of the bottom level of the recognition kernel. The search continues at the third level of the recognition kernel where the edge orientations and the MFO vectors are checked and confirmed.



(a) (b)

Figure 4.9: (a) Object Models. (b) All solutions for pink book.

The search by object model using feature localization proceeds differently and uses the Locales introduced in Section 4.2.3.

We first identify the pixels with dominant colours and the colours that the transitional pixels would merge into. Transitional pixels are changed to the closest dominant colour in their neighbourhood.

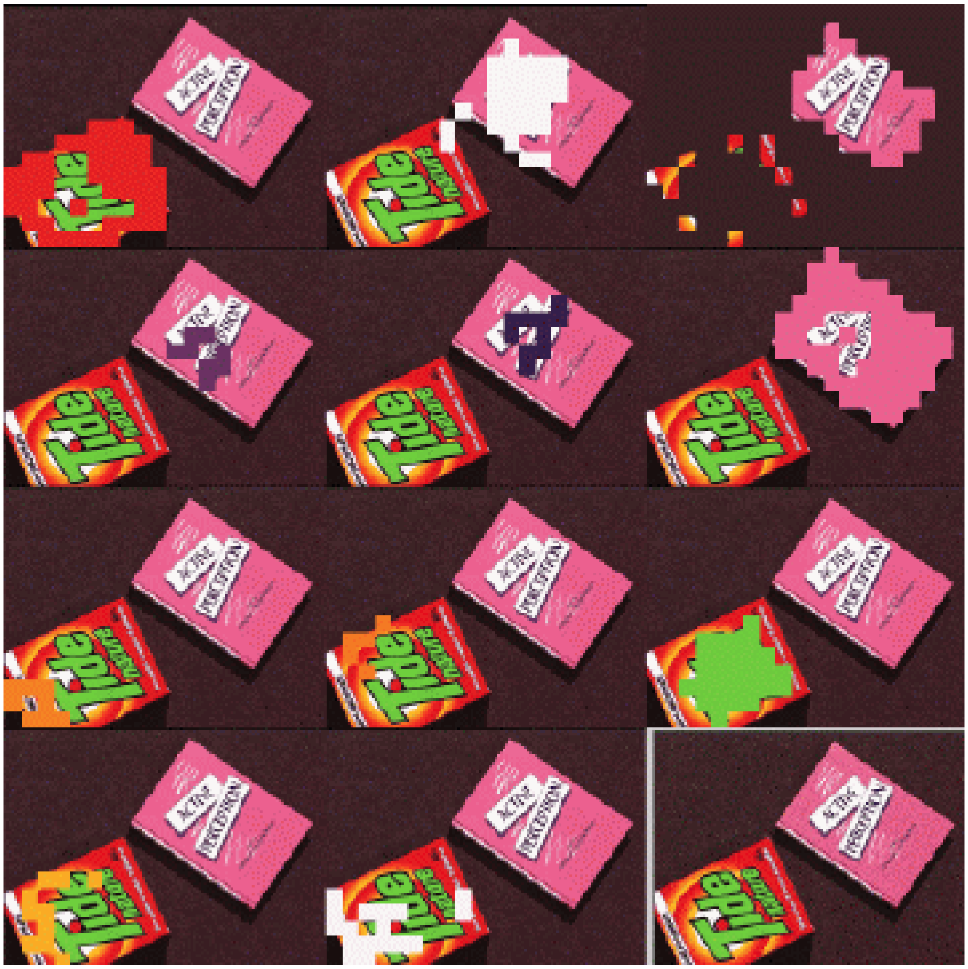
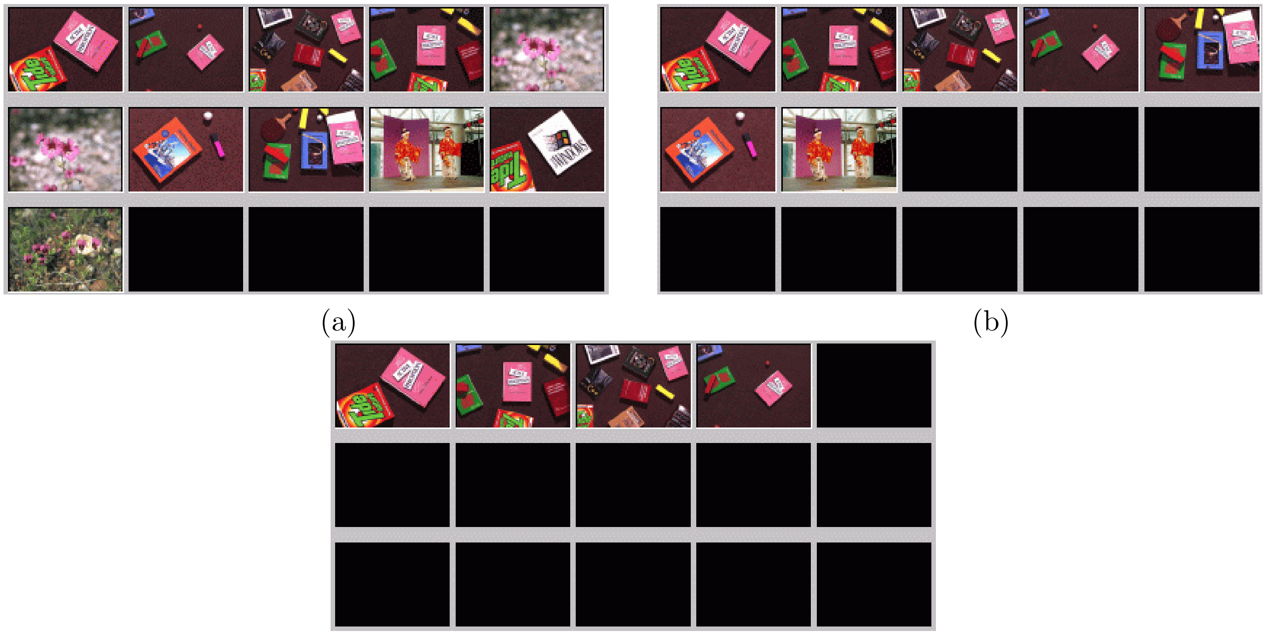


Figure 4.10: The tiles generated for the sample image

We generate the image tiles array using the dominant colours we identified, and then generate all the Locales for the image. An example of different Locales generated from an image are shown in Figure 4.10. The original image, is the image in the bottom-left corner of Figure 4.10. Most features are correctly enveloped. In particular, the white colours of the book and the "Tide" box are split into 2 locales, yet other locales with similar mass are not split. The orange rainbow colour on the "Tide" box is split into 2 locales because it is not compact enough, but has a large mass. The dark brown background is merged with all the black colours in the lower book because the black pixels are just as close to the background as they are to each other and the compactness of the merged Locale is good enough.



(a)

(b)

(c)

Figure 4.11: Three-step matching: (a) Retrieved Images after the colour hypothesis; (b) Images that also have texture support; (c) Images that finally pass the shape verification step.

Figure 4.9 illustrates an example of search by object model using feature localization. Figure 4.9(a) shows the eight book models of which the first book is selected. The object model in this case is a book. All five occurrences in our test database of this book with various sizes, positions and orientations are shown in Figure 4.9(b). In the current implementation only a few book models as shown in Figure 4.9(a) are tested. In the future, users will be able to crop out any object/pattern in any image and use it as a model. Figures 4.11(a), 4.11(b) and 4.11(c), respectively show the results of each of the three steps in our matching algorithm. Figure 4.11(a) shows the preliminary result after the colour hypothesis (step 1). All the images retrieved contain the MFCs of the object model. We can see that in Figure 4.11(b), after the texture support has been verified, the tree pictures with flowers have been eliminated. Figure 4.11(c) is the final result after the shape verification step. Four occurrences of the object model (pink book) with various sizes, positions and orientations are correctly retrieved. The result has a perfect precision (100%), but the recall is only 80%. Indeed, there are five occurrences of the object in the database (Figure 4.9(b)) but the final result shows four, despite the fact that all five occurrences of the book appear in the preliminary results in Figures 4.9(a) and 4.9(b). The missing occurrence is due to the fact that the pink book was placed on a white sheet overlapping from the top. The white area next to the top of the pink book was merged with the white locale of the book, hence the rejection by the third step even though the pink book was hypothesized at the first two steps. The algorithm mistakenly interpreted the two objects (white paper and pink book) as one with a different shape.

4.4 Conclusion and Discussion

Content-based image retrieval is an important issue in the research and development of digital libraries which usually employ large multimedia databases. This Chapter presented our prototype system C-BIRD for content-based image retrieval from large image and video databases. Issues in both database design and image content based retrieval are addressed. A multi-level recognition kernel is developed to support search by model. Unlike most existing systems which use only global image features (colour, texture, etc.), the modeling and matching methods described are capable of handling a range of different sizes, 2-D rotation, and multiple occurrences of the objects in the images. Feature localization and a three-step matching algorithm are presented to support Search by Object Model. It is shown that instead of image segmentation, feature localization should be used as a preprocessing step before matching.

Several content-based image and video retrieval systems use region-based search methods. For example, QBIC [99] uses rectangular shaped coloured regions; Video-Q [225] keeps the description and spatial relationship of regions, so that user can sketch the trajectory of moving colour regions for the retrieval of certain moving objects. These systems rely heavily on a good segmentation preprocess and they do not have a systematic means of retrieving objects. To the best of our knowledge, C-BIRD is the first system that successfully performs object model search from image and video databases.

This work also shows how pertinent information can be extracted from images and videos from a global network information system in order to build the first layer of the Multiple-Layered Database (MLDB) structure defined in Chapter 2. Concept hierarchies could be defined on colours, textures and other visual and non-visual features to aid in the generalization process towards higher layers. Such an approach is adopted in the next Chapter for OLAP and data mining from visual media repositories.

The desire of knowledge, like the thirst of riches, increases ever with the acquisition of it.

LAURENCE STERNE, TRISTRAM SHANDY

Everything is simpler than you think
and at the same time more complex than you imagine.

JOHANN WOLFGANG VON GOETHE

Chapter 5

OLAP and Data Mining from Visual Media

Data Mining is a young but flourishing field. Many algorithms and applications exist to mine different types of data and extract different types of knowledge. Mining multimedia data is, however, at an experimental stage.

We have implemented a prototype for mining high-level multimedia information and knowledge from large multimedia databases. MultiMediaMiner has been designed based on our experience in the research and development of a relational data mining system, DBMiner[120, 119, 121], and a Content-Based Image Retrieval system from Digital Libraries, C-BIRD, described in the previous Chapter.

MultiMediaMiner includes the construction of multimedia data cubes which facilitate multiple dimensional analysis of multimedia data, and the mining of multiple kinds of knowledge, including summarization, classification, and association, in image and video databases. The images and video clips used in our experiments are collected by crawling the WWW. Many challenges have yet to be overcome, such as the large number of dimensions, and the existence of multi-valued dimensions.

Substantial progress in the field of data mining and data warehousing research has been witnessed in the last few years. Numerous research and commercial systems for data mining and data warehousing have been developed for mining knowledge in relational databases and data warehouses [92]. Despite the fact that Multimedia has been the major focus for many researchers around the world, data mining from multimedia databases is still in its infancy.

While one of the first dominant and referenced papers in the field of knowledge discovery by Fayyad et al.[90, 91] describes discovering patterns from satellite pictures, multimedia mining still seems shy on results. Many techniques for representing, storing, indexing, and retrieving multimedia data have been proposed. However, rare are the researchers who ventured into the multimedia data mining field. Most of the studies done are confined to the data filtering step of the KDD process as defined by Fayyad et al. in [202]. In [63], Czyzewski shows how KDD methods can be used to analyze audio data and remove noise from old recordings. Chien et al. in [54] use knowledge-based AI techniques to assist image processing in a large image database generated from the Galileo mission. Others use multimedia to complement data mining systems. Bhandari et al. [30], for instance, marries a data mining application with multimedia resources. His application does not claim to mine a multimedia database, but uses video clips to support the knowledge discovered from a numerical database. More recently, Tucakov and Ng in [248] used a method for outlier detection to identify suspicious behaviour from videos taken by surveillance cameras.

Multimedia data mining is a subfield of data mining that deals with the extraction of implicit knowledge, multimedia data relationships, or other patterns not explicitly stored in multimedia databases. A multitude of applications can benefit from multimedia data mining such as interesting pattern discovery in medical imaging, global weather understanding from satellite and radar imagery, patterns detection in surveillance cameras, solar storms understanding, etc. Multimedia data mining is not limited to images, video or sound, but encompasses text mining as well. There has been interesting research in text mining from text documents[93, 94] and Web or semi-structured data querying and mining[276, 149, 81, 182]. The availability of affordable imaging technology is leading to an explosion of data in the forms of image and video. Many relational databases are now including multimedia information, such as photos of customers, videos about real estate, etc. The proliferation of huge amounts of multimedia data is becoming prominent. Global information networks like the Internet, as well as specialized databases, are filled with a variety of multimedia, medical images, satellite pictures, etc., necessitating means to retrieve, classify and understand this data. Moreover, with the popularity of multimedia objects in extended and object-relational databases, it is becoming important to mine knowledge related to both multimedia and relational data in large databases, and maybe, to deal with them in the same manner.

Most of the recent work on multimedia systems has concentrated on transmission, synchronization and management of continuous data streams of audio, video and text. Other fields, no less important, are authoring, coding, indexing and retrieving of media data. The last focused area has drawn the attention of many. Researchers, for instance, try to "summarize" video clips in one image. Salient stills were introduced in [247], in an attempt to represent an abstract of a video clip in one still image. The salient stills reflect aggregates of temporal changes that occur in a moving image sequence. Stills are created automatically or with user intervention by combining affine transformation and multiple frames of the image sequence. Taniguchi et al.[243] use "mosaicing" to glue overlapping video frames to create a panoramic still image representing the video sequence. Despite the fact that representing a video clip in one still image summarizes in a way video clips, it is hard to claim that this is data mining from video.

With huge amounts of multimedia data collected by video cameras and audio recorders, satellite telemetry systems, remote sensing systems, surveillance cameras, and other data collection tools, it is crucial to develop tools for discovery of interesting knowledge from large multimedia databases.

In addition, many relational databases start including multimedia information as well, such as the photos of a customer, etc. Therefore, it is important to mine knowledge related to both multimedia and relational data in large databases. Unfortunately, there have not been many multimedia data mining systems reported in previous studies.

Recent advances in the research on multimedia databases [144, 48, 195, 99] enable creation of large multimedia databases which can be queried in an effective way. These advances, in combination with the research into multimedia database and advances in data mining in relational databases [92], created a possibility for the creation of multimedia data mining systems.

We have DBMiner system [119, 122, 120] and the C-BIRD system [169, 167] to manipulate and interpret multimedia data for knowledge discovery purposes.

The current MultiMediaMiner system, which was demonstrated at the SIGMOD98 conference, includes a module for *characterization* of knowledge in image and video databases, a module for *classification* of multimedia data, and a module for detection of *association* between multimedia features.

A more detailed description of the MultiMediaMiner system is presented in Section 5.1. The challenges and obstacles that we encountered with mining multimedia data, and the turn-arounds for our prototype implementation are presented in Section 5.2. Section 5.3 summarizes our on-going research. Finally, a section is dedicated to multimedia association rules which attempt to go beyond the rules discovered by the MM-associator of the current MultiMediaMiner system, by integrating more content-based descriptors.

5.1 A database mining system prototype

The MultiMediaMiner system is based on our experiences in the development of an on-line analytical data mining system, DBMiner, and C-BIRD, a system for Content-Based Image Retrieval from Digital libraries.

The DBMiner system[1], demonstrated in SIGMOD'96, KDD'96/97, CASCON'96/97, and other conferences, currently contains the following five data mining functional modules: *characterizer, comparator, associator, predictor*, and *classifier.* A general description of these functional modules is in [120]. Several additional functional modules, especially with time-related data, clustering, and visual data mining, are at the research and development stage. DBMiner applies multi-dimensional database structures [120], attribute-oriented induction [119], multi-level association analysis [121], statistical data analysis, and machine learning approaches for mining these different kinds of rules in relational databases and data warehouses. C-BIRD system, presented in Chapter 4, was demonstrated in CASCON97 (some of the function modules can be played on the Internet interactively via http://jupiter.cs.sfu.ca/cbird/). It contains four major components: (i) Image Excavator (a web agent) for the extraction of images and videos from multimedia repositories, (ii) a pre-processor for the extraction of image features and storing precomputed data in a database, (iii) a user interface, and (iv) a search kernel for matching queries with image and video features in the database. C-BIRD allows searches by conjunctions and disjunctions of keywords, colour histograms, colours with illuminance invariance, colour percentage, colour layout, edge density, edge orientation and texture coarseness. In particular, C-BIRD is characterized by its ability to cope with significant changes in image chrominance and to search by object model. The database used by C-BIRD is an addition to the image repository and contains mainly meta-data extracted by the pre-processor and the Image Excavator, like colour, texture, and shape characteristics and automatically generated keywords. MultiMediaMiner, the general architecture of which is shown in Figure 5.1, inherits the CBIRD database.

[1] Some of the function modules can be played on the Internet interactively via http://db.cs.sfu.ca/DBMiner with a web-based user interface we developed, see Figures 3.2 to 3.5.

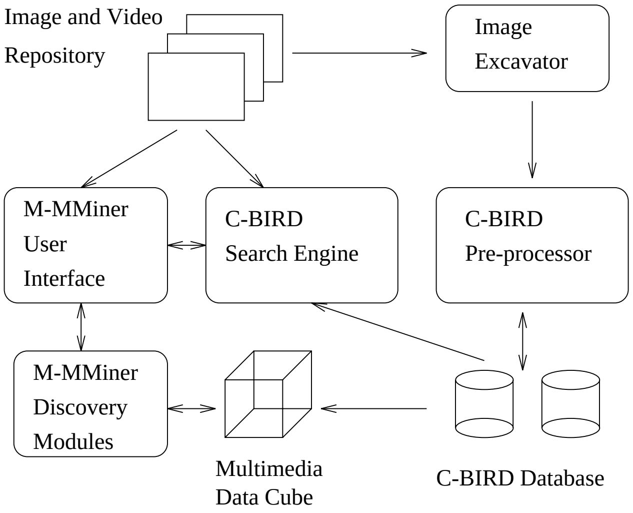


Figure 5.1: General Architecture of MultiMediaMiner.

The Image Excavator and the pre-processor have been enhanced to collect and preprocess more information necessary for the MultiMediaMiner. Video clips are segmented after cuts have been detected. Each video segment is represented by one or more video frames which are later treated and processed by the system like images. For each image collected, the database contains some descriptive information, a feature descriptor, and a layout descriptor. The original image is not directly stored in the database; only its feature descriptors are stored. The descriptive information encompasses fields like: image file name, image URL, image and video type (i.e. gif, jpeg, bmp, avi, mpeg, . . .), a list of all known web pages referring to the image (i.e. parent URLs), a list of keywords, and a thumbnail used by the user interface for image and video browsing. The feature descriptor is a set of vectors for each visual characteristic. The main vectors are: a colour vector containing the colour histogram quantized to 64 colours (all colours are represented in the RGB space by 4 values in red, 4 values in green and 4 values in blue), MFC (Most Frequent Colour) vector, and MFO (Most Frequent Orientation) vector. The MFC and MFO contain 5 colour centroids and 4 edge orientation centroids for the 5 most frequent colours and 4 most frequent orientations (the edge orientations used are: 0°, 45°, 90°, 135°). The layout descriptor contains a colour layout vector and an edge layout vector. These vectors allow matching with user-defined layouts as in the user interface shown at the right of Figure 4.4(a). Regardless of their original size, all images are assigned an 8×8 grid. The most

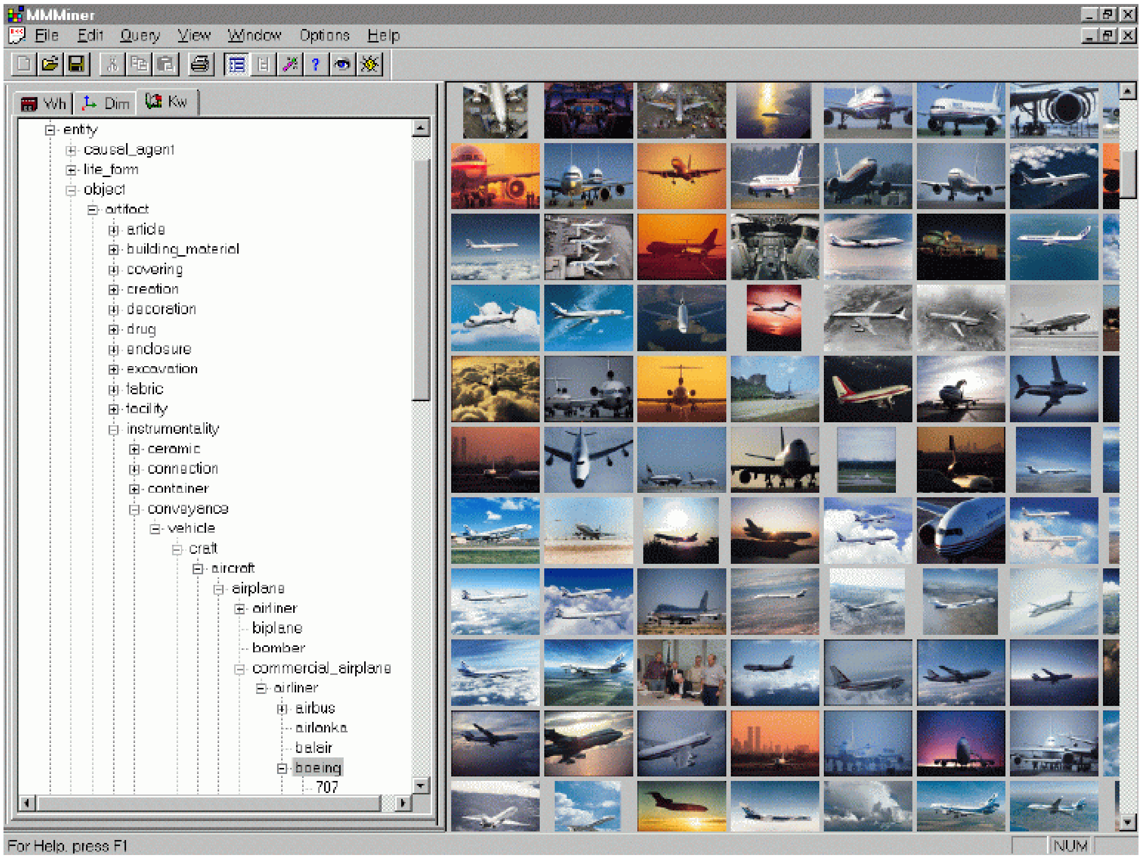


Figure 5.2: Selecting (and browsing) data sets of images using keyword hierarchy.

frequent colours for each of the 64 cells are stored in the colour layout vector and the number of edges for each orientation in each of the cells is stored in the edge layout vector. Other sizes of grids, like 4 × 4, 2 × 2 and 1 × 1, can be derived easily. These colour layout grids can be used for spatial relationships between colours at different levels of resolution.

The Image Excavator uses image contextual information, like HTML tags in web pages, file name and path, neighbouring text, etc., to derive keywords (see Chapter 4).

The hierarchy of keywords with its hypernymy and hyponymy relationships allows one to browse the image and video collection by topic. In Figure 5.2, for example, thumbnails of commercial airplanes pertaining to the aircraft manufacturer Boeing are displayed. This user interface also allows the selection of a multimedia data set to be mined. The hierarchy of keywords on the left of Figure 5.2 is a section of the concept hierarchy automatically generated by visiting some web sites containing aircraft images.

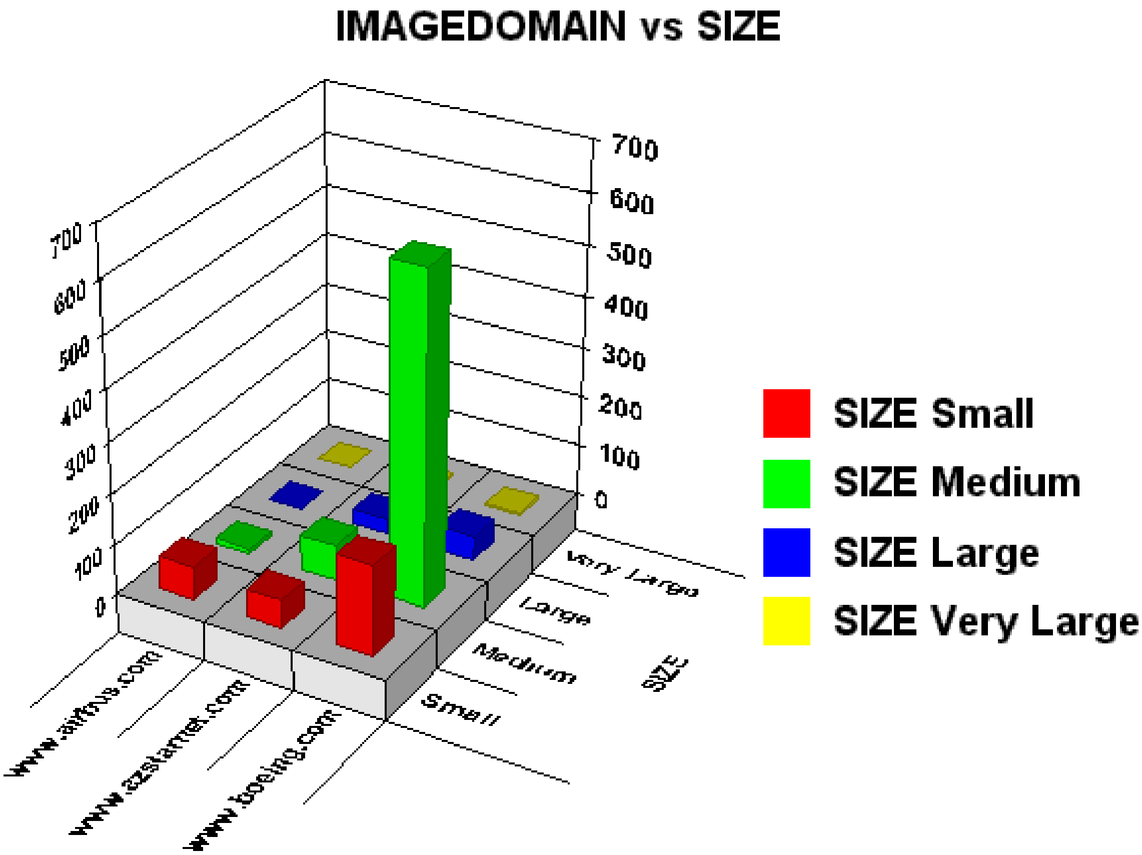


Figure 5.3: Snapshot of MultiMediaMiner Characterizer

The mining modules of the MultiMediaMiner system include three major functional modules, *characterizer, classifier*, and *associator*. Many data mining techniques are used in the development of these modules, including *data cube construction and search* [47], *attribute-oriented induction* [120], *mining multi-level association rules* [121], etc.

The functionalities of these modules are described as follows:

- MM-Characterizer: This module discovers a set of characteristic features at multiple abstraction levels from a relevant set of data in a multimedia database. It provides users with a multiple-level view of the data in the database with roll-up and drill-down capabilities. Figure 5.3 describes in a histogram graph the general characteristics for two dimensions: the size of the media in bytes and the Internet domain from which the media were extracted. For this example, only three Internet domains were considered, while the sizes were "rolled-up" to a higher concept of media size (i.e.

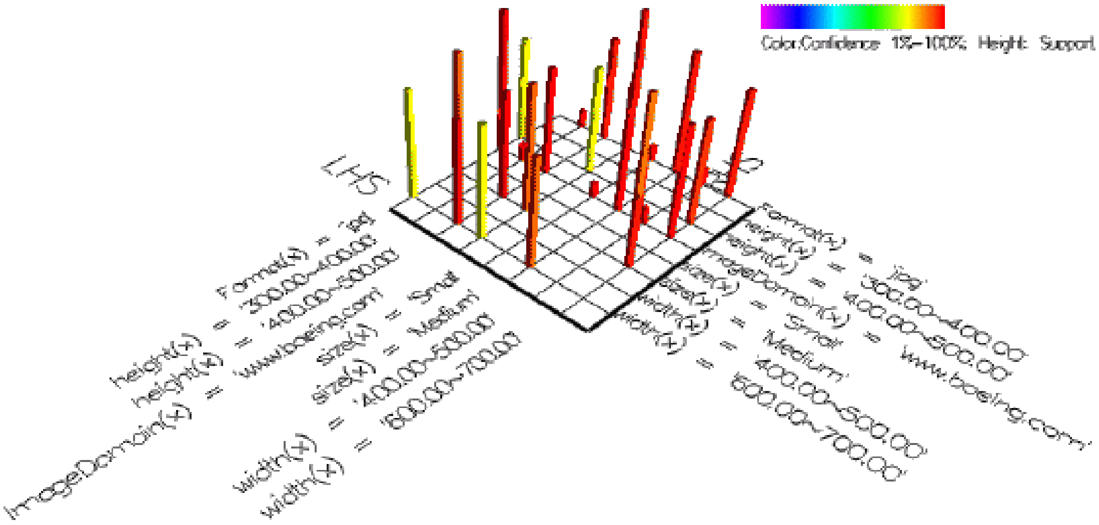


Figure 5.4: Visualization of association rules.

small, medium, large and very large). With this user interface, it is possible to visualize any two dimensions at a time, and drill-down or roll-up along a given dimension to find characteristics on more concrete values or specialized concepts.

For example, the module may describe the general characteristics of image sequences based on the topic of the video, the topic being a high level keyword defined in the concept hierarchy. The user can drill-down along the topic dimension to find characteristics of the image sequences based on more concrete topics.

- MM-Associator: This module finds a set of association rules from the relevant set(s) of data in an image and video database. An association rule shows the frequently occurring patterns (or relationships) of a set of data items in a database. A typical association rule is in the form of "$X \rightarrow Y[s\%, c\%]$" where X and Y are sets of predicates, $s\%$ is the support of the rule (the probability that X and Y hold together among all the possible cases), and c% is the confidence of the rule (the conditional probability that Y is true under the condition of X). For example, the module mines association rules like: "*what are relationships among still images, the frequent colours used in them, their size and the keyword '*sky*'?*" One possible association rule among many to be found is "*if image is big and is related to* sky*, it is* blue *with a possibility of 68%*" or "*if image is small and is related to* sky*, it is* dark blue *with a possibility*

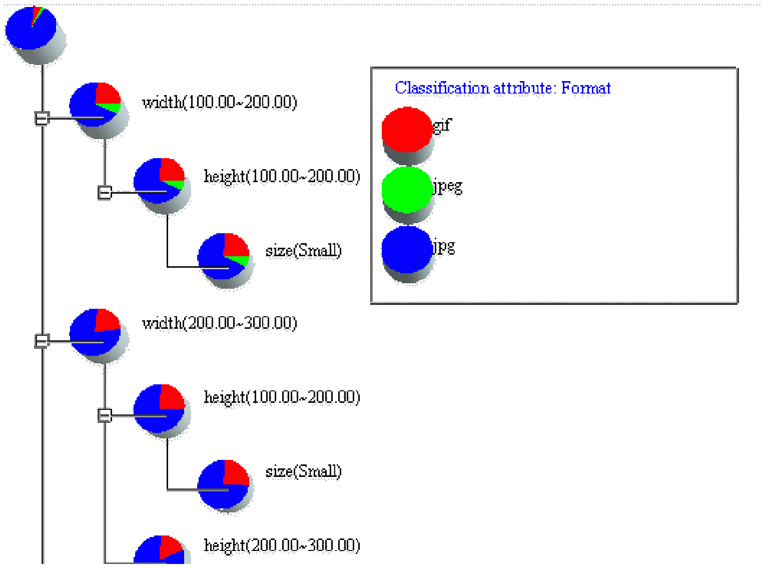
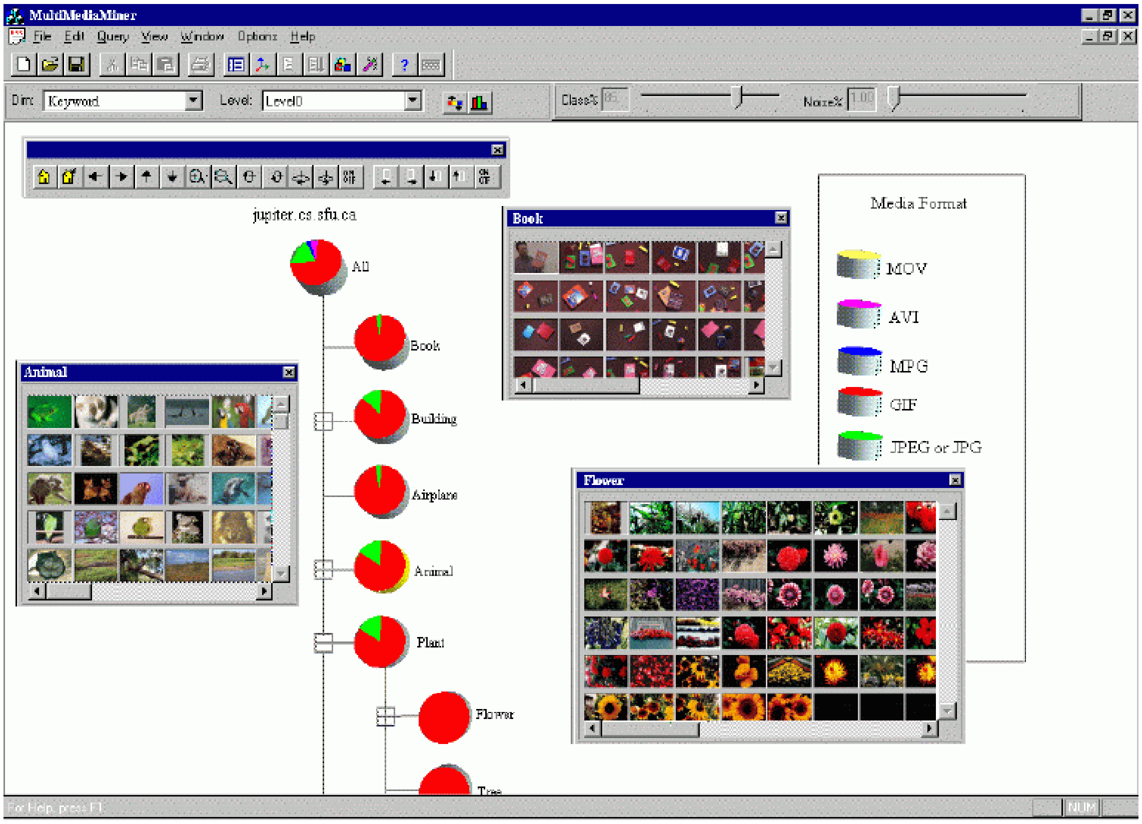


Figure 5.5: Excerpt from a classification tree generated by MultiMediaMiner

of 55%". Figure 5.4 shows a visualization of some association rules. The existence of a column on the grid represents an association between the left-hand side parameters and the right-hand side parameters. The height of the column depicts the support of the rule it represents, while the colour of the column describes the confidence of the rule.

- MM-Classifier: This module classifies multimedia data based on some provided class labels. The result is an elegant classification of a large set of multimedia data and a characteristic description of each class. This classification represented as a decision tree can also be used for prediction. Figure 5.5 shows an output of this module where a classification of images and frames based on their topic, with reference to the distribution of image format, is made for a given Web site. By clicking on a class, it is possible to drill through to the raw data. A window displays the images pertaining to the class (ex. book, animal, flower in Figure 5.6).

The user interface of all these modules allow drilling and rolling-up along the different concept hierarchies defined on the dimensions, and thus, allow interactive mining. It is also possible to drill through right to the raw data. In our case the raw data are images and videos stored on the Web. MultiMediaMiner calls a Web browser and displays the original image in its original size or even the web pages that contain the image. This gives an



Dim: Keyword

Level: Level0

Figure 5.6: MultiMediaMiner Classifier user interface with drill-through to the class images.

opportunity for information retrieval from the Web, based on the data mining results.

The MM-Characterizer, MM-Associator, and MM-Classifier are useful modules for visual asset management and indirect visual media retrieval. The additional OLAP capabilities attach strength to the interactive management and retrieval for multimedia repositories.

5.2 Obstacles and Challenges with Multimedia Mining

The first problem with mining multimedia databases is gaining access to significantly large multimedia data sets. This may seem trivial, but getting access to CT scans from hospitals, for instance, is not easy due to privacy issues. CT scans would have been an interesting application for the discovery of association rules based on colours in these scans. We chose the World-Wide Web as our image and video source because it is free, available and has a reasonably large collection. Another advantage of using the Web as our source for images and video is that we can use the context of the images to automatically extract additional information like the keywords from the pages containing the image, the popularity of the image (i.e. how many pages use the same image), the Internet domain of the image, etc. All this information was added to the already dimension-rich database. Moreover, by saving the URL of images, we avoid the need for large storage space for the images and videos. The World-Wide Web is used as the repository. This, however, requires regular validation due to the dynamic nature of the World-Wide Web. Indeed, some images may disappear and some new ones appear in the web pages already visited by our crawler. If images disappear from the Web, they are discarded from our database. If the images change, they are processed again and the descriptors in our database replaced while the changes are propagated to the data cube structure. In addition, by saving the URLs of the images and the URLs of the pages that contain the images, it is possible to do information retrieval and resource discovery from the Web by drilling through the results of the data mining process.

5.2.1 Keyword hierarchies

Keywords describing images are very important and useful when dealing with large collections of images. However, automatically associating keywords to images is not easy, while manual keywording is definitely not scalable. As mentioned in Section 5.1, we take advantage of the semi-structure of the web pages and the syntax of the URLs to extract candidate keywords that, after normalization and filtering, are associated to the images. The normalization process uses morphological analysis to draw forth the canonical forms of words, while the filtering process uses a list of stopwords and WordNet lexical database to eliminate illicit or unwanted words. While the candidate keyword selection and the keyword filtering eliminate most of the unwanted words, the list of keywords per image still remains large. This can be reduced by adding new stopwords and/or use natural language heuristics to eliminate outliers.

For On-Line Analytical Processing (OLAP), concept hierarchies are needed to drill-down and roll-up along the dimensions defined on the data. These hierarchies are also important for multi-level mining in order to specialize or generalize the knowledge discovered. Thus, organizing the keywords in a concept hierarchy is pertinent for multimedia mining. However, building a concept hierarchy of natural language words is difficult because of the controversies it may generate. We had to build an explicit representation of the set of keywords in the form of concept hierarchy that most people (users) would agree upon. The solution was

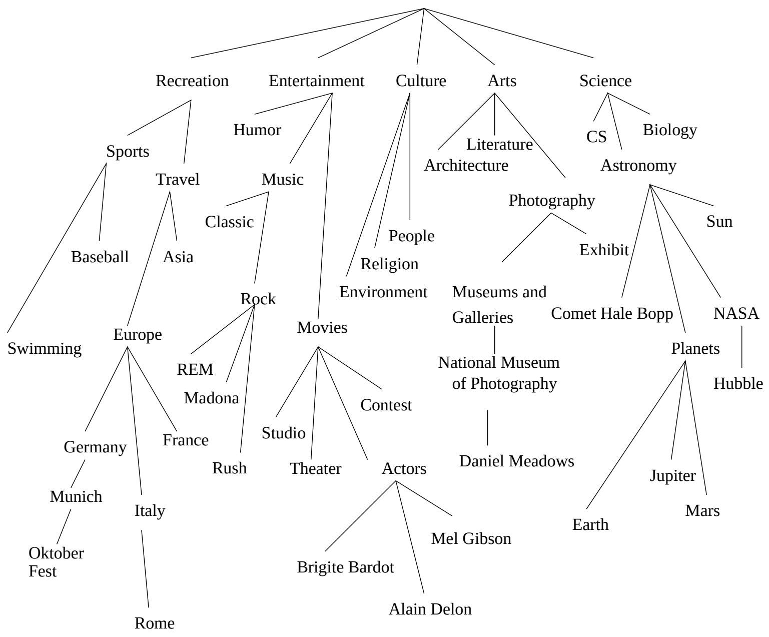


Figure 5.7: Portion of the keyword hierarchy generated by traversing the Yahoo directories.

to use existing word hierarchies that are widely and extensively used and accepted. Our first attempt was to automatically build a concept hierarchy by traversing a manually-built and widely-used on-line directory structure. By traversing the on-line Yahoo directory, for instance, one can build a general hierarchy with all nodes of the directories. Figure 5.7 shows a portion of the keyword hierarchy generated by traversing the Yahoo directories and mapping the directories to keywords. Unfortunately, this hierarchy tends to be too general and is not flexible enough to accommodate new terms. In other words, the hierarchy generated is shallow, narrow and not flexible.

Ultimately, we opted to use the on-line dictionary, thesaurus, and semantic network WordNet developed at the University of Princeton [263, 25] and used by many researchers in linguistics and cognitive science. WordNet version 1.6 contains approximately 95,600 different word forms organized into 71,100 word meanings interconnected with links representing subsumptions. Unfortunately, WordNet's word list does not contain specific words like "Boeing 747" or "fighter F15" that were extracted from the Web sites our crawler visited. After consulting the list of words rejected by the filtering process, some words were selected and added to enrich WordNet's semantic network with these new domain related terms.

Finally, the subsumption connections in the enhanced WordNet semantic network were used to build a concept hierarchy with all (and only) the keywords extracted and accepted from the web pages. This hierarchy is used to classify images by topic and browse the image and video collection. Figure 5.2 shows a portion of such hierarchy starting from the node "entity" of the enhanced WordNet network. The method for creating the concept hierarchy is presented in Algorithm 2.4.1 in Chapter 2. Figure 5.8 illustrates the use of WordNet for keyword filtering and word hierarchy building.

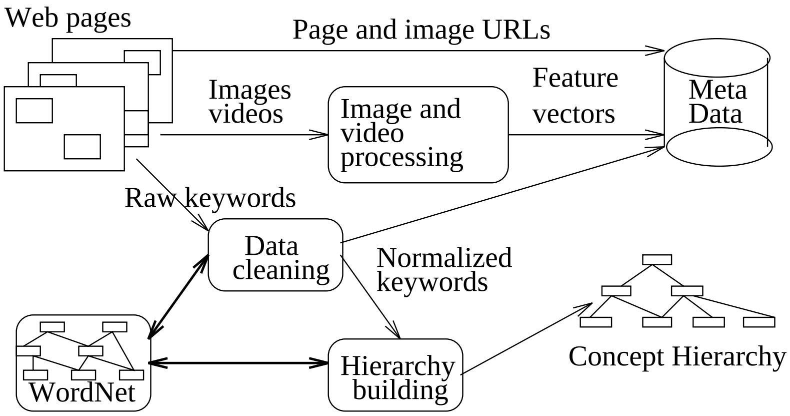


Figure 5.8: Keyword Normalization and Concept Hierarchy building using WordNet.

5.2.2 The curse of dimensionality

A data cube is a particular structure for storing multi-dimensional data and handling queries that aggregate over some of these dimensions at different levels of abstraction. This structure can be stored either in main memory or on disk.

The multimedia data cube we use has many dimensions. The following are some examples: (1) The size of the image or video in bytes with automatically generated numerical hierarchy. (2) The width and height of the frames (or picture) constitute 2 dimensions with automatically generated numerical hierarchy. (3) The date on which the image or video was created (or last modified) is another dimension on which a time hierarchy is built. (4) The format type of the image or video with two-level hierarchy containing all video and still image formats. (5) The frame sequence duration in seconds (0 seconds for still images)with numerical hierarchy. (6) The image or video Internet domain with a pre-defined domain hierarchy; each image or video collected has a unique URL (Unified Resource Locator) that indicates the location (Internet domain) where the image or video is stored. (7) The Internet domain of pages referencing the image or video (parent URL) with a pre-defined domain hierarchy; when an image or video is located in a web page, a reference to that page (parent URL) is stored with the image meta-data in our database. (8) The keywords with a term hierarchy defined as described above; (9) A colour dimension with a pre-defined colour hierarchy; colours are quantized and indexed in a range between 0 and 255. A colour hierarchy is defined from specific colours to more general colours. An image or a video is considered containing a given colour if the percentage of pixels in that colour exceeds a given threshold. (10) An edge-orientation dimension with a pre-defined hierarchy, etc. An image is considered containing a certain edge orientation if the percentage of edges in the orientation in the image exceeds a given threshold. (11) The popularity of an image or video with a numerical hierarchy. The popularity of an object is the known number of pages that reference that object. (12) The richness of a web page with a numerical hierarchy. The richness of a web page is the number of multimedia objects referenced in the page.

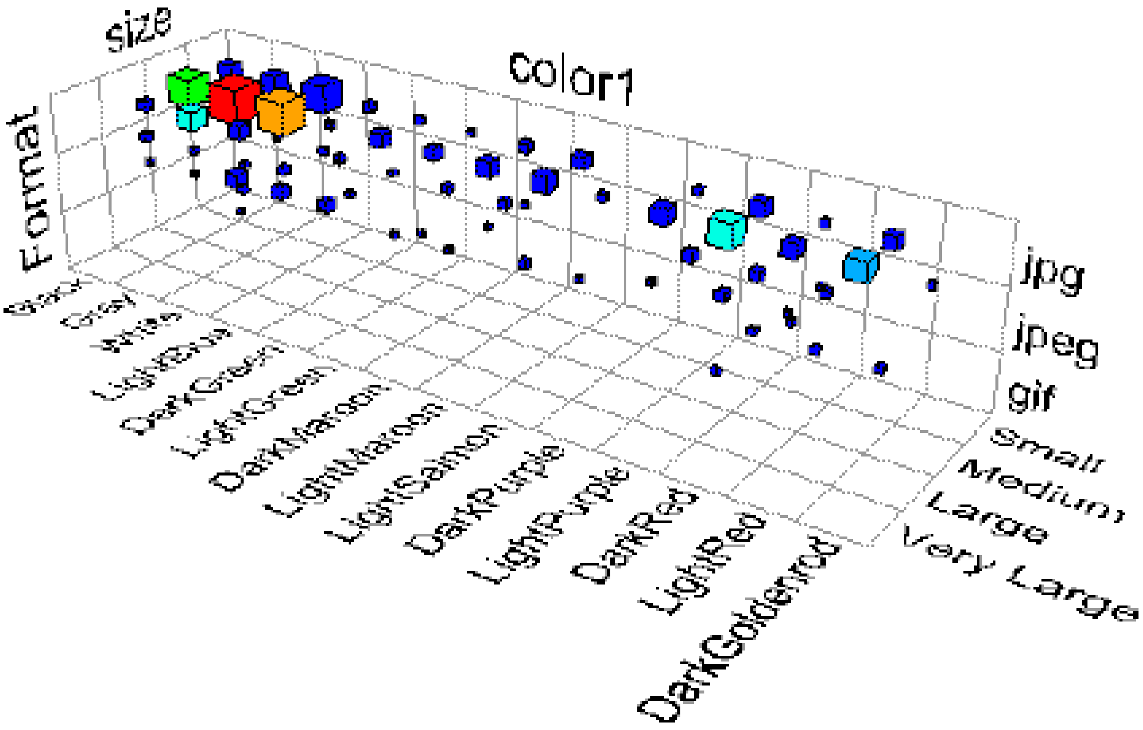
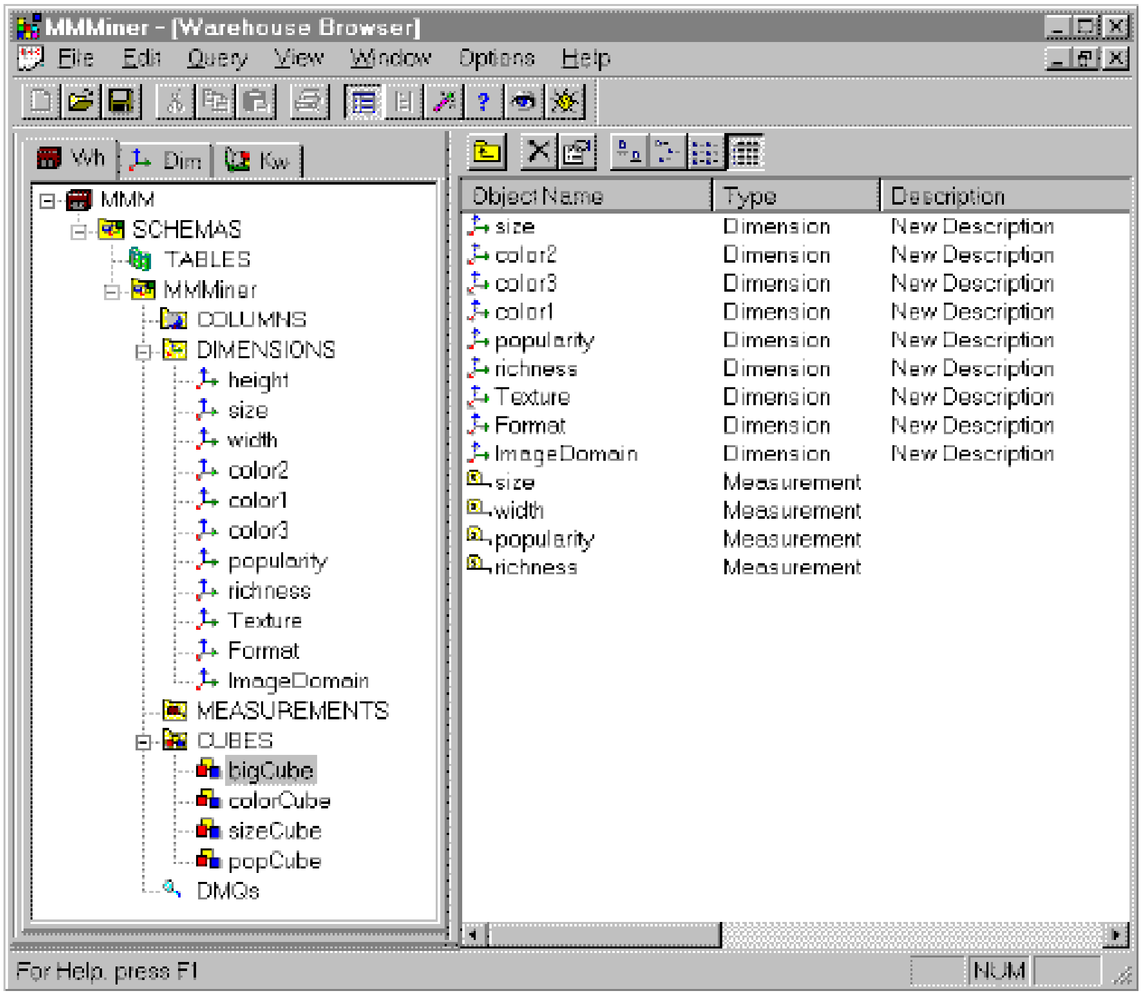


Figure 5.9: Browsing 3 dimensions of the multimedia data cube.

Using these different dimensions and their respective concept hierarchies, it is possible to build a multi-dimensional data cube that aggregates the values for all attributes in each dimension domain. Figure 5.9 shows a visualization tool used to browse such multi-dimensional data cubes, 3 dimensions at a time. The concept hierarchy defined on each dimension allows drilling-down and rolling-up along any given dimension. This type of data cube browsing gives a big picture of the content of the database and even allows one to see rough clustering of data values. Selecting a sub-cube from the view drills through it up to the raw data, and one can see the set of multimedia items in the selected sub-cube and even the web pages that contain them.

Unfortunately, it is very difficult, if not impossible, to have more than a given number of dimensions in a physical data cube. This is not due to the visualization or conceptualization as it may seem, but it is due to the fact that the size of the data cube grows exponentially with the number of dimensions. Each time a dimension is added, the size of the data cube is multiplied by the number of distinct values in the new dimension. This is the curse of dimensionality. In [214] Ross illustrates how the number of dimensions in a data cube is physically limited due to the physical size of the memory.

The colour attribute of an image has 256 dimensions, for instance. Each of the dimensions counts the frequency of a given colour in images. This already goes beyond the limit of most data cube-based systems. Even after quantizing the colours to 64 values, the number of dimensions is still too large for MultiMediaMiner to handle. In order to reduce the number of dimensions, we decided to collapse and pivot the 64 colour dimensions into one. One previous colour dimension represented a colour and the values were frequencies of that colour in an image. With the collapsed dimension, the values represented are colours (presence of colours), and the colour frequencies are discarded. This loss of information is a compromise to reduce to dimensionality. The same principal was applied for the dimensions of the attribute texture. This brings up yet another challenge: the problem of multi-valued attributes. The collapsed colour dimension represents all the colours, however, an image or a video frame has more than one colour. If all the colours of an image are represented in the same dimension, the aggregate values in the aggregated layers of the data cube become wrong and meaningless. To solve this problem, a colour dimension for each colour present in an image is needed. However, this contradicts the goal of reducing the dimensionality. In our implementation, we have chosen to represent only the three most frequent colours of an image with 3 colour dimensions. This reduces the colour representation from 256



Object Name	Type	Description
size	Dimension	New Description
color2	Dimension	New Description
color3	Dimension	New Description
color1	Dimension	New Description
popularity	Dimension	New Description
richness	Dimension	New Description
Texture	Dimension	New Description
Format	Dimension	New Description
ImageDomain	Dimension	New Description
size	Measurement	
width	Measurement	
popularity	Measurement	
richness	Measurement	

Figure 5.10: MultiMediaMiner data warehouse with cubes and dimensions.

dimensions to 3.

As might be expected, colour is not the only multi-valued dimension. An image has many textures, is described by many keywords, and can be present in many web pages. In other words, the dimension texture, the dimension keyword, and the dimensions related to the web page (page richness and parent page Internet domain) are all multi-valued. For our prototype implementation, we had to compromise by choosing to represent only the most frequent texture in an image, only the first parent web page of an image found by our crawler, and we chose not to represent the keywords in our data cube. Not only it is not significant to select only one keyword by image or video since the keywords can not be ranked effectively, but the keyword dimension has also a very large number of potential values formed from words and phrases. This would cause the size of the cube to rapidly exceed the physical available limit.

Despite the fact that keywords are not represented in our cube, we use the keywords as a data set selection attribute to select a set of images on which to build our data cube. Thus, the aggregate values in the data cube pertain to the multimedia objects that are associated with the keyword used for the selection. By doing so, the selected keyword can be appended as a predicate to any rule discovered by our data mining modules based on the constructed data cube. Figure 5.2 shows the selection process using the keyword concept hierarchy. This selection is used for browsing images and for data set selection for data cube construction. When a keyword is selected, all keywords subsumed by it are also selected. This allows generalization and specialization along the word hierarchy. Note that selecting the image sub-set by constraining the keywords (left of Figure 5.2) can be replaced by a content-based constraint such as the content-based retrieval provided by C-BIRD. In other words, the user interface in Figure 5.2 could be linked to C-BIRD in order to select images to mine.

Although we reduced the number of dimensions, the number is still large. For the implementation of the MultiMediaMiner prototype, we have chosen to create not one cube, but a set of different data cubes with different (overlapping) dimensions. Figure 5.10 shows the user interface of the MultiMediaMiner data warehouse with 4 data cubes and the dimensions and measurements defined in one of them. Separating the data cube into smaller ones is a limitation. This restriction brought up new challenges. It is not trivial to choose which dimension should be represented in which cube when we have our data materialized in separate cubes. It is important to mention that the OLAP interaction and the data mining algorithms operate on one given cube at a time. Thus, it is not possible to discover correlations, for example, between two dimensions in different data cubes. Moreover, merging rules discovered from two cubes that do not overlap, is not possible.

In [130] selective materialization of data cubes is proposed to select the appropriate cuboids for materialization rather than materializing all the views. This approach, using a lattice that expresses dependencies among views and contains cube materialization costs, is intended to optimize the data cube construction based on the needs dictated by the user queries. In our implementation, as mentioned above, we chose to materialize 4 cuboids and pre-compute them after the user selects a data set using the keyword hierarchy. The cubes are built on-the-fly, once the images are selected, and can easily be built in parallel. There are some heuristics regarding the selection of the dimensions in the different cuboids, some based on the access frequency and some based on the size of the dimensions themselves. We opted for a more semantic approach. The set of dimensions was divided into 3 sub-sets: a content-based dimension set (colour and texture), a size-based dimension set (size, width, height, etc.), and a resource-based dimension set (Internet domain, popularity, etc.). Each set was materialized in a different cuboid. In addition, a fourth cuboid was materialized with dimensions from the 3 dimension sets. In order to create an overlap between the cuboids, the Internet domain and the size dimensions were repeated in all 4 cuboids.

Each cell of a data cube can contain aggregate values (i.e. measurements) like a count, a sum, etc. Because measurements are not expensive in memory size, we decided to materialize numeric attributes (like size, richness, popularity, etc.) as measurements, rather than as cube dimensions, whenever the attribute is not selected as dimension effectively present in the cuboid. This allows the consideration of values of that attribute, however, without the possibility to drill-down or roll-up along the dimension it represents.

5.3 On-going work and Conclusions

In this chapter, we have discussed online analytical processing (OLAP) and descriptor-based data mining from a multimedia repository. We have designed and developed an interesting multimedia data mining system prototype, MultiMediaMiner, with the following features: (i) a multi-dimensional multimedia data cube, (ii) multiple data mining modules, including characterization (or summarization), association, and classification, and (iii) an interactive mining interface and display with Web information retrieval capabilities. Our preliminary experiments demonstrate that multimedia data mining may lead to interesting and fruitful knowledge discoveries in multimedia databases.

There are some major tasks calling for further research into the design and development of the MultiMediaMiner system.

A new model for data cube materialization is under study. In this model, called MDDB for Multi-Dimensional DataBase, we conceptualize the entire data cube in a database with a special-purpose structure. The structure contains all dimensions and the aggregation of interesting values in preparation for cube materialization. The structure is not a data cube per se, but the "definition" of the hypercube which helps speed up the materialization of

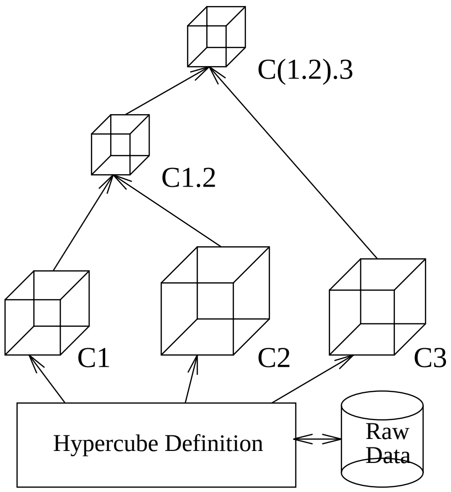


Figure 5.11: Multi-Dimensional Database model with materialization of cuboids.

cuboids. Cuboids are then materialized on-the-fly depending on the dimensions needed by the query. Moreover, borrowing from the multi-layered database technology presented in [276], a cuboid can generalize a set of cuboids along the hierarchies of its dimensions. A cuboid would join the dimensions of other cuboids at a higher conceptual level. This model allows the creation and manipulation of data cubes with an unrestricted number of dimensions, and allows multi-dimensional selection on raw data. Figure 5.11 shows the cuboid materialization path from a hypercube definition.

Multi-dimensional data cubes are created in order to reduce the response time when querying large databases for decision support or data mining. Typically, all the dimensions are aggregated in the cube. However, it is not always necessary to represent all the dimensions in the cube. Depending upon the application and the user needs, we can choose not to materialize some dimensions, and keep them as raw data in the database. For example, if colour is considered unnecessary for some applications, we can avoid materializing the colour dimensions and keep the colour information in the database. This obviously reduces the size of the data cube. However, if colour is required for some queries, we need to build on-the-fly a new data cube with colour dimensions directly from the raw data. This can be very costly. Another approach would be to adapt the data mining algorithms to use simultaneously the aggregations in the data cube and the raw data in the database without materializing the portion of the data that is still in the database. This is acceptable if the queries accessing the non materialized portions are scarce.

The design and construction of multimedia data cube can be improved by integrating the MDDB model or by using a virtual composite data cube that has some of its dimensions not materialized but in the database. The current design of the multimedia data cube, although it works, produces a huge multimedia data cube, due to the big size of two numeric dimensions: *colour* and *texture*. Most relational data cubes contain only categorical dimensions each having a relatively small number of distinct values. However, since we would like to support search from colour and edge-orientation, it is necessary for the data mining algorithms to have access to the data either materialized in a cube or directly from the database. Our current implementation supports only a limited number of intervals on these two dimensions in the data cube. The search along these dimensions with finer granularities than those currently supported has to access the C-BIRD database, which degrades the performance but can be improved by using the hypercube structure of the MDDB model.

There are plans to add new data mining functionalities into the system, like a clustering module which would group images into different clusters based on their multiple dimensional features, including both multimedia features, such as colour and edge-orientation, and relational features, such as keywords, URL information, and duration.

We have used the keyword hierarchy for browsing our image collection and selecting a data set for mining. In other words, the selection of images to mine is done based on keywords. We plan to use the content-based image retrieval features of C-BIRD to also select the images for mining.

No matter how much we seek, we never find anything but ourselves.

Anatole France

In rivers, the water you touch is the last of what has passed and the first that which comes: so with time present.

Leonardo da Vinci

Chapter 6

Content-Based Multimedia Data Mining

Discovering knowledge from large data has been the focus of many research studies and applications in the last few years. Many effective algorithms and successful applications have been suggested. However, most of these studies emphasised corporate data typically in alphanumeric databases. Very little research has been conducted on mining multimedia data. [236] describes the CONQUEST system that combines satellite data with geophisical data to discover patterns in global climat change. The SKICAT system described in [91] integrates techniques for image processing and data classification in order to identify "sky objects" (i.e. patterns) captured in a very large satellite picture set. Visual data are also the focal point of our research, and we integrate image processing with database mining techniques in order to discover frequent item-sets to ascertain content-based multimedia association rules. Current database mining technologies are still not capable of extracting knowledge from images and videos, although some researchers are starting to investigate how to determine interesting patterns in multimedia. Recently, Tucakov and Ng in [248] used a method for outlier detection to identify suspicious behaviour from videos taken by surveillance cameras.

What was presented heretofore in this thesis, was essentially online analytical processing (OLAP) and mining (OLAM) from a database containing visual data descriptors. While extracting and processing the descriptors for OLAP and OLAM is a challenging task, it can arguably be depicted as limited multimedia mining. Indeed, most of the descriptors are not content-based, as size, popularity, keywords, etc. The content-based descriptors, such as colour and texture, were taken at a high level. For example, the presence of a colour (most frequent colours) was taken into account, but not its position in the picture, its size within the picture, its movement in time, etc. Clearly, there are other content-based features that can be exploited in multimedia data mining such as in association rule discovery or classification.

In this Chapter, we undertake the task to enhance our data mining algorithms to take advantage of content-based features pre-processed and stored in the C-BIRD database, such as the MFC and MFO centroids, the layouts, the locales, etc. We extend the concept of content-based multimedia association rules using feature localization and introduce the concept of progressive refinement in the discovery of patterns in images from coarse to fine resolution. Our contribution in this Chapter is a progressive resolution refinement approach for the discovery of multimedia association rules with recurrent objects, and for the discovery of spatial relationships between visual descriptors in large image collections.

Feature localization is a new concept of rough segmentation that we introduced in Chapter 4. Image segmentation is a process to segment an entire image into disjoint regions. A region consists of a set of pixels that share certain properties, e.g., similar colour (or gray-level intensity), similar texture, etc. In short, the traditional segmentation algorithms assume (1) regions are mostly *connected*; (2) regions are *disjoint* ($R_i \cap R_j = \emptyset$, for $i \neq j$); and (3) segmentation is *complete* in that any pixel will be assigned to some region, and the union of all regions is the entire image ($\cup_{k=1}^{m} R_k = I$). Although regions do not have to be connected, most available region-based and/or edge-based segmentation methods would yield connected regions, and it is error-prone to merge some of them into non-connected regions. Such a segmentation algorithm will yield more than a dozen purple regions, one for each character, for the title of the book shown in Figure 6.1. It will also yield (unexpectedly) many white regions, since all the white blobs inside the letters 'A', 'P', 'R' 'O' will unfortunately be identified as regions. The above example, albeit simple and not at all unusual, indicates that the traditional image segmentation does not yield useful grouping and representation for object recognition. A more useful and attainable process is feature localization that will identify features by their locality and proximity. As we defined in [167], a *locale* $\mathcal{L}_x$ is a local enclosure (or locality) of feature x. $\mathcal{L}_x$ has an envelope L_x which is a set of tiles to represent the locality of $\mathcal{L}_x$, and some geometric parameters: *mass* $M(\mathcal{L}_x)$, *centroid* $C(\mathcal{L}_x)$, *variance* $\sigma^2(\mathcal{L}_x)$, and shape parameters for the locale, etc. A tile is a square area in an image. Its size is arbitrarily chosen as 16×16, but could be bigger or smaller. The tile is the building-unit for envelopes. A tile is 'red' if a sufficient number of pixels within the tile are red. It follows that a tile can be both 'red' and 'blue' if some of its pixels are red and some are blue. While a pixel is the unit for image segmentation, a tile is the unit for feature localization. Thus, feature localization is a kind of rough segmentation where overlap is possible and completeness is not necessary. The right columns of Figure 6.1 show an example of feature localization; each image illustrates a different locale. Perhaps the closest to this description a locale is the "blob" in the "blobworld" system from the University of California Berkley, presented in [116]. However, blobs have always an elliptic shape and there are only up to 10 representatives in an image. Blobs give just an approximate and vague representation of an image. We believe that locales depict better the content of an image in terms of dominant colours and textures.

Tiles are grouped into an envelope, if they are geometrically close. The closeness will be measured by variance to be defined below. $M(\mathcal{L}_x)$ is the number of pixels in L_x that actually have feature x. $M(\mathcal{L}_x)$ is usually less than the Area of L_x, although it could be equal to it. $C(\mathcal{L}_x)$ is simply the centroid of the mass. $\sigma^2(\mathcal{L}_x)$ is the variance of the Cartesian distance from pixels in L_x to the centroid, and it measures the eccentricity of $\mathcal{L}_x$. Note, M, C, σ^2, etc. are measured in unit of pixels, not in tiles. This guarantees the granularity. Hence the feature localization is not merely a low-resolution variation of image segmentation. We also define a minimum bounding circle around a locale to approximate the locale when evaluating topological relationships at different resolution levels.

The centroids of locales can help in discovering interesting spatial relationships within an image or between frames of a video clip. We are defining spatial primitives like *next_to, ontop_of* and *under* to describe relationships between colours or colour segments in an image. These primitives and colour layout grids extracted by the preprocessor can help discover association rules about colours within an image or a video clip. In the previous chapter we defined the notion of localization or locales [167] which are rough colour and texture segments in an image. We will study the option to use these locales, rather than all the colour and texture of an image, to describe the colour features of the image or objects within the image, since they are perceptually more accurate.

In Figure 6.2, we enumerate some of the locale characteristics and relationships that we would like to capture in association rules discovered in a multimedia database. Colour, texture, size and shape do not need explanation. The centroid of a locale determines its

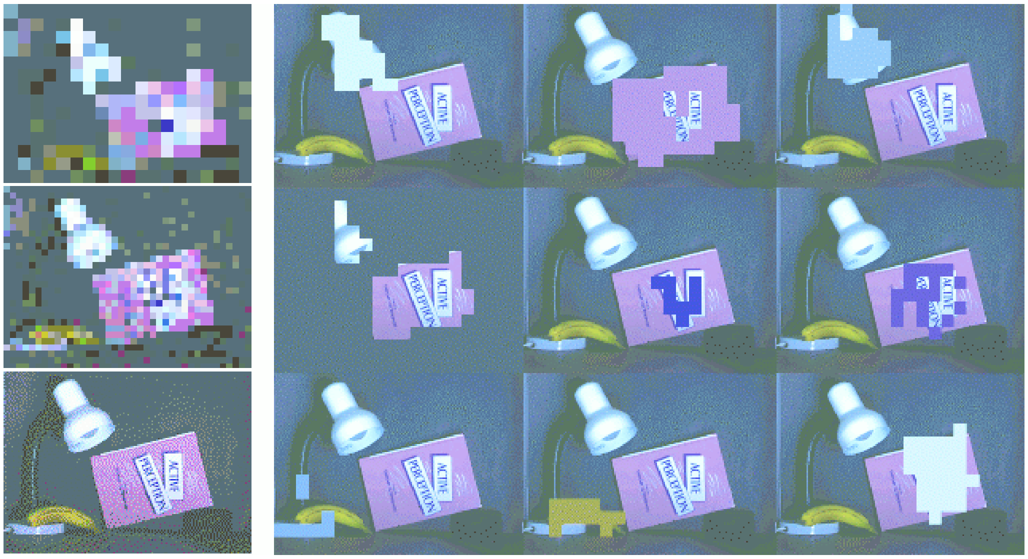


Figure 6.1: Example of feature localization based on colour for a multi-level resolution image.

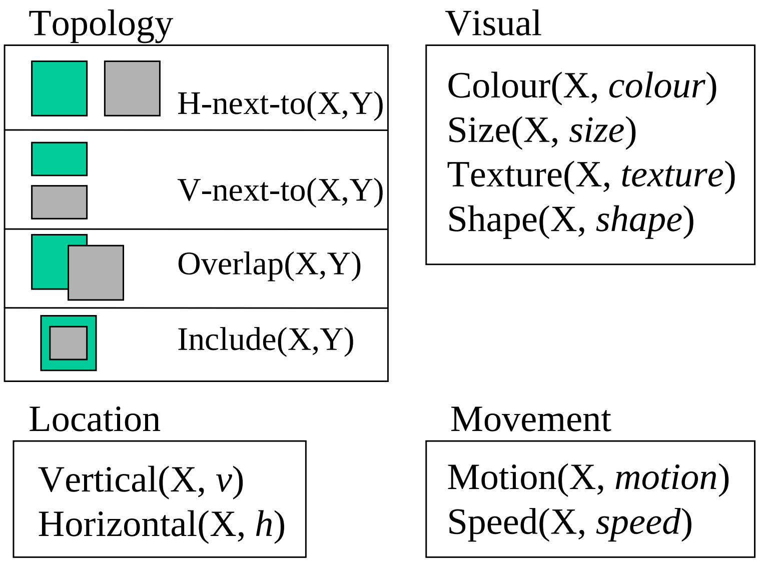


Figure 6.2: Feature Relationships for Locales.

position in a picture. *Vertical(X, v)* and *Horizontal(X,h)* can give this location. Note that this location can also be given by the layout grid that we use for colour layout and texture layout searches. This location would be relative to the granularity level of the grid: $8 \times 8, 4 \times 4, 2 \times 2$, or 1×1. Given the location (x, y) of a locale and its size, a topology, or spatial relationship with other locales can be determined, such as closeness with *H-next-to* and *V-next-to*, overlap with *Overlap*, and inclusion with *Include*. *H-nest-to* and *V-next-to* are generalizations of primitives such as *ontop-of, under, left-next-to, right-next-to*, etc. Again, the closeness of colours and textures can also be determined with the layout grids at different resolution levels. The vertical next-to and the horizontal next-to could be determined with the coordinates i, j of cells in the grid and their content (orientation and colour). To a certain extent, overlap and inclusion can also be determined with the layout grids, since each cell holds more than one colour at a time. We chose locales to illustrate the concept of topological closeness, overlap and inclusion in multimedia association rules without claiming that using locales is better than using the grid layout. We believe that the choice should be determined by the application domain.

In a video sequence, locales can be identified in different frames and their motion vector can be determined. In that case, a motion direction can be associated to the locale with *Motion(X, m)*, as well as a relative speed (for example pixels by frame) with *Speed(X, s)*.

In the subsequent sections, we will re-introduce the association rules and underline the limitations of the current algorithms for association rule discovery vis-à-vis multimedia data. We will present a coarse-to-fine strategy for mining multimedia and discuss two algorithms for the discovery of multimedia association rules with recurrent items and recurrent spatial relationships.

In our discussion, we will assume locales as being objects. This is an approximation that will simplify the discussion. Note that locales that always have a similar movement vector in an image can be merged into one object.

The remainder of the Chapter is laid out as following: In Section 1 we discuss the progressive resolution refinement approach and present our algorithm for mining multimedia association rules with recurrent items; we put forth a method for mining association rules with spatial relationships in Section 2; Section 3 describes our performance study; finally, our conclusions are presented in Section 4.

6.1 Multimedia Association Rules

Association rules have been extensively studied in the literature [6, 176, 8, 146, 121, 105, 150, 198, 184, 7, 94, 51, 52, 108, 53, 186, 24, 39, 14, 164, 206, 190, 196]. The efficient discovery of such rules has been a major focus in the data mining research community. Many algorithms and approaches have been proposed to deal with the discovery of different types of association rules discovered from a variety of databases. However, typically, the databases relied upon are alphanumerical and often transaction-based. While some of these algorithms proposed can be applied to visual data, to a certain extent, after transforming the data into a form that can be processed, new algorithms should be better suited. Indeed, visual data has some peculiarities proper to images and videos. For example, some visual features can be repeated in an image, and the repetition of the feature can carry more information than the existence of the feature itself.

The problem of discovering association rules is to find relationships between the existence of an object (or characteristic) and the existence of other objects (or characteristics) in a large repetitive collection. Such a repetitive collection can be a set of transactions for example, also known as the market basket. Typically, association rules are found from sets of transactions, each transaction being a different assortment of items, like in a shopping store ({milk, bread, etc}). Association rules would give the probability that some items appear with others based on the processed transactions, for example milk→bread[50%], meaning that there is a probability 0.5 that bread is bought when milk is bought. Essentially, the problem consists of finding items that frequently appear together, known as frequent or large item-sets.

Formally, as defined in [8], the problem is stated as follows: Let $\mathcal{I} = \{i_1, i_2, ...i_m\}$ be a set of literals, called items. Let $\mathcal{D}$ be a set of transactions, where each transaction T is a set of items such that $T \subseteq \mathcal{I}$. A unique identifier TID is given to each transaction. A transaction T is said to contain X, a set of items in $\mathcal{I}$, if $X \subseteq T$. An *association rule* is an implication of the form "$X \Rightarrow Y$", where $X \subseteq \mathcal{I}, Y \subseteq \mathcal{I}$, and $X \cap Y = \emptyset$. The rule $X \Rightarrow Y$ has a *support* s in the transaction set $\mathcal{D}$ is s% of the transactions in $\mathcal{D}$ contain $X \cup Y$. In other words, the support of the rule is the probability that X and Y hold together among all the possible presented cases. It is said that the rule $X \Rightarrow Y$ holds in the transaction set $\mathcal{D}$ with *confidence* c if c% of transactions in $\mathcal{D}$ that contain X also contain Y. In other words, the confidence of the rule is the conditional probability that the consequent Y is true under the condition of the antecedent X. The problem of discovering all association rules from a set of transactions $\mathcal{D}$ consists of generating the rules that have a *support* and *confidence* greater that given thresholds. These rules are called *strong rules.*

Looking at this formal definition, we immediately see limitations vis-à-vis mining association rules from an image and video collection. An image for instance, can indeed be represented by a transaction with items being the visual features in the image, however, items in the antecedent of the rule repeated in the consequent can be an interesting factor in image analysis applications. For example, in an infra-red satellite picture for weather forecast, the existence of a blue pocket (cold front) may suggest the existence another blue pocket. Moreover, the repetition of the same item in an image is not negligible. As mentioned previously, the repetition of a same object in an image can be more important than its occurrence in the image. Besides, recurrent objects in images are very frequent. In addition, one may be interested in finding associations with a coarse-to-fine search strategy. In other words, association rules can first be found at a low resolution, then progressively confirmed at higher resolutions. Indeed some visual features, such as dominant colours in an image, are well preserved at a low resolution level. Thus, we can rapidly approximate multimedia association rules at a coarse level, then eliminate false positives by verifying them at a higher resolution. Moreover, the approximation of a locale by a minimum bounding circle can speed up the discovery of association rules at a high conceptual level for spatial topological concepts such as closeness, overlap or containment. The precision of the rules discovered are improved by eliminating the minimum bounding circle and using the locale envelope with higher image resolutions. The main advantages of this approach is that: (1) locale features extraction can be conducted at multiple (often reduced) resolutions to save processing time; (2) locale intrinsic features can be defined at appropriate resolutions to avoid too much detail/noise or insufficient detail. Dominant colours are well preserved at a low resolution, but some texture information can be lost when the resolution is too low. The coarse-to-fine search strategy is important for large image and video databases even when the features are extracted and analyzed at pre-processing time. The left column of Figure 6.1 shows an example of image at different resolution levels.

The proposed algorithms for discovering association rules all assume the items are unique in $\mathcal{I}$, hence the definition of *support.* With the well-known *Apriori* algorithm [8], for example, duplicates are never considered when the k-item candidate sets C_k are formed. It is assumed that the items are unique. In multimedia mining, we would like to discover rules such as

"2*blue circles* $\Rightarrow$ *high texture density*". This means that the sole existence of blue does not necessarily imply the consequent *high texture density.* Two occurrences have to coexist in the image for the rule to be "confident". In addition, the definition of *strong rules* based on *large support* is quite inadequate in some imaging applications. Features appearing very frequently (i.e. having a large support) in some medical images, for example, can be normal phenomenon, and uninteresting to users. A low support, on the other hand, could generate item-sets with extremely rare items. While these rare items could either be just meaningless noise or sought for rare phenomenon (in medicine applications for example) they fall in the realm of outlier analysis[1] and are out of the scope of this study. We believe that a range for an acceptable support should be introduced for such applications. Hence the definition of the *sufficiently strong association rule.* We would also like to introduce a new definition of *support.* Typically, the support is the percentage of transactions that contain an item or verify a condition; it measures how interesting and frequent an item or predicate is in a data set. Since images are represented by transactions, but identical objects can be repeated in an image, our *support* could be a count of objects rather than a count of transactions. This, of course, should depend upon the application, and it is up to the user to choose the appropriate support definition. We call this support Object-based support while the "traditional" support is called transaction-based support. We also call Association Rules with Recurrent Items association rules that allow items to be repeated in the rules.

Definition 6.1.1 *An* **Association Rule with Recurrent Items** *is a rule of the form:*

$$\alpha P_1 \wedge \beta P_2 \wedge ... \wedge \gamma P_n \;\rightarrow\; \delta Q_1 \wedge \lambda Q_2 \wedge ... \wedge \mu Q_m (c\%)$$

where $c\%$ is the confidence of the rule, predicates $P_i, i \in [1..n]$ and $Q_j, j \in [1..m]$ are predicates bound to variables, and $\alpha, \beta, \gamma, \delta, \lambda$ and μ are integers. αP is true if and only if P has α occurrences. □

Definition 6.1.2 *A* **Multimedia Association Rule** *is an association rule with recurrent items that associates visual object features in images and video frames, and is of the form:*

$$\alpha P_1 \wedge \beta P_2 \wedge ... \wedge \gamma P_n \;\rightarrow\; \delta Q_1 \wedge \lambda Q_2 \wedge ... \wedge \mu Q_m (c\%)$$

[1] There are studies in data mining about outliers [147]

where $c\%$ is the confidence of the rule, one or more predicates $P_i, i \in [1..n]$ and $Q_j, j \in [1..m]$ are predicates instantiated to topological, visual, kinematics, or other descriptors of images, and $\alpha, \beta, \gamma, \delta, \lambda$ and μ are integers quantifying the occurrence of the object feature or item. αP is true if and only if P has α occurrences. □

The predicates P_i and Q_j in the rules are not just topological, visual or kinematics descriptors such those in Figure 6.2, but can also be other descriptors such as picture size, video duration, or just related keywords. In a medical imaging system, for example, the physician's diagnosis attached to the image can be extremely beneficial in an association rule.

Definition 6.1.3 *The* **Support** *of a predicate P in as set of images $\mathcal{D}$ denoted by $\sigma(P/\mathcal{D})$ is the percentage of objects in all images in $\mathcal{D}$ that verify P at a given conceptual level. The* **Confidence** *of a multimedia association rule $P \rightarrow Q$ is the ratio $\sigma(P \wedge Q/\mathcal{D})$ versus $\sigma(P/\mathcal{D})$, which is the probability that Q is verified by objects in images in $\mathcal{D}$ that verify P at the same conceptual level. Such support is called* **object-based support** *in contrast to* **transaction-based support***, which is the percentage of images having a given feature.* □

As mentioned earlier, depending upon the application, the definition of support can also be dependent on the number of images. In that case the support of a predicate is the percentage of images in which the predicates holds (transaction-based support). We define three thresholds that verify the adequate frequency of a pattern and the adequacy (or certainty) of a rule. To find *sufficiently frequent* image objects that verify a predicate P, in other words a frequent pattern P in $\mathcal{D}$, the support of P should be not greater that a *maximum support* Σ' and not smaller than a *minimum support* σ'. To find sufficiently strong multimedia association rules $P \rightarrow Q$, the following should be true: $\sigma' \leq \sigma(P \wedge Q/\mathcal{D}) \leq \Sigma'$ and the confidence of $P \rightarrow Q$ should be greater than a *minimum confidence* φ'. The minimum and maximum support are defined regardless of the type of support transaction-based or object-based.

Definition 6.1.4 *A pattern p is* **sufficiently frequent** *in a set $\mathcal{D}$ at a level ℓ if the support of p is no less than its corresponding minimum support threshold, and no more than its corresponding maximum support threshold.* □

Definition 6.1.5 *A multimedia association rule $P \rightarrow Q$ in a set of images $\mathcal{D}$ is* **sufficiently strong** *in $\mathcal{D}$ if P and Q are sufficiently frequent (P and $Q \in [\sigma\prime..\Sigma\prime]$) and the confidence of $P \rightarrow Q$ is greater than $\varphi\prime$.* □

Note that the *strength* of a rule and the values of $\sigma\prime$ and $\Sigma\prime$ depend upon the concept level in which the predicates are applied. All attributes such as colour, texture, motion direction, etc., are defined on concept hierarchies. Depending on the concept level selected by the user, $\sigma\prime$ and $\Sigma\prime$ can be higher or lower.

Given an image I as a transaction and locales $\mathcal{L}_i$ (or objects) as the items in the image I, we envision two types of multimedia association rules: association rules based only on atomic visual features that we call Content-Based Multimedia Association Rules with Recurrent Visual Descriptors, and association rules with spatial relationships that we call Multimedia Association Rules with Recurrent Spatial Relationships. What we call atomic features are descriptors such as colour, texture, etc. They are attributes of an object defined along concept hierarchies. Association rules based on atomic visual features are similar to multi-dimensional, multi-level association rules, emphasizing on the presence of values of some attributes at given concept levels. They are multi-dimensional because each object has different attributes, each being a dimension, and they are multi-level, since the values of each attribute are defined at different conceptual levels, for example the colour blue could be defined along this hierarchy: All_blue(dark_blue(NavyBlue , RoyalBlue, DeepSkyBlue), blue(LightSteelBlue, SlateBlue, SkyBlue, MediumTurquoise), light_blue(PaleTurquoise, LightCyan, Cyan)). One such association rule could be: $Dark\_Red\ circle \wedge Light\_Blue\ circle \rightarrow Green\ square(56\%)$. Note that we used only two dimensions in this example: colour and shape. Any other dimension or other descriptors such as image size or keyword could be used as well.

The second type of multimedia association rules uses the topological relationships between locales (v-next-to for vertical closeness, h-next-to for horizontal closeness, overlap, and include). Perhaps the closest to what we intend to work on for multimedia association rule enumeration with spatial relationships is the work presented in [150] about spatial association rules which uses primitives to describe spatial relationships between entities in maps. However, there is a fundamental difference between our approaches. In [150], it is assumed that a grouping is done based on a data set selection, with a spatial mining language GMQL. The associations found are on the grouping, making spatial predicates holding only one argument. In other words, the spatial association rules are restricted and describe only one type of objects at a time, example road or water. In our case, we have two-argument predicates and objects are not typed. Each predicate P describes the relationship between two objects O_a and O_b, such as *Overlap*(O_a, O_b), each object being multi-dimensional. Binary predicates involve a join of more than one relation. Moreover, spatial predicates on the same object values can be recurrent. One such multimedia association rule with spatial relationships could be: *V-Next-to*([red, circle, small], [blue, square, $*$]) $\wedge$ *H-Next-to*([red, circle, $*$],[yellow, $*$, large]) $\rightarrow$ *Overlap*([red, circle, $*$],[green, $*$, $*$]) (34%). Note that not all dimensions of the locales are used. The maximum dimensionality would be specified by the user. In this example, only three dimensions were needed and we made use of the wildcard $*$ to replace absent values.

6.1.1 Progressive Resolution Refinement

For effective and efficient discovery of patterns in multimedia databases, we chose a multi-resolution strategy by first finding patterns at a low (i.e. rough) resolution and persevering the search at a higher (i.e. finer) resolution with only the data selected in lower resolutions. This assumes the preservation of the patterns to be discovered in coarse resolutions. Recently, some researchers started to employ multi-resolution image representations in their content-based retrieval. We have adopted the same approach when matching object models in images and videos [169, 167] (see Chapter 4). In his earlier work, Burt [43] proposes a structure of *pattern tree* for active sensing. The structure describes objects in different levels of details in a hierarchy with multi-resolutions. A coarse-to-fine search strategy is adopted to actively and rapidly locate objects or events in a scene. Smith [232] uses wavelets for multiscale image representation in the Alexandria Digital Library project. Lately, Koperski in [151] proposes a progressive refinement approach for spatial data mining using two steps to filter out large data sets.

The basic idea of progressive refinement is to quickly approximate patterns at a coarse level, then eliminate false positives by verifying them at a higher resolution. The refinement, however, has to be done carefully without inadvertently eliminating false negatives. For instance, by knowing how visual features are preserved in coarse resolutions, some visual features can be tested at low resolution such as colours, others like edge density could be tested at an intermediate level, while fine texture should only be tested at a high resolution. Spatial relationships are not completely preserved. The topological characteristics are not fully retained, making the the topological features change from one resolution level to the other. In Section 6.2.1 we discuss the preservation and the potential changes of topological features when the image resolution is altered or improved. The refinement of the image resolution can be done in many ways. We distinguish three different refinements: (i) a cleansing at the pixel level (raster refinement). This refinement has many resolution levels; (ii) an approximation with minimum bounding circles. This refinement has only two resolution levels; and (iii) a zooming by changing the size of local tiles (tile shrinking). This refinement has five or more levels, with tile sizes $32 \times 32, 16 \times 16, 8 \times 8, 4 \times 4$, and 2×2.

The following is the general algorithm of the progressive resolution refinement for multimedia data mining.

Algorithm 6.1.1 (PRR) Progressive Resolution Refinement for Mining Multimedia Association Rules in Image Collections.

Input: (i) $\mathcal{D}$ a set of transactions representing images at different resolution levels, with items being the visual and non visual descriptors of the images; (ii) a set of concept hierarchies for each attribute; (iii) the minimum and maximum support thresholds σ' and Σ' for each conceptual level; (iv) the maximum number of resolution level available.

Output: Sufficiently frequent item-sets with recurrent items at different resolution levels R_i.

Method. The progressive resolution refinement mining of multimedia association rules proceeds as follows:

```
begin
    $i \leftarrow 0$ /* Lowest resolution level */
    $\mathcal{D}_0 \leftarrow \mathcal{D}$
    while ($i <$ maximum resolution level) do { /* Coarse to fine discovery */
        $R_i \leftarrow \{r \mid r$ is a sufficiently frequent item-set at resolution level $i$ $(in\mathcal{D}_i)\}$
        $i \leftarrow i + 1$ /* Move to higher resolution level */
        $\mathcal{D}_i \leftarrow Filter(\mathcal{D}_{i-1}, R_{i-1})$
    }
end
```

□

The algorithm is a loop with two considerable steps: (a) finding frequent item-sets at a given resolution level; (b) Reducing the size of the data set by filtering out images and infrequent objects to prepare the next round at a higher resolution. The move from one level to another does not have to be one at a time (Line 5). It is sometimes desirable to skip some resolution levels and jump to a higher one. Note that depending upon the application and the user's needs, it is not always necessary to do all the resolution levels and iterate to the highest resolution (Line 3). Line 4 calls the algorithm for enumerating frequent item-sets with recurrent items at a given resolution level. This can either be for frequent visual features or for frequent spatial relationships. We will discuss in the coming subsections the discovery procedure for these two types of association rules. $Filter(\mathcal{D}_{i-1}, R_{i-1})$ in line 6 removes images that do not contain the frequent item-sets discovered at the resolution level $i-1$ and filters out the infrequent objects in the remaining images. This reduces the set of images and visual features to be processed at higher resolution. The filtering, however, does not consider the re-occurrence of items since the low resolution can affect the numbering of visual features. Figure 6.3(a), for example, shows one blue locale at a coarse level that becomes clearly two distinct blue locales at a finer resolution. This shows that only the presence and absence of a feature should be considered in the filtering process, and not the frequency of appearance of the features in the image. Figure 6.3(a) also illustrates an example depicting the relativity of some spatial relationships, like overlap, based on the resolution used for defining locales. While two locales may appear overlapping because their minimum bounding circles intersect, considered at the locale envelop level, they do not. Moreover, reducing the size of the tile's edge form 16 × 16, as in our experiments, progressively down to pixel by pixel, another level of coarse-to-fine refinement can be performed.

We will discuss in the following subsection the algorithm for enumerating sufficiently frequent item-sets with recurrent items.

6.1.2 Generating Synthetic Images

We believe that data mining from images content is effective only when the collection of images considered for mining is homogeneous; meaning that the content of images in the collection should have analogous semantic content. If the content of the images is not comparable, the content mining becomes meaningless. For example, mining the content of a conglomerate of images randomly collected from the Internet, like flowers, people, boats, etc., would lead to unavailing and useless results. For these types of images, a mining based

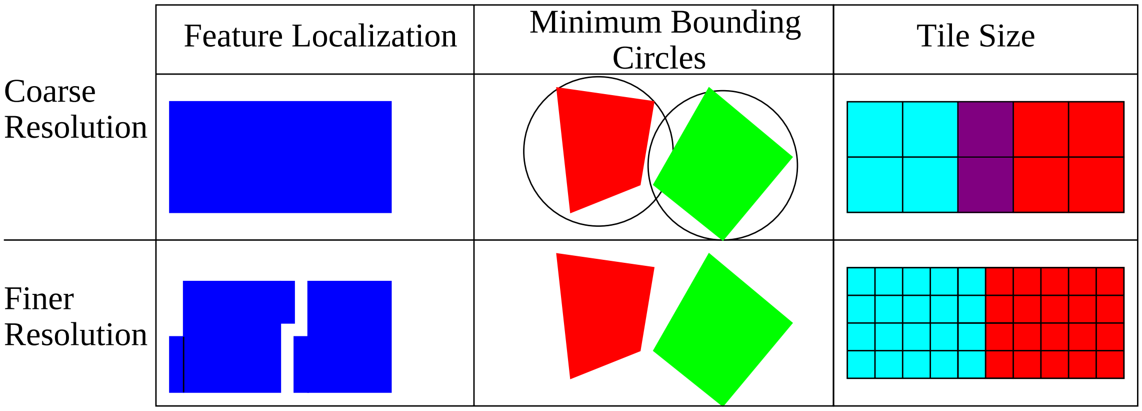


Figure 6.3: Relativity of visual feature concepts at different resolution levels.

on external descriptors and minimal content, such as the mining presented in Section 5.1, is more useful. A repository of infra-red satellite pictures, a collection of brain CT scans, a set of frames from a fixed surveillance camera, are good examples of homogeneous images collections where content-based multimedia data mining can be effective.

To illustrate the algorithms and test their performance, we have generated synthetic images with random locales and random features. All the images are generated the same way to get a homogeneous collection. We generated the synthetic images as follows: (i) we generated n images each with a random background colour; (ii) for each image, we generated a random number k of locales; (iii) for each locale, we randomly generated features (colour, mass (i.e. number of pixels), texture, shape, position, etc.); For each image, given the generated locales, we randomly gave movement directions to each locale, by generating a random number of frames with random new positions.

Algorithm 6.1.2 Generating synthetic images.

Input: n the number of frame sets.

Output: N images where $N = \sum_1^n (1 + random(m))$

Method. Algorithm for generating synthetic images is as follows:

```
begin
(1)     get $n$;   /* number of original images */
(2)     for $i = 1$ to $n$ do {
```

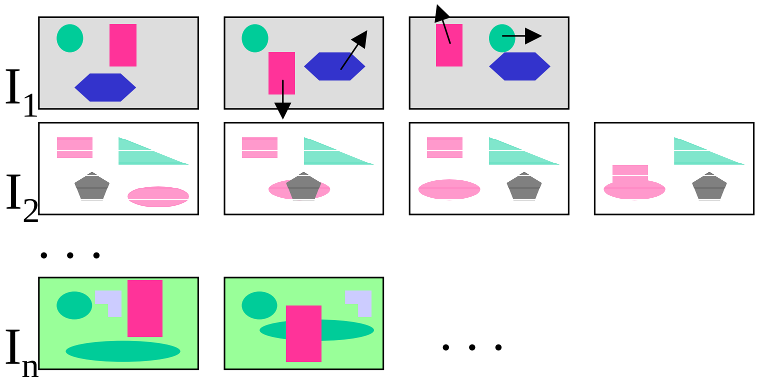


Figure 6.4: Synthetic images.

```
        generate background colour for $\mathcal{I}_i$;
        generate $k$;   /* number of objects in image i */
        for $j = 1$ to $k$ do {
            generate (position, colour, shape, size, texture,...) for $\mathcal{L}_{(i,j)}$ in $\mathcal{I}_i$;
        }
        generate $m$;   /* number of frames associated to image i */
        for $f = 2$ to $m + 1$ do {
            copy image $\mathcal{I}_{i_{(f-1)}}$ into $\mathcal{I}_{i_f}$
            for $j = 1$ to $k$ do {
                generate(newposition) for $\mathcal{L}_{(i^f,j)}$ in $\mathcal{I}_{i^f}$;
            }
        }
    }
end                                                                    □
```

The result of Algorithm 6.1.2 is an assortment of n sets of frames, each with a different number (m) of images. Each image has a certain number (k) of different objects. Images in different frame sets may have a different number of objects. Figure 6.4 shows a portion of a synthetic image set with n images (in rows), each with a random number of objects with random features, and each image replicated in a random number of frames (in columns) with the objects in different position. The first raw example shows the motion vectors of moving objects. Algorithm 6.1.2 fills Table 6.1 with the generated objects descriptors and movement vectors, one row for each image frame. This table is used for discovering

Content-based multimedia association rules with recurrent items. The Naïve algorithm and MaxOccur algorithm, presented later, mine the information in this table. Given the size and the positions of the objects, the extended-relation in Table 6.2 is filled with spatial relationships attributed to each object in individual images. This second table along with the previous table, is exploited to discover multimedia association rules with recurrent spatial association rules. The algorithm is presented later in this Chapter.

Image ID	Object ID	Colour	Texture	Mass	Shape	Motion	...
I_1	$O_{(1,1)}$	Colour$_1$	Texture$_1$	Size$_1$	Shape$_1$	Direction$_1$	...
I_1	$O_{(1,2)}$	...	...	...	...	...	...
...							
I_2	$O_{(2,1)}$	...	...	...	...	...	...
...							
I_n	$O_{(n,\alpha)}$	...					

Table 6.1: Relation with Visual Atomic Features.

Image ID	Object ID	V-Next-to	H-Next-to	Overlap	Include
I_1	$O_{(1,1)}$	$\{O_{(1,3)}, O_{(1,5)}\}$	$\{O_{(1,2)}, O_{(1,6)}\}$	$\{O_{(1,7)}\}$	$\{\}$
I_1	$O_{(1,2)}$	$\{...\}$	$\{...\}$	$\{...\}$	$\{...\}$
I_n	$O_{(n,\alpha)}$	...			

Table 6.2: Extended-Relation with Spatial Relationships.

6.1.3 Naïve Approach for Finding Frequent Itemsets with Recurrent Items at a Given Resolution Level

If the Apriori algorithm [8] is to be used to discover frequent item-sets in such data sets as the image collections, it would miss all item-sets with recurrent items. A naïve way to find all frequent item-sets with recurrent items would be to first find all frequent 1-item-sets, check how often they might re-occur in an image (maximum occurrence), and then, for each k, combine these frequent 1-item-sets in sets of k elements where elements can be repeated up to their respective maximum occurrence possible. The calculation of the support would filter out the infrequent ones. The pseudo-code of such algorithm is as follows:

```
begin
(1)   $C_1 \leftarrow \{$Candidate 1 item-sets and their $support\}$
(2)   $F_1 \leftarrow \{Sufficiently\ frequent$ 1 item-sets and their $support\}$
(3)   $M \leftarrow \{Maximum\ occurrence$ in an image of $frequent$ 1 item-sets$\}$
(4)   Count # of k-item-sets (total[1..$k$])
(5)   for $(i \leftarrow 2; F_{i-1} \neq \emptyset; i \leftarrow i+1)$ $do\{$
(6)       $C_i \leftarrow \{c = \{x_1, x_2, ...x_i\} \mid \forall \alpha \in [1..i] x_\alpha \in F_1 \wedge (M[x_\alpha] \geq CARD\ of\ x_\alpha\ in\ c)\}$
(7)       $C_i \leftarrow C_i - \{c \mid (i-1)$ item-set of $c \notin F_{i-1}\}$
(8)       $\mathcal{D}_i \leftarrow FilterTable(\mathcal{D}_{i-1},\ F_{i-1})$
(9)       foreach image $I$ in $\mathcal{D}_i$ do {
(10)          foreach $c$ in $C_i$ do {
(11)              $c.support \leftarrow c.support + \ Count(c,\ I)$
(12)          }
(13)      }
(14)      $F_i \leftarrow \{c \in C_i \mid \frac{c.support}{total\ i\ itemset} > \sigma\prime\}$
(15)  }
(16)  Result $\leftarrow \bigcup_i \{c \in F_i \mid i > 1 \wedge c.support < \Sigma\prime\}$
end
```

This naïve algorithm, which guarantees to find all frequent item-sets with recurrent items, could be improved by replacing F_1 as the starting set for enumerating candidates of all k-item-sets by a set composed of F_1 and all item-sets with single items twinned to their maximum capacity, such as $\{x_\alpha\}, \{x_\alpha, x_\alpha\}, \{x_\alpha, x_\alpha, x_\alpha\}$, etc., where the number of x_α is smaller or equal to $M[x_\alpha]$. This would improve the processing of C_i in line 6.

In the next sub-section we present our algorithm MaxOccur, a more efficient algorithm for discovering multimedia association rules with recurrent items. The performance of this naïve algorithm and our MaxOccur algorithm are compared in Section 6.3.

6.1.4 Max-Occur Algorithm

A method for enumerating sufficiently strong multimedia association rules that are based on recurrent atomic visual features is presented in this section. We will give an abstract example and then present the algorithm. To simplify our discussion, we will use a one dimension, one level problem where images are transactions of objects and the same objects can be repeated. While objects are multi-dimensional, in this discussion we will treat them as items with only one dimension and no concept hierarchy. The algorithm can be extrapolated to the multi-level association rules discovery algorithm presented in [106]. The multi-dimensional issue can also be solved by using a data cube [286].

Example 6.1.1 Let us consider the images represented in Table 6.3(left) by a set of transactions $\mathcal{D}_1$. Each image is a set of objects that can be repeated. At this point, we ignore the descriptors of the objects for simplicity. To determine the support of each object, a first scan of the database is done and each time a distinct object appears, its counter is incremented. At the same time, a second counter keeps track of the maximum appearances of the same object in an image (i.e. transaction). Table 6.3(right) shows the result of the counting. C_1 contains all unique objects with their support and M contains the maximum number of times a given object occurs in an image. To simplify the discussion, since the total number of images and object occurrences are fixed, the support of the objects is expressed in an absolute value (number of occurrences) rather than a relative percentage. Let the minimum support $\sigma\prime$ be 2 and the maximum support $\Sigma\prime$ be 5. To derive the sufficiently frequent 1 item-sets, if C_1 is filtered and only the objects that have a support between $\sigma\prime$ and $\Sigma\prime$ are kept, very frequent objects ($\sigma > \Sigma\prime$) would be eliminated. While this may seem the natural thing to do, it is counter-productive at this stage. Indeed, item-sets that are not frequent enough should be eliminated, since combining infrequent objects with other objects would be bound to generate infrequent item-sets (apriori property [8]). However, very frequent item-sets that are greater than the maximum support, when combined with other objects may generate item-sets that are less frequent than the maximum support but still frequent enough to be interesting. Thus, too frequent item-sets should not be eliminated until all frequent item-sets are found. Table 6.4(left) shows F_1, the list of frequent 1 item-sets. Notice that O_2 and O_4 were not eliminated even if they appear too often in the data set ($\sigma(O_2/\mathcal{D}_1)$ and $\sigma(O_4/\mathcal{D}_1) > \Sigma\prime$). However, O_1, O_5, O_6 and O_7 were eliminated because they do not appear frequently enough ($\sigma(O_1/\mathcal{D}_1), \sigma(O_5/\mathcal{D}_1), \sigma(O_6/\mathcal{D}_1)$ and $\sigma(O_7/\mathcal{D}_1) < \sigma\prime$). Given F_1, we can filter out from $\mathcal{D}_1$ all irrelevant objects, and all transactions that do not contain frequent objects present in F_1. This would considerably reduce the time for scanning the data set in search for k-item-sets. Table 6.4(right) shows $\mathcal{D}_2$, the image transactions with only the interesting objects. The generation of the candidate 2 item-sets is done by joining F_1 with itself to create all possible pairs with frequent objects. It is similar to the *apriori-gen* in [8] except that the information stored in M, regarding replication of objects in images, is used to generate new pairs of the same objects that occur in a transaction more than once. The 2 item-sets $\{O_2, O_2\}$ and $\{O_4, O_4\}$ in Table 6.5(left) are produced that way. Notice that when filtering C_2 to generate $F2$ (Table 6.5), $\{O_2, O_4\}$ was not eliminated despite the fact that its support $\sigma(\{O_2, O_4\}/\mathcal{D}_2)$ is higher than the maximum support; this is for the same reason O_2 and O_4 were not eliminated when generating F_1. The candidate 3 item-set list C_3 is produced by joining F_2 elements and eliminating 3 item-sets that contain 2 item-sets not recognized as frequent (i.e. not in F_2). The counters in M are also used to generate item-sets such as $\{O_2, O_2, O_2\}$ in Table 6.6(left). After filtering the infrequent 3 item-sets, F_3 is produced. The candidate 4 item-sets is produced the same way by joining the frequent 3-item sets and pruning the unnecessary ones. For instance $\{O_2, O_2, O_3, O_4\}$ and $\{O_2, O_3, O_4, O_4\}$ are eliminated since, respectively, $\{O_2, O_2, O_3\}$ and $\{O_3, O_4, O_4\}$ are not in F_3. Finally, since no 5 item-set can be induced, the result is all F_i without their item-sets that have a support higher than the maximum support Σ'. The following are the sufficiently frequent item-sets:

$$\boxed{\begin{array}{l}\{O_2, O_2, O_4, O_4\},\ \{O_2, O_3, O_4\},\ \{O_2, O_2, O_4\},\\ \{O_2, O_4, O_4\},\ \{O_2, O_3\},\ \{O_3, O_4\},\ \{O_2, O_2\},\ \{O_4, O_4\}\end{array}}$$

Given these sufficiently frequent item-sets, sufficiently strong association rules could be found by generating all rules from a k-item-set of the form "(k-p) item-set → p item-set" with 0 < k < p and such that the confidence of the rule is higher than a given confidence threshold. With a confidence threshold set to 100%, only these rules are induced:

(1) $\{O_4, O_4\} \rightarrow \{O_2, O_2\}[100\%]$
(2) $\{O_2, O_4, O_4\} \rightarrow \{O_2\}[100\%]$
(3) $\{O_3, O_4\} \rightarrow \{O_2\}[100\%]$
(4) $\{O_3\} \rightarrow \{O_2, O_4\}[100\%]$
(5) $\{O_2, O_2\} \rightarrow \{O_4\}[100\%]$
(6) $\{O_4, O_4\} \rightarrow \{O_2\}[100\%]$
(7) $\{O_3\} \rightarrow \{O_2\}[100\%]$
(8) $\{O_3\} \rightarrow \{O_4\}[100\%]$

A simple scan of these rules can count replicated objects (or similar objects depending upon the conceptual level and the dimensions used) and produce the following rules:

$2\ O_4 \rightarrow 2\ O_2$ [100%], $O_2 \wedge\ 2\ O_4 \rightarrow O_2$ [100%], $O_3 \wedge O_4 \rightarrow O_2$ [100%], $O_3 \rightarrow O_2 \wedge O_4$ [100%], $2\ O_2 \rightarrow O_4$ [100%], $2\ O_4 \rightarrow O_2$ [100%], $O_3 \rightarrow O_2$ [100%], and $O_3 \rightarrow O_4$ [100%].

Notice that the rule “$O_4 \rightarrow O_2$” is not confident enough, while “2 $O_4 \rightarrow 2\ O_2$” or “2 $O_4 \rightarrow O_2$” are 100% reliable. This would not have been true had the support been based on the number of images rather than on the number of objects.

Image ID	Objects
I_1	$\{O_2, O_2, O_2, O_4, O_5\}$
I_2	$\{O_2, O_2, O_4, O_4\}$
I_3	$\{O_2, O_3, O_4\}$
I_4	$\{O_6, O_7\}$
I_5	$\{O_1, O_2, O_2, O_3, O_4, O_4\}$

Object	Support	Max. Occurrence
$\{O_1\}$	1	1
$\{O_2\}$	8	3
$\{O_3\}$	2	1
$\{O_4\}$	6	2
$\{O_5\}$	1	1
$\{O_6\}$	1	1
$\{O_7\}$	1	1

Table 6.3: Left: Image transaction table $\mathcal{D}_1$. Right: C_1 and M tables.

Object	Support	Max. Occurrence
$\{O_2\}$	8	3
$\{O_3\}$	2	1
$\{O_4\}$	6	2

Image ID	Sufficiently Frequent Objects
I_1	$\{O_2, O_2, O_2, O_4\}$
I_2	$\{O_2, O_2, O_4, O_4\}$
I_3	$\{O_2, O_3, O_4\}$
I_4	$\{O_2, O_2, O_3, O_4, O_4\}$

Table 6.4: Left: F_1 and M tables. Right: Filtered image transaction table $\mathcal{D}_2$.

2 item-sets	Support
$\{O_2, O_3\}$	2
$\{O_2, O_4\}$	6
$\{O_3, O_4\}$	2
$\{O_2, O_2\}$	3
$\{O_4, O_4\}$	2

2 item-sets	Support
$\{O_2, O_3\}$	2
$\{O_2, O_4\}$	6
$\{O_3, O_4\}$	2
$\{O_2, O_2\}$	3
$\{O_4, O_4\}$	2

Table 6.5: Candidate 2 item-sets C_2 and sufficiently frequent 2 item-sets F_2.

The above example and discussion proceed to the following algorithm for mining content-based multimedia association rules. Note that the supports used in the example are absolute values for the sake of simplicity. Support for a k-item-set should be $\frac{\text{Count k−item−set in } \mathcal{D}_\text{k}}{\sum_{\forall \text{ transaction t}} \binom{|\text{t}|}{\text{k}}}$,

3 item-sets	Support
$\{O_2, O_3, O_4\}$	2
$\{O_2, O_2, O_3\}$	1
$\{O_2, O_2, O_4\}$	3
$\{O_2, O_4, O_4\}$	2
$\{O_3, O_4, O_4\}$	1
$\{O_2, O_2, O_2\}$	1

3 item-sets	Support
$\{O_2, O_3, O_4\}$	2
$\{O_2, O_2, O_4\}$	3
$\{O_2, O_4, O_4\}$	2

Table 6.6: Candidate 3 item-sets C_3 and sufficiently frequent 3 item-sets F_3.

4 item-sets	Support
$\{O_2, O_2, O_4, O_4\}$	2

4 item-sets	Support
$\{O_2, O_2, O_4, O_4\}$	2

Table 6.7: Candidate 4 item-sets C_4 and sufficiently frequent 4 item-sets F_4.

where $\binom{|t|}{k}$ are k-combinations of objects in transaction t without redundancy of unique objects. Algorithm 6.1.4 gives a glimpse into the combination enumeration process. A recursive algorithm could also be implemented.

Algorithm 6.1.3 (**MaxOccur**) Find sufficiently frequent item-sets for enumerating content-based multimedia association rules in image collections.

Input: (i) $\mathcal{D}_1$ a set of transactions representing images, with items being the visual and non visual descriptors of the images; (ii) a set of concept hierarchies for each attribute; (iii) the minimum and maximum support thresholds $\sigma\prime$ and $\Sigma\prime$ for each conceptual level.

Output: Sufficiently frequent item-sets with repetitions allowed.

Method. The pseudo-code for generating sufficiently frequent item-sets is as follows:

```
begin
    $C_1 \leftarrow \{$Candidate 1 item-sets and their $support\}$
    $F_1 \leftarrow \{Sufficiently\ frequent$ 1 item-sets and their $support\}$
    $M \leftarrow \{Maximum\ occurrence$ in an image of $frequent$ 1 item-sets$\}$
    Count # of k-item-sets (total[1..$k$])
    for $(i \leftarrow 2; F_{i-1} \neq \emptyset; i \leftarrow i+1)$ $do\{$
(6)        $C_i \leftarrow (F_{i-1} \bowtie F_{i-1}) \cup \{y \oplus X \mid X \in F_{i-1} \wedge y \in F_1 \wedge Count(y, X) < (M[y] - 1)\}$
(7)        $C_i \leftarrow C_i - \{c \mid (i-1)$ item-set of $c \notin F_{i-1}\}$
(8)        $\mathcal{D}_i \leftarrow FilterTable(\mathcal{D}_{i-1},\ F_{i-1})$
(9)        foreach image $I$ in $\mathcal{D}_i$ do {
(10)           foreach $c$ in $C_i$ do {
(11)               $c.support \leftarrow c.support + Count(c,\ I)$
(12)           }
(13)       }
(14)       $F_i \leftarrow \{c \in C_i \mid \frac{c.support}{total\ i\ itemset} > \sigma\prime\}$
(15)   }
(16)   Result $\leftarrow \bigcup_i \{c \in F_i \mid i > 1 \wedge c.support < \Sigma\prime\}$
end                                                                                    □
```

Line 1, 2, 3 and 4 are done doing the same initial scan. M contains the maximum number of times an object appears in the same image. This counter is used later to generate potential k-item-sets. The total number of k-item-sets is used for the calculation of the item-set support in line 14.

In line 6 and 7, the candidate item-sets are generated by joining (i-1) frequent item-sets and the use of M to generate repetitive objects ($M[y] > 1$). The pruning process (line 7) eliminates infrequent item-sets based on the *apriori* property.

Line 8 filters the transactions in $\mathcal{D}$ to minimize the data set scanning time.

In line 14, only the frequent item-sets that are higher than the minimum support $\sigma\prime$ are kept. It is only at the end of the loop (line 16) that maximum support $\Sigma\prime$ is used to eliminate item-sets that appear too frequently.

The calculation of the support for one item-set is based on the occurrence of the item-set in the images. Line 11 cumulates this count. A particular precaution has to be taken when counting appearances of k-item-set in an image, especially that objects and features can be repeated. A simple k-permutation ($\mathcal{C}_n^k = \frac{n!}{n!(n-k)!}$ where $n = \mid t \mid$) can lead to miscalculations. For example, let the transaction t be composed of repeated four objects such as $t = \{\diamondsuit\spadesuit\spadesuit\spadesuit\heartsuit\heartsuit\clubsuit\clubsuit\clubsuit\clubsuit\}$. $\mathcal{C}_{10}^2 = 45$ while we have only 9 possible unique 2-item-sets as shown below. There are also 14 possible 3-item-sets while $\mathcal{C}_{10}^3 = 240$.

{♢}	1
{♠}	3
{♡}	2
{♣}	4

{♢♠}	1	{♢♡}	1
{♢♣}	1	{♠♡}	2
{♠♣}	3	{♡♣}	2
{♠♠}	2	{♣♣}	2
{♡♡}	2		

{♢♠♡}	1	{♢♠♣}	1
{♢♡♣}	1	{♠♡♣}	2
{♠♠♠}	1	{♣♣♣}	1
{♢♠♠}	1	{♢♡♡}	1
{♢♣♣}	1	{♠♠♡}	1
{♠♠♣}	1	{♠♣♣}	1
{♡♣♣}	2	{♡♡♣}	2

Possible one, two and three item-sets and their occurrences in t.

The correct calculation of the repetitions of these item-sets in the transaction requires caution in order not to calculate occurrences more than necessary. The algorithm for enumerating the k-item-sets and counting their occurrences in the images transaction is given in Algorithm 6.1.4.

Algorithm 6.1.4 Counting occurrences of k-item-sets in an image transaction.

Input: (i) Image transaction $\mathcal{I}$; (ii) item-set size k.

Output: Set of k-item-sets and number of times they appear in $\mathcal{I}$.

Method. Generate all combinations from the unique objects in $\mathcal{I}$ and verify if they can be replicated (*Combination and Replication*); Generate item-sets with k times the same objects (*Twinning*); Generate item-sets with combinations of repeated objects (*Combination of twinned objects*). The pseudo-code for generating and counting the item-sets is as follows:

```
begin
    $U \leftarrow \{$unique 1-item-sets and their count in $\mathcal{I}\}$
    $C \leftarrow \{$k-combinations of $u$ in $U\}$
    foreach $c$ in $C$ do {   /* counting combinations and replications */
        $c.count \leftarrow 1$
        do $CountReplication(c)$
    }
    $V \leftarrow U$
    foreach $u$ in $V$ do {   /* Twinning */
        while $V[u].count > k$ do {
                $c \leftarrow \otimes_k u$
                $V[u].count \leftarrow V[u].count - k$
                Add $c$ in $C$ if not in set; $c.count \leftarrow c.count + 1$
            }
}
foreach $u$ in $U$ do {   /* Combination of twinned objects */
        for($n = 2$; $n < k - 1 \wedge n \leq U[u].count$; $n++$) do {
                $d \leftarrow \otimes_n u$
                $B \leftarrow$ {k-combinations of $d$ and $d' \mid v \in d' \wedge v \neq u \wedge U[v].count > 0$}
                foreach $c$ in $B$ do {
                        $c.count \leftarrow 1$
                        Add $c$ to $C$
                        do $CountReplication(c)$
                }
        }
}
Result $\leftarrow C$
end
begin $CountReplication(c)$
        $V \leftarrow U$
        foreach 1-item-set $i$ in $c$ do {$V[i].count \leftarrow V[i].count - 1$}
        while $V[j].count > 1 (\forall j$ in $c)$ do {
                $c.count \leftarrow c.count + 1$
                foreach 1-item-set $i$ in $c$ do {$V[i].count \leftarrow V[i].count - 1$}
        }
end
```

□

6.2 Multimedia Association Rules with Spatial Relationships

While the previously presented content-based multimedia association rules exclusively use visual atomic features such as in Table 6.1, multimedia association rules with spatial relationships in addition use the extended relation with spatial predicates such as in Table 6.2. A method for mining multimedia association rules with spatial relationships is introduced in this section. The method uses MaxOccur after minimizing predicates. Since spatial predicates (next-to, overlap, etc.) have two arguments, the strategy is to find frequent one and two-item-sets, combine the spatial predicates with only these frequent item-sets and consider the result as the candidate 1-item-sets of the multimedia association rules with spatial relationship. MaxOccur is then used to find the k-item-sets of spatial predicates. This strategy is based on the following property: for a spatial predicate $P(X, Y)$ to be sufficiently frequent, X and Y have to be sufficiently frequent, and the 2-item-set $\{X, Y\}$ has to be sufficiently frequent. This can be done at any conceptual level, starting from the highest concept in the hierarchy to the lowest ones. The naïve method would be to combine all pairs of object attributes at a given conceptual level and join them with all spatial predicates to derive potential 1-item-sets. This, however, would generate a very large number of candidates and even candidates that do not exist in the data set. Our modus operandi is to lessen the candidate set to the minimum before computing the frequent spatial predicate k-item-sets. To simplify the discussion, we will analyze an abstract example with one conceptual level and one dimension (shape) as follows:

Example 6.2.1 Considering the three images in Figure 6.5 with one dimensional objects, we would like to find association rules involving the spatial relationships between the objects in the images. For simplicity, we are only considering the dimension shape at a given conceptual level, but the same can be applied for other dimensions such colour, texture, etc. with related concept hierarchies. Finding sufficiently strong association rules with spatial relationships essentially consists of finding the sufficiently frequent conjunctions of spatial predicates. To do so, given the transaction-based minimum support threshold $\sigma\prime = 3$, a first scan of the image set reveals only three frequent items: ○, △ and ▭, each occurring in the three images and appearing at maximum twice in an image. Considering only these three frequent items, a second scan of the data set reveals the frequent pairs of items. The first table in Tables 6.8 indicates the support of each of these pairs. Only three of them are frequent enough and are coupled with the spatial predicates. Notice that if we added a wildcard $*$ to the frequent items with a de facto support equal to $\sigma\prime$, we could combine it wiith the frequent pairs of items, and thus generate association rules with wildcard attributes. Since we only have four spatial predicates (H-next-to, V-next-to, overlap, and include), this gives us up to 12 possibilities. However, a scan of the data set would reveal that only 7 combinations are possible, and at the same time, would also compute their support and

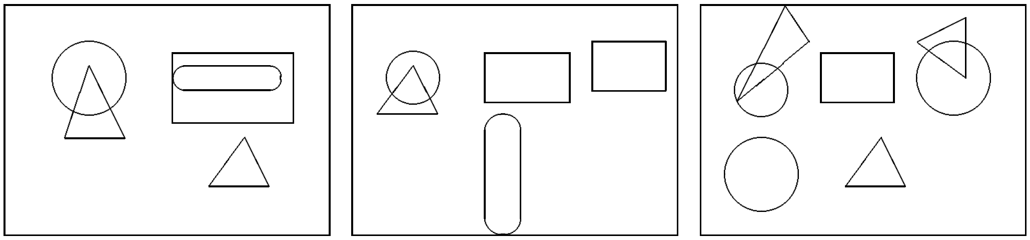


Figure 6.5: Examples of images with objects.

maximum occurrence in an image. The second table in Tables 6.8 shows the result of this scan, which is the set of frequent 1-item-set found in the first step of the MaxOccur. MaxOccur can then be used to discover the following frequent k-item-sets: Overlap($\bigcirc, \triangle$), H-Next-to($\bigcirc, \triangle$); Overlap($\bigcirc, \triangle$), H-Next-to($\bigcirc$,▭); H-Next-to($\bigcirc, \triangle$), H-Next-to($\bigcirc$,▭); Overlap($\bigcirc, \triangle$), H-Next-to($\bigcirc, \triangle$), H-Next-to($\bigcirc$,▭), and all the derived association rules such as: H-Next-to($\bigcirc$,▭) $\wedge$ H-Next-to($\triangle$,▭) $\rightarrow$ Overlap($\bigcirc, \triangle$) [100%]

Pairs of Objects	Frequency
{$\bigcirc, \bigcirc$}	1
{$\bigcirc, \triangle$}	3
{$\bigcirc$,▭}	3
{$\triangle, \triangle$}	2
{$\triangle$,▭}	3
{▭,▭}	1

1-item-set	Frequency	Max Occurrence
Overlap($\bigcirc, \triangle$)	3	2
H-Next-to($\bigcirc, \triangle$)	1	1
H-Next-to($\bigcirc$,▭)	3	2
H-Next-to($\triangle$,▭)	3	2
H-Next-to(▭,▭)	1	1
V-Next-to($\bigcirc, \triangle$)	1	1
V-Next-to($\triangle$,▭)	2	1

Table 6.8: Frequent pairs of objects and Frequent spatial predicates.

The above example and discussion proceed to the following algorithm for mining multimedia association rules with spatial relationships.

Algorithm 6.2.1 (**MM-Spatial**) Find sufficiently frequent item-sets for enumerating multimedia association rules with spatial relationships in image collections.

Input: (i) $\mathcal{D}_1$ a set of image descriptors with spatial relationships being the visual and non visual descriptors of the images; (ii) a set of concept hierarchies for each attribute; (iii) the minimum and maximum support thresholds $\sigma\prime$ and $\Sigma\prime$ for each conceptual level.

Output: Sufficiently frequent spatial predicate conjunctions.

Method. The pseudo-code for generating sufficiently frequent item-sets with spatial relationships is as follows:

```
begin
    $P_1 \leftarrow \{$Frequent atomic items$\}$
    $P_2 \leftarrow \{$Frequent pairs in $P_1 \times P_1\}$
    $C_1 \leftarrow \{P_2 \times$ {Spatial predicate set} and their *support*$\}$
    $F_1 \leftarrow \{$*Frequent* 1 item-sets from $C_1\}$
    line 3 to line 16 of **MaxOccur**
end
```

□

In the process of discovering multimedia association rules with recurrent spatial relationships, we have assumed the existence of enumerated spatial relationships such as in Table 6.2. These relationships are simply processed by comparing the centroid of each locale as well as the radius of the locale's shape approximated to a circle (minimum bounding circle). The centroids and the radii of locales are sufficient to rapidly and efficiently give a good approximation of spatial relationships between objects in an image such as closeness, overlap and inclusion. There exist other methods for determining more precise spatial relationships. However, these methods to be effective can be computationally costly. The coarse-to-fine strategy of the PRR algorithm simplifies the process by de facto eliminating in each round the images and objects not leading to interesting rules. Ideally, we would preprocess once the detailed spatial relationships at a fine granularity and lower granularity, and have a table such as Table 6.2 provided to the mining module. If this computation is not preprocessed before the discovery of association rules, another step could be added to the loop of PRR (Algorithm 6.1.1) to determine rough spatial relationships at the current resolution level and discover association rules with these approximate spatial relationships; then, the next rounds would refine the spatial relationships for only the frequent item-sets discovered. Notice that removing the minimum bounding circles at any resolution level like in Figure 6.3(a), assists at removing false positives from enumerated frequent spatial relationships. We will discuss in the next subsection topological changes from rough to fine resolution.

6.2.1 Topological Relationships with Resolution Refinement

We have introduced in Figure 6.2 some simple spatial relationships: closeness (vertical and horizontal), overlap, and containment. While these relationships have some level of detail like for vertical and horizontal closeness, they are still kept simple to make the discussion on multimedia association rules with spatial relationship understandable. Spatial relationships are essential components in query languages for geographic information systems and spatial databases, and describe topology of areas or regions in maps. In [75, 76] Max Egenhofer presents a formal derivation for eight spatial relationships namely disjoint, inside, contains, equals, meets, covered by, covers, and overlap. The relationships are formulated for areas based on intersections of the boundary of an area A denoted ∂A, the interior of the area denoted A° , and the exterior of the area denoted A^-. The intersections boundary/boundary, boundary/interior, boundary/exterior, interior/interior, interior/exterior, and exterior/exterior of two areas are characterized by the value empty ($\emptyset$) or non-empty ($\neg\emptyset$). For two areas A and B we can distinguish $2^9 = 512$ different relationships by combining the boundaries, interiors and exteriors (i.e. 9 combinations and 2 values). Most of these combinations however, are not valid combinations and only 8 valid relationships are derived. Because two of the relationships are symmetric, namely cover and covered by, inside and contains, some view these relationships as 6 distinct ones: disjoint, in, touch, equal, cover, and overlap [117]. In our discussion, we will use the eight relationships as described by Egenhofer, but we will use only boundary (∂A) and interior(A°) to define them since the boundary and interior suffice to distinguish between the different spatial relationships in our case. Table 6.9 shows the $2^4 = 16$ combinations among which 8 are valid.

The idea of progressive resolution refinement is to progressively reduce the size of the data set to be analyzed. Spatial relationships are defined for a given resolution level. If the spatial relationships are to be analyzed at the highest resolution level, the analysis could be very costly if the data set is large. However, analyzing the same data set at a rough resolution level can effortlessly yield some preliminary results. This preliminary result at a rough resolution level can be used to filter the large data set and obtain a smaller data set to be analyzed at a higher resolution. Doing the process recursively until reaching the finest resolution level available is more efficient than analyzing the whole large data set directly at the highest resolution level (see Algorithm 6.1.1). However, for the process to be

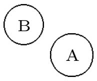
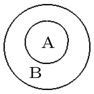
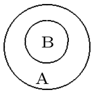
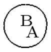
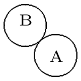
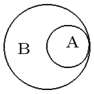
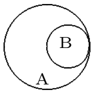
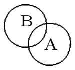
$A^\circ \cap B^\circ$	$\partial A \cap \partial B$	$\partial A \cap B^\circ$	$A^\circ \cap \partial B$	Relationship	Graphic
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	A disjoint B	
$\neg\emptyset$	$\emptyset$	$\neg\emptyset$	$\emptyset$	A inside B	
$\neg\emptyset$	$\emptyset$	$\emptyset$	$\neg\emptyset$	A contains B	
$\neg\emptyset$	$\neg\emptyset$	$\emptyset$	$\emptyset$	A equals B	
$\emptyset$	$\neg\emptyset$	$\emptyset$	$\emptyset$	A meets B	
$\neg\emptyset$	$\neg\emptyset$	$\neg\emptyset$	$\emptyset$	A covered by B	
$\neg\emptyset$	$\neg\emptyset$	$\emptyset$	$\neg\emptyset$	A covers B	
$\neg\emptyset$	$\neg\emptyset$	$\neg\emptyset$	$\neg\emptyset$	A overlaps B	
$\neg\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	not valid	
$\emptyset$	$\emptyset$	$\neg\emptyset$	$\emptyset$	not valid	
$\emptyset$	$\emptyset$	$\emptyset$	$\neg\emptyset$	not valid	
$\emptyset$	$\emptyset$	$\neg\emptyset$	$\neg\emptyset$	not valid	
$\emptyset$	$\neg\emptyset$	$\neg\emptyset$	$\emptyset$	not valid	
$\emptyset$	$\neg\emptyset$	$\emptyset$	$\neg\emptyset$	not valid	
$\emptyset$	$\neg\emptyset$	$\neg\emptyset$	$\neg\emptyset$	not valid	
$\neg\emptyset$	$\emptyset$	$\neg\emptyset$	$\neg\emptyset$	not valid	

Table 6.9: Topological relationships based on intersections of boundaries and interiors.

effective, the filtering operation should only remove data that would not yield interesting results at a higher resolution level. In other words, the filtering operation weeds out data that is proven not useful at the current resolution level and at all finer resolutions. If the filtering removes data that is determinant at a higher resolution after all, the final results could be incomplete or even erroneous. This is the reason why the number of occurrences, for example, is not used to filter out non-frequent item-sets from one resolution level to the other, since one rough locale could end up being two distinct locales at a finer resolution. Thus, an infrequent two occurrences of a visual feature could become more frequent at a higher resolution. As mentioned in Section 6.1.1, the same applies for visual features like colour and texture: some are better preserved than others from fine to coarse resolution. Spatial relationships between locales are non deterministic from one resolution level to a finer resolution level. In other words, a given topological configuration between two areas can become a different topological configuration at a higher resolution level.

As stated in Section 6.1.1, we envision three models for resolution refinement for objects (or locales) in visual media: pixel based, bounding circle based, and tile size based. We will discuss in the following subsections the resolution refinement with exclusion of minimum bounding circles and the resolution refinement with resizing of locale tiles.

Resolution Refinement with Exclusion of Minimum Bounding Circles

With the bounding circle model, locales are first roughly estimated by a circle that comprises the totality of the locale. A minimum bounding circle is the smallest circle that could contain the whole locale. The centroid of the locale is taken as the centre of the circle and the longest distance across the locale is the diameter of the minimum bounding circle. While there could be many different minimum bounding rectangles for a polygon, there is only one unique minimum bounding circle (Figure 6.6).

Resolution with minimum bounding circle has only two levels. In the first level (rough resolution), the locales are approximated by circles and the topology is based on the bounding circles. In the second level (fine resolution), the circles are excluded and the topology is based on the envelope and mass of the locales (See Chapter 4 about locale definition and characteristics). Obviously, the topology (i.e spatial relationships) could change from the configuration with the bounding circles to the configuration without the bounding circles and this potential change should be taken into account in the filtering process. As mentioned before, there is a finite number of spatial relationships (8). For the filtering process to be advantageous, a given spatial relationship at a rough resolution should not potentially result in all the eight other topologies at the fine resolution, but only lead to a limited number of possible new spatial relationships (< 8).

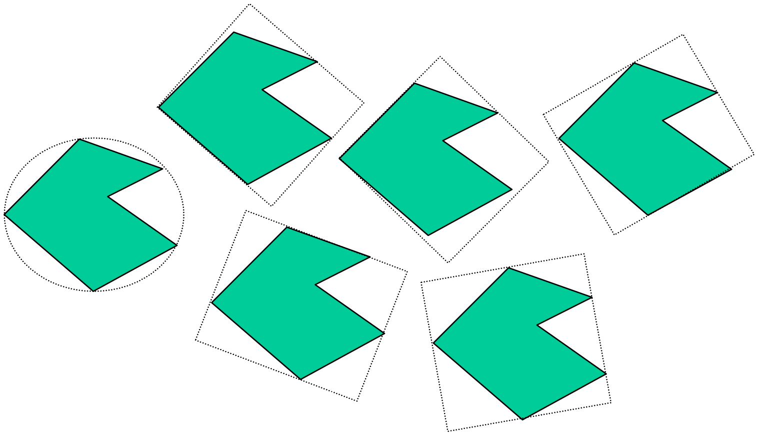


Figure 6.6: Minimum bounding circle and minimum bounding rectangles.

When taken one spatial relationship type at a time, it is possible to determine the outcome of a resolution refinement (i.e. elimination of the minimum bounding circle) by analyzing the valid combinations of intersections between boundaries and interiors. Let A and B be two areas at a rough resolution (i.e. approximated to their minimum bounding circles), and a and b the same areas at a higher resolution (i.e. without minimum bounding circle). $\partial A, \partial B, \partial a$ and ∂b are the boundaries of respectively A, B, a and b, and $A^\circ, B^\circ, a^\circ$ and b° are the interiors of A, B, a, and b respectively.

The topological relation between two areas A and B at any resolution level is defined by a matrix $\mathcal{R}$:

$$\mathcal{R}(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & \partial A \cap B^\circ \\ A^\circ \cap \partial B & \partial A \cap \partial B \end{pmatrix}$$

The following are the eight possible topological relations and their valid outcomes with a resolution refinement by elimination of the minimum bounding circle:

- *A disjoint B* $(A^\circ \cap B^\circ = \emptyset, \partial A \cap \partial B = \emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \emptyset)$

  $A^\circ \cap B^\circ = \emptyset \Longrightarrow a^\circ \cap b^\circ = \emptyset$

  $\partial A \cap \partial B = \emptyset \Longrightarrow \partial a \cap \partial b = \emptyset$

  $\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset$

$A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset$

$$\mathcal{R}(a,b) = \begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix}$$

- A *inside* B $(A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \emptyset, \partial A \cap B^\circ = \neg\emptyset, A^\circ \cap \partial B = \emptyset)$

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \neg\emptyset \Longrightarrow \partial a \cap b^\circ = \neg\emptyset$

  $A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset \vee a^\circ \cap \partial b = \neg\emptyset$

  $$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset \end{pmatrix}$$

- A *contains* B $(A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \neg\emptyset)$

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset \vee \partial a \cap b^\circ = \neg\emptyset$

  $A^\circ \cap \partial B = \neg\emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset \vee a^\circ \cap \partial b = \neg\emptyset$

  $$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix}$$

- A *equals* B $(A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \neg\emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \emptyset)$

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset \vee \partial a \cap b^\circ = \neg\emptyset$

  $A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset \vee a^\circ \cap \partial b = \neg\emptyset$

  $$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \emptyset \\ \emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix}$$

- A *meets* B $(A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \emptyset)$

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset$

  $A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset$

$$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix}$$

- *A covered by B ($A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \neg\emptyset, \partial A \cap B^\circ = \neg\emptyset, A^\circ \cap \partial B = \emptyset$)*

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \emptyset \vee a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \neg\emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset \vee \partial a \cap b^\circ = \neg\emptyset$

  $A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset$

  $$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset \end{pmatrix}$$

- *A covers B ($A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \neg\emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \neg\emptyset$)*

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \emptyset \vee a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset$

  $A^\circ \cap \partial B = \neg\emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset \vee a^\circ \cap \partial b = \neg\emptyset$

  $$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset \end{pmatrix}$$

- *A overlaps B ($A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \neg\emptyset, \partial A \cap B^\circ = \neg\emptyset, A^\circ \cap \partial B = \neg\emptyset$)*

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \emptyset \vee a^\circ \cap a^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \neg\emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset \vee \partial a \cap b^\circ = \neg\emptyset$

  $A^\circ \cap \partial B = \neg\emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset \vee a^\circ \cap \partial b = \neg\emptyset$

  $$\mathcal{R}(a,b) = \begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix}$$

  The matrix $\begin{pmatrix} \neg\emptyset & \emptyset \\ \emptyset & \neg\emptyset \end{pmatrix}$ is not valid in this context because a and b could not be equal, since A overlaps B and by definition the minimum bounding circle of an area is unique. If A and B overlap, this means that their minimum bounding circles are

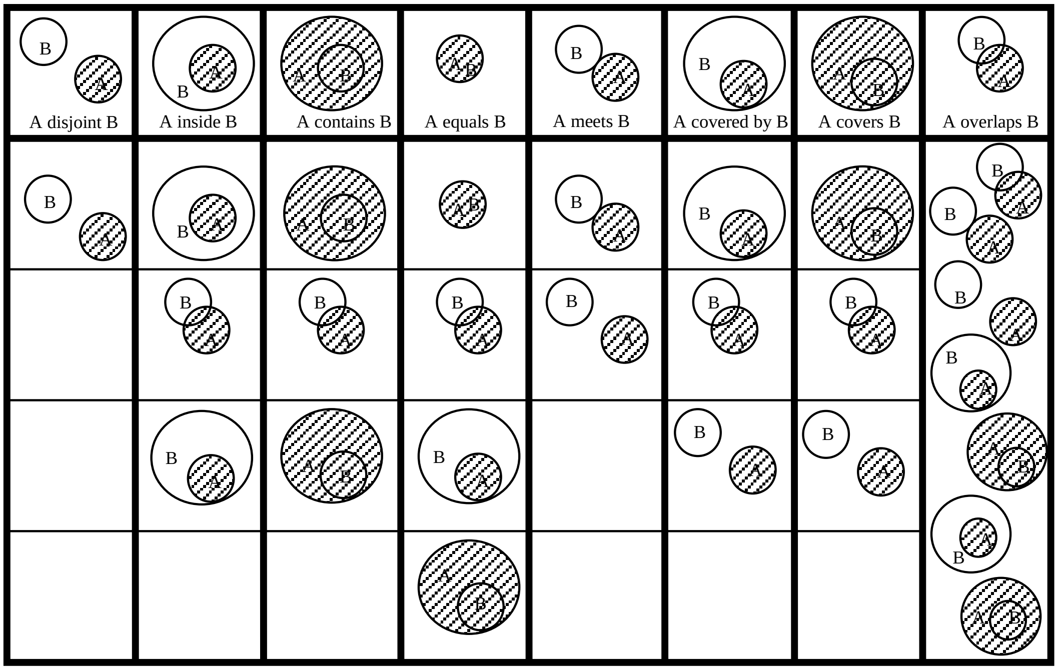


Figure 6.7: Topology and resolution increase with minimum bounding circles.

different, thus a and b could not be equal, otherwise it contradicts the unicity of the minimum bounding circle.

The table in Figure 6.7 graphically summarizes the possible topological changes when minimum bounding circles are eliminated for resolution refinement. Figure 6.8 gives examples for each case.

Resolution Refinement with Tile Resizing

Tiles are squares of 2 × 2, 4 × 4, 8 × 8, 16 × 16 or 32 × 32 pixels. As presented in Chapter 4, a tile can have different colours (or other visual features) at the same time; the bigger the tile is, the higher the likelihood of a multi-valued visual attribute. Considering colour as an example of a visual feature and starting from a large 32 × 32 tile, the more the tile shrinks the more it is possible to distinguish between the different colours of the original tiles. In other words, if the resolution refinement consists of dividing each tile by four, the four new tiles would share or split the colours among themselves. It is also possible (for peripheral tiles) that the new smaller tiles would lose the features from the parent tile. If colour is the building factor for locales (see Chapter 4), whenever a tile is divided into four

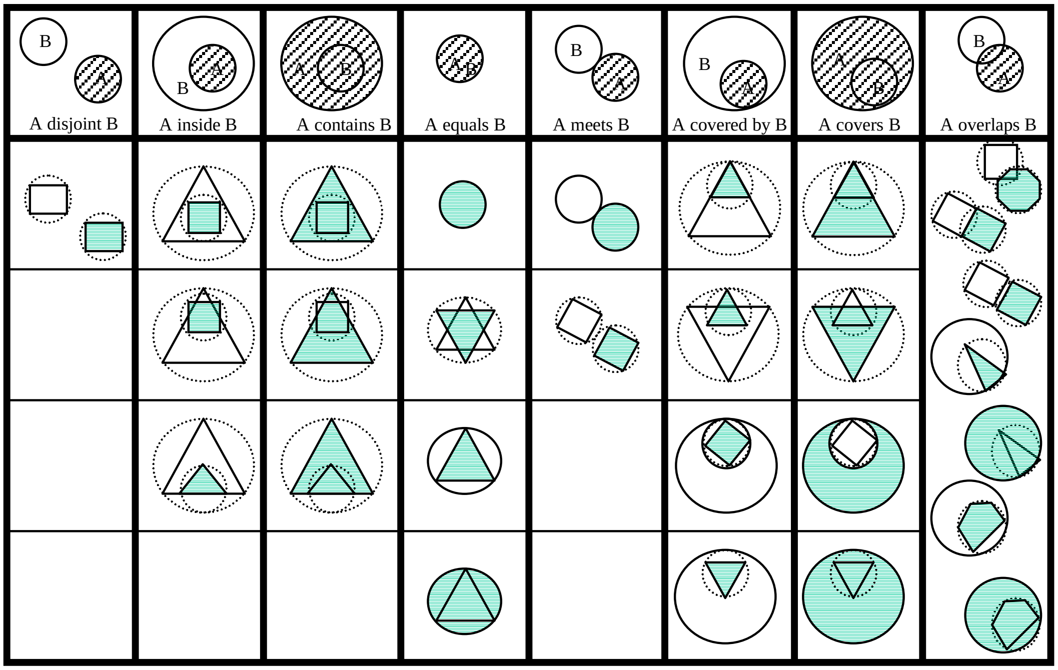


Figure 6.8: Examples of topological refinement with minimum bounding circles.

child tiles from one resolution level to a finer one, at least one child tile inherits the colour. This is true for peripheral tiles. Peripheral tiles are considered the boundary of a locale, and all other tiles are the interior. Figure 6.9 shows a locale with boundary and interior. The intersection between boundaries and interiors of locales is based on shared tiles. For two locales A and B, ∂A is intersecting ∂B if there exist a tile belonging to the boundary of A and the boundary of B. A° is intersecting B° is there exists a tile belonging at the same time to the interior of A and the interior of B. The same applies to $\partial A \cap B^\circ$ and $A^\circ \cap \partial B$. Neighbouring tiles are not intersecting (See Figure 6.9).

Figure 6.10 illustrates the progressive refinement in the case of shrinking tiles. The roughest resolution level uses a 32×32 tile size. Each finer resolution level divides the tile size by four. Since the child tiles of peripheral tiles may lose colour, the boundary of a locale may "retreat" inward which may result in a topological change from rough resolution to finer.

The following are the eight possible topological relations and their valid outcomes with a resolution refinement by shrinking the tiles:

- *A disjoint B* ($A^\circ \cap B^\circ = \emptyset, \partial A \cap \partial B = \emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \emptyset$)

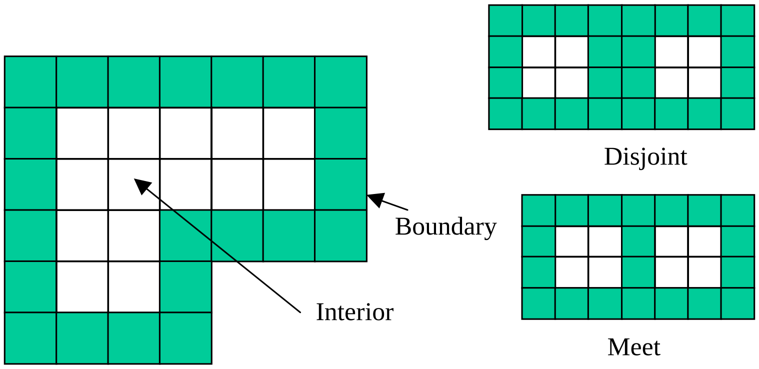


Figure 6.9: Locale envelope with boundary tiles.

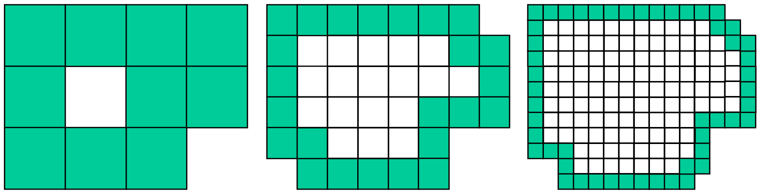


Figure 6.10: Progressive tile shrinking.

$$A^\circ \cap B^\circ = \emptyset \Longrightarrow a^\circ \cap b^\circ = \emptyset$$

$$\partial A \cap \partial B = \emptyset \Longrightarrow \partial a \cap \partial b = \emptyset$$

$$\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset$$

$$A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset$$

$$\mathcal{R}(a,b) = \begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix}$$

- *A inside B* $(A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \emptyset, \partial A \cap B^\circ = \neg\emptyset, A^\circ \cap \partial B = \emptyset)$

$$A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$$

$$\partial A \cap \partial B = \emptyset \Longrightarrow \partial a \cap \partial b = \emptyset$$

$$\partial A \cap B^\circ = \neg\emptyset \Longrightarrow \partial a \cap b^\circ = \neg\emptyset$$

$$A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset$$

$$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset \end{pmatrix}$$

- *A contains B* $(A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \neg\emptyset)$

$$A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$$

$\partial A \cap \partial B = \emptyset \Longrightarrow \partial a \cap \partial b = \emptyset$

$\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset$

$A^\circ \cap \partial B = \neg\emptyset \Longrightarrow a^\circ \cap \partial b = \neg\emptyset$

$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset \end{pmatrix}$

- *A equals B* ($A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \neg\emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \emptyset$)

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \emptyset \vee a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset \vee \partial a \cap b^\circ = \neg\emptyset$

  $A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset \vee a^\circ \cap \partial b = \neg\emptyset$

  $\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \emptyset \\ \emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee$
  $\begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix}$

- *A meets B* ($A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \emptyset$)

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset$

  $A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset$

  $\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix} \vee \begin{pmatrix} \emptyset & \emptyset \\ \emptyset & \emptyset \end{pmatrix}$

- *A covered by B* ($A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \neg\emptyset, \partial A \cap B^\circ = \neg\emptyset, A^\circ \cap \partial B = \emptyset$)

  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \neg\emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset \vee \partial a \cap b^\circ = \neg\emptyset$

  $A^\circ \cap \partial B = \emptyset \Longrightarrow a^\circ \cap \partial b = \emptyset \vee a^\circ \cap \partial b = \neg\emptyset$

  $\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset \end{pmatrix}$

- *A covers B* $(A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \neg\emptyset, \partial A \cap B^\circ = \emptyset, A^\circ \cap \partial B = \neg\emptyset)$
  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \emptyset \vee \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \emptyset \Longrightarrow \partial a \cap b^\circ = \emptyset \vee \partial a \cap b^\circ = \neg\emptyset$

  $A^\circ \cap \partial B = \neg\emptyset \Longrightarrow a^\circ \cap \partial b = \neg\emptyset$

  $$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix} \vee \begin{pmatrix} \neg\emptyset & \emptyset \\ \neg\emptyset & \emptyset \end{pmatrix}$$

- *A overlaps B* $(A^\circ \cap B^\circ = \neg\emptyset, \partial A \cap \partial B = \neg\emptyset, \partial A \cap B^\circ = \neg\emptyset, A^\circ \cap \partial B = \neg\emptyset)$
  $A^\circ \cap B^\circ = \neg\emptyset \Longrightarrow a^\circ \cap b^\circ = \neg\emptyset$

  $\partial A \cap \partial B = \neg\emptyset \Longrightarrow \partial a \cap \partial b = \neg\emptyset$

  $\partial A \cap B^\circ = \neg\emptyset \Longrightarrow \partial a \cap b^\circ = \neg\emptyset$

  $A^\circ \cap \partial B = \neg\emptyset \Longrightarrow a^\circ \cap \partial b = \neg\emptyset$

  $$\mathcal{R}(a,b) = \begin{pmatrix} \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset \end{pmatrix}$$

The table in Figure 6.11 graphically summarizes the possible topological changes when the tile size is reduced for resolution refinement. Figure 6.12 gives examples for each case.

6.3 Performance of MaxOccur Algorithm

We have implemented Algorithm 6.1.2 to generate sets of synthetic images as presented in Table 6.10, each image transaction had up to 15 objects. The different sized image sets which were produced were intended to demonstrate the scalability of the algorithms and compare their performance. Since The algorithm for mining multimedia association rules with recurrent spatial relationships uses the Max-Occur algorithm after two extra scans of the data set, we will only show in this section the performance of MaxOccur. It is obvious that the scalability of both algorithms are related. We implemented the Apriori algorithm [8] and two versions of the MaxOccur algorithm, as well as the naïve algorithm presented earlier, in ANSI *C* on a PC 166Mhz and Ultra-Sparc workstation, both with 64Mb of main memory. Since the Apriori algorithm uses the number of transactions as support, and we wanted to compare our algorithm with Apriori, we have implemented MaxOccur and the naïve with transaction based support (MaxOccur1). The second version of MaxOccur

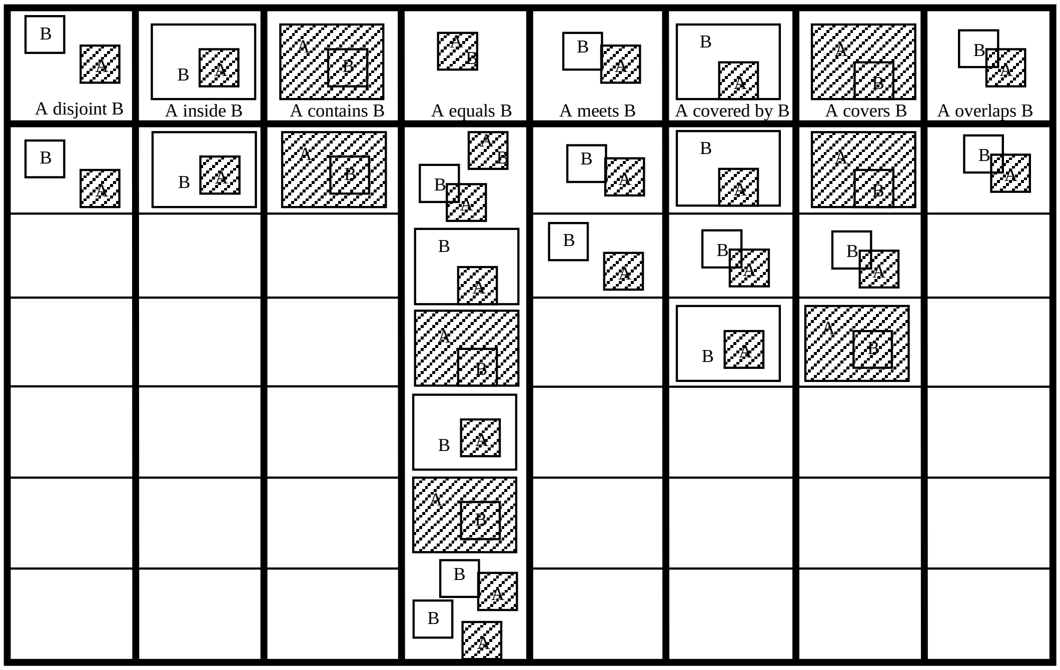


Figure 6.11: Topology and resolution increase with tile size shrinking.

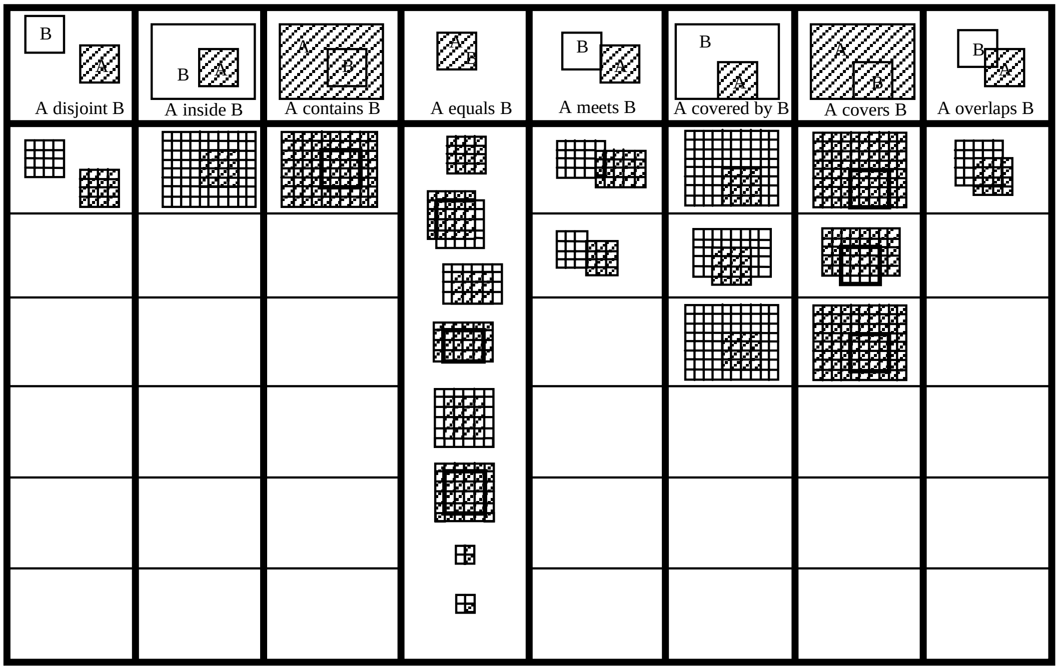


Figure 6.12: Examples of topological refinement with tile size shrinking.

(MaxOccur2) used the object-based support as presented in Algorithm 6.1.3. Surprisingly, the algorithms run 40% faster on the PC than on the Ultra-Sparc workstation. We believe it is a network overhead since the PC used a local disk, while the Unix machine was connected to a network. Table 6.11 shows the average execution times for the four algorithms running on the PC with different image set sizes and $\sigma\prime = 0.05$ for Apriori, “Naïve” and MaxOccur1, and 0.0035 for MaxOccur2. The results are graphically illustrated in Figure 6.13. Clearly, MaxOccur scales well with both versions treating one thousand images in 1.3 seconds, on average, regardless of the size of the data set. The running time for filtering the frequent item-sets with $\sigma\prime$, the maximum support threshold (line 16 of Algorithm 6.1.3), is negligible since it is done in main memory once the frequent item-sets are determined. Moreover, the calculation of the total number of items (line 4 of Algorithm 6.1.3) is done during the first scan of the data set and has limited repercussion on the algorithms' execution time. The major difference between Apriori and MaxOccur is in ascertaining the candidate item-sets and counting their repeated occurrences in the images. Obviously, MaxOccur discovers more frequent item-sets. The naïve algorithm also finds the same frequent item-sets but is visibly capable of less performance in execution time. Figure 6.15 shows the average number of frequent item-sets discovered with the three algorithms: Apriori found on average 109 different frequent k-item-sets, while MaxOccur1 and Naïve found 148 on the same data sets, and MaxOccur2 found 145 on average. The discrepancy between MaxOccur1 and MaxOccur2 is basically due to the different definition of support. The price we pay in performance loss with MaxOccur, is gained by more frequent item-sets and thus, more potentially useful association rules discovered. We have experimented with different settings of support thresholds with MaxOccur and we found that the scalability is not compromised. The curves in Figure 6.14 show that while the performance is reduced when the minimum threshold is reduced, the scalability remains. The foreseeable reduction in the performance is due to the increase of the number of frequent item-sets in each round ($\mid F_i \mid$) because of the lower support filter in line 14 of Algorithm 6.1.3. This intuition is also predictable and true with the Apriori algorithm.

6.4 On-going work and Conclusions

In this chapter, we have introduced multimedia association rules based on image content and spatial relationships between visual features in images using coarse to fine resolution

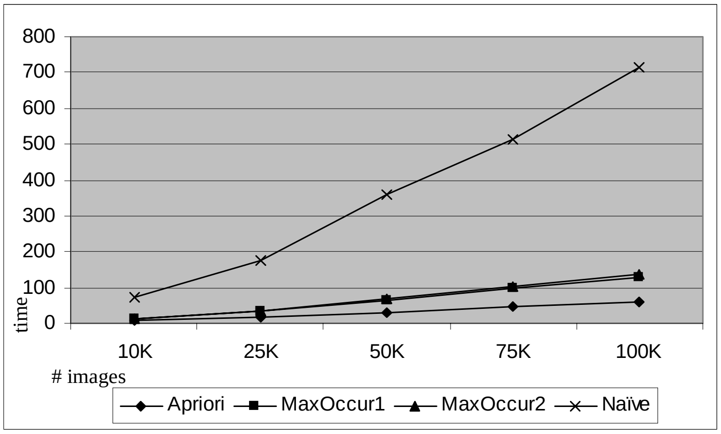


Figure 6.13: Scale up of the algorithms.

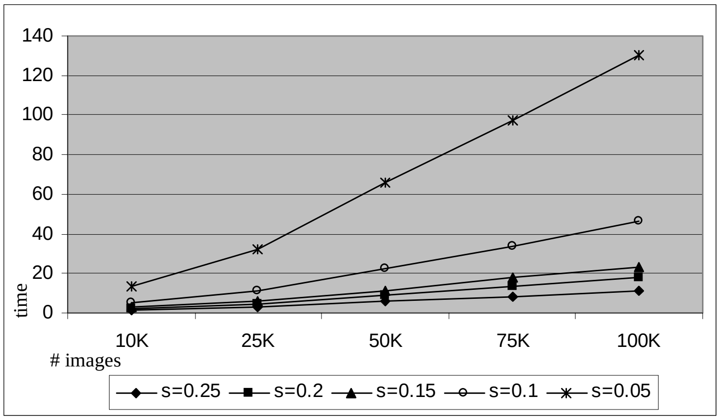


Figure 6.14: Performance with variable σ*ı* values.

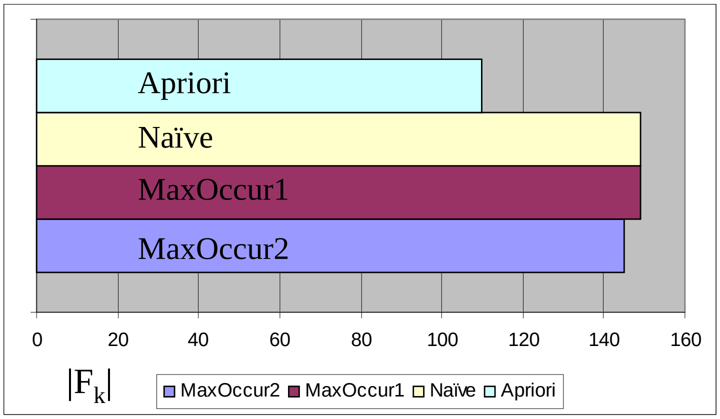


Figure 6.15: Frequent item-sets found by the different algorithms.

Database	Number of images	Size
MM-10	10,000	1.5MB
MM-25	25,000	3.7MB
MM-50	50,000	7.5MB
MM-75	75,000	11.3MB
MM-100	100,000	15.2MB

Table 6.10: Sample Databases of Synthetic Image Descriptors.

# of images	Apriori	Naïve	MaxOccur1	MaxOccur2
10K	6.43	70.91	13.62	13.68
25K	15.66	176.69	32.35	34.11
50K	30.54	359.38	66.07	67.44
75K	44.93	514.33	97.27	101.23
100K	60.75	716.01	130.12	137.81

Table 6.11: Average execution times with different number of images.

approach. We have put forth a Progressive Resolution Refinement (PRR) approach for mining visual media at different resolution levels, and have presented two algorithms for the discovery of content-based multimedia association rules. These rules would be meaningful only in a homogeneous image collection; a collection of semantically similar images or received from the same source channel. For other heterogeneous image collections, the descriptor-based association rules described in the previous chapter, and implemented in the MultiMediaMiner, would suffice. We have also formally presented the topological changes with resolution refinement. We have enumerated all changes in spatial relationships with

# of images	$\sigma\prime = 0.25$	$\sigma\prime = 0.20$	$\sigma\prime = 0.15$	$\sigma\prime = 0.10$	$\sigma\prime = 0.05$
10K	1.43	2.20	2.70	5.06	13.51
25K	2.80	4.78	6.31	11.20	32.35
50K	6.27	9.28	11.59	22.74	66.07
75K	8.24	13.57	17.69	33.94	97.27
100K	11.32	17.63	23.13	46.74	130.12

Table 6.12: Average execution time of MaxOccur with different thresholds.

the minimum bounding circle model and the locale tile shrinking model.

One major improvement in the performance of the multimedia association rules discovery algorithms is the addition of some restrictions on the rules to be discovered. These restrictions would add focus on the data set to be analyzed. A data mining query with DMQL (see Chapter 3), for example, would substantially reduce the data set and put a focal point on the interesting information to be discovered. While reducing and focusing the data set would lessen the execution time, it does not necessarily improve the performance of the algorithms. However, giving some restrictions on the form of the rules to be discovered could decrease the execution time. Such restrictions could be given in a meta rule form (see DMQL in Chapter 3 and Appendix C). *MetaQueries* and Meta-Rule Guided Mining of association rules have been presented in [227] and [105]. Meta rules (or queries) are logical descriptions in the form of the rules to be discovered. Such logical "templates" could be for example $P(u, v) \wedge ... \wedge Q(w, x) \rightarrow R(y, z)$, where P, Q and R are predicate variables bound to any concrete variable (or spatial relationship), and u, v, w, x, y and z are variables bound to any data in the database or constants from the data set at multiple conceptual levels. Meta-rule guided mining consists of discovering rules that not only are frequent and confident, but also comply with the meta-rule template. The meta rule template can convey a lot of information that could be exploited to improve the performance of multimedia association rule mining. For example, the conjunction of the antecedent and the consequent of the meta rule indicates the size of the frequent item-sets needed. In other words, k is revealed by the template and would help stop the algorithm's main loop when necessary and sufficient k-item-sets are found. Moreover, the predicate variables P, Q, R and the variables u, v, w, x, y, z in the rule template can substantially reduce the candidate set size $\mid C_k \mid$. For example, with a meta rule such as "H-Next-to$(X, Y) \wedge$ Colour$(x$, red$) \wedge$ Overlap$(Y, Z) \rightarrow P(Y, Z)$" one need only to find frequent 3-item-sets of the form {H-Next-to(red, Y), Overlap$(Y, *)$, $P(Y, *)$} where Y is an attribute value and P a visual descriptor or spatial relationship predicate defined in Table 6.2. Obviously, such a filter would greatly reduce the complexity of the search problem. A method for exploiting meta-rules for mining multi-level association rules is given in [105].

In the process of discovering multimedia association rules with recurrent spatial relationships, we have assumed the existence of the enumerated spatial relationships such as in Table 6.2. These relationships are simply processed by comparing the centroid of each locale as well as the radius of the locale's shape approximated to a circle. The centroids and the radii of locales are sufficient to rapidly and efficiently give a good approximation of spatial relationships between objects in an image such as closeness, overlap and containment. There exist other methods for determining more precise spatial relationships. However, these methods to be effective can be computationally costly. Ideally, we would preprocess once the detailed spatial relationships at a fine granularity and have a table such as Table 6.2 provided to the mining module. If this computation is not preprocessed before the discovery of association rules, a first step could determine rough spatial relationships and discover association rules with these high level approximate spatial relationships; then, at a second stage, refine the spatial relationships for only the frequent item-sets discovered. Notice that a concept hierarchy could be built on top of the spatial relationships, and an iterative approach could be applied to discover sufficiently strong multimedia association rules with recurrent spatial relationships from less accurate to refined spatial relationships.

Object recognition (or identification) in image processing and computer vision is a very active research field. Accurately identifying an object in a video, for example, as being a object in itself, is a very difficult task. We believe that data mining techniques can help in this perspective. Multimedia association rules with spatial relationships using the motion vector of locales as a conditional filter, can be used to discover whether locales moving together in a video sequence are part of the same object with a high confidence. While this cannot ascertain all objects in a set of video frames, it may help distinguish between composite locales that might be objects and those that are definitely not composite objects.

Man stays wise as long as he searches for wisdom;
as soon as he thinks he has found it, he becomes a fool.
Talmud

Chapter 7

Conclusion

The information revolution is upon us. We are publishing all sorts of documents at an amazing speed. In fact, we are overwhelmed by these publications. Techniques to manage this excess of resources and retrieve pertinent documents when needed, have yet to be developed. The phenomenal growth of the Internet, in terms of resources it contains and in terms of users accessing these resources, shows the urgency and the crucial need for efficient and effective resource discovery techniques.

Much of the bandwidth on the Internet is taken up by transmission of visual data. Furthermore, it can be argued that image and video processing will be a central technology in the information age. Visual data from satellites to video cameras and medical scanners are provided in remarkable overwhelming amounts. The bulk of this data is never analyzed or used because of lack of efficient and scalable techniques.

We have addressed in this thesis both of these problems. We have studied resource and knowledge discovery from the Internet and investigated data mining from image collections. In the following sections, our work is summarized and research directions are discussed.

7.1 Summary of the Thesis Work

In this thesis, we have demonstrated the inefficiency and inadequacy of the current information retrieval technology applied on the Internet. We have proposed a framework, called Virtual Web Views, for intelligent interactive information retrieval and knowledge discovery from global information systems, and have put forward a query language, WebML, for resource discovery and data mining from the Web using the Virtual Web Views.

We have illustrated in this thesis how descriptors collected for virtual web view building can be exploited for content-based image retrieval, and have shown how to carry out on-line analytical processing and data mining on visual data from the World-Wide Web, or other multimedia repositories.

The major contributions of this thesis are summarized as follows:

1. We have proposed a framework in which virtual web views can be created to make the web appear structured and use database technology to efficiently and interactively retrieve information and discover implicit knowledge from the Internet. A multiple layered database approach is used with concept hierarchies along which data is generalized at high conceptual level. Different scenarios for interaction between heterogeneous views via a mediating agent are also proposed. This work has been published in [127, 128, 276, 275].

2. We have proposed WebML a declarative web mining language for resource and knowledge discovery from the Internet. The language allows interactive and progressive querying of global information systems seen through virtual web views. Embedded in other traditional programming languages, WebML could be used as a programming language for Web mining, more than just an interactive query language. The language has been published in [277].

3. Using the image descriptors extracted for summarizing images and videos in the Virtual Web Views, we have designed and developed a system prototype, C-BIRD, for content-based image retrieval from large image and video databases. The system was demonstrated at the CASCON conference in 1997. It allows image search based on colour and texture characteristics as well as search by objects contained in images. We have put forth a feature localization and a three-step matching algorithm to support search by object model. The system is also linked to the Internet allowing resource discovery in the World-Wide Web based on image and video containment. The results of this research was published in [168, 169, 167].

4. We have designed and developed an multimedia data mining system prototype, MultiMediaMiner, which offers on-line analytical processing (OLAP) with multi-dimensional data cubes built on top of the descriptors initially extracted for content-based image

retrieval, and descriptor-based data mining from multimedia repositories. MultiMediaMiner, which was demonstrated at the ACM SIGMOD conference in 1998 [279], has the following features: (i) a multi-dimensional multimedia data cube, (ii) multiple data mining modules, including characterization (or summarization), association, and classification, and (iii) an interactive mining interface and display with Web information retrieval capabilities. The system combines data mining and information retrieval from the Web by "drilling through" the visualized results. Our preliminary experiments demonstrate that multimedia data mining may lead to interesting and fruitful knowledge discoveries in multimedia databases. The result of this research was published in [280].

5. We have studied and introduced association rules with recurrent items which are association rules with items that may appear more than once in the same rule. This type of association rules is crucial in the multimedia context, especially for images where features such as colours, textures, shapes, etc., can be repeated. We have presented two scalable algorithms for the discovery of content-based multimedia association rules with recurrent objects and multimedia association rules with recurrent spatial relationships between visual features. The algorithms and the results of the comparative experiments conducted were submitted for publication in [281].

6. We have developed and implemented a data mining system DBMiner [125, 120] for mining several kinds of knowledge and rules. Figure 3.2 to 3.5 in Chapter 3 show examples of outputs of this system. As depicted in Figure 7.1, we have implemented DBMiner with a multi-tier client-server architecture in a web-based environment exceeding the current stateless conjuncture of web servers by simulating user status and implementing communication protocols between the different tiers. Our web-based implementation of DBMiner was demonstrated at CASCON conference in 1996.

7. We have designed a data mining query language DMQL to describe needs and constraints in a data mining process. The language lets users specify the data mining task at hand and the representation needed. It was implemented in the DBMiner system and was published in [126]. The language was inspirational in the design of WebML.

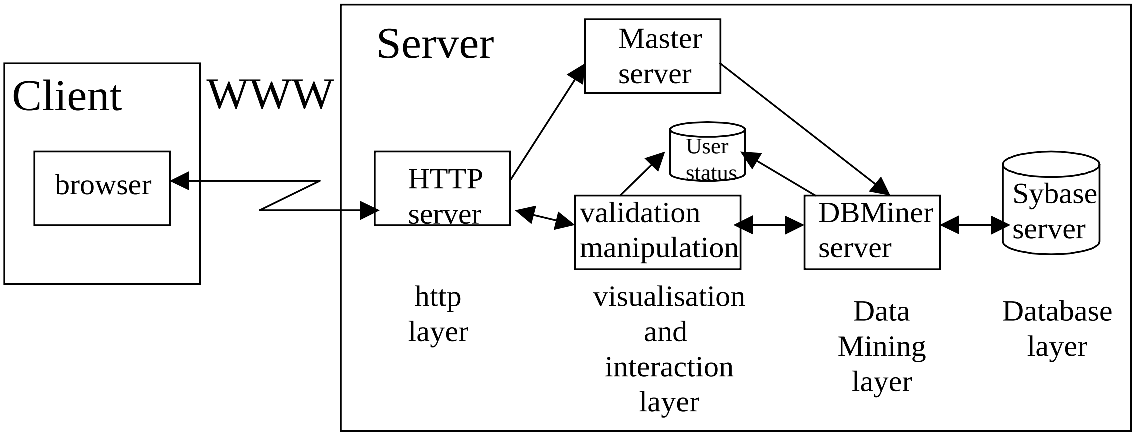


Figure 7.1: DBMiner multi-tier client-server architecture for the web-based implementation

7.2 Discussion and Research Directions

- The massive and overwhelming aggregate of various documents on the Internet is obvious. This continuous accumulation of documents is not only large, but also heterogeneous. Creating a data warehouse for Internet data (Web Warehousing) is a good idea to alleviate some this heterogeneity problem and appease information gathering as well as resource and knowledge discovery issues. Our study presents a general framework of the VWV approach for the resource and knowledge discovery in the global information system. More studies are needed in the construction and utilization of the global multiple layered databases. A larger scale implementation and experiment for an automatic construction and maintenance of a global MLDB, is needed to study the efficiency of the initial design and probably refine and improve it.

- XML is widely predicted to improve the degree of interoperability between applications on the Internet. However, the utility of XML is severely limited unless the semantics of terms used in metadata is agreed upon. This has lead researchers to work on new studies in an attempt to put forth means to provide interoperability between applications that exchange information on the Internet. The Dublin Core elements, already in use, could be a starting point to build a warehouse (i.e. VWV) with a niche of resources that uphold the element set. The Resource Description Framework (RDF) currently being revised by the World-Wide Web Consortium, is a domain-neutral mechanism for describing on-line resources and a new foundation for processing metadata. Once ratified, it could be adopted by the information extraction tools (i.e. for creation of Layer-1 of the VWV) which would comply with other web applications for information exchange. The VWV itself could be accessed and queried through the RDF.

- We have presented some scenarios in which co-existing heterogeneous VWVs exchanged information through Mediators as intermediary. Mediators allow VWVs to specialize in geographic areas or topics, and count on other VWVs that warehouse other web artifacts, when extra data is needed. However, Mediators may need to translate exchanged data from ontologies to others if a global or common ontology is not available. Common ontologies for restricted applications are in the works, like the Ontology.com initiative for the electronic commerce domain. Many industries are working on standard vocabulary and semantic requirements to ease interoperability in their application domain. There is nevertheless a need for efficient algorithms for intra and inter-domain ontology translations, in order for Mediators to play their role of intermediary between VWVs.

- Web usage mining is a new research field that is drawing the attention of many scholars and industry people. Unfortunately, web server logs collect only very limited information. The structure and the content of web access logs were not designed for data mining purposes. Data mining on this compilation of data necessitates expensive and complex data cleaning and transformation. Established data mining techniques are very difficult in this context. Often, information not collected in the logs are needed. Further research is required in the collection of web access information, the data cleaning and transformation procedures, and in sophisticated algorithms. Moreover, there is still an important challenge due to missing information resulting from caching practices at different levels, client browser, proxy server, etc. Accessing a cache instead of a web site can mislead web access data analysis. In addition, the HTTP protocol used for web information exchange makes it very difficult to identify user sessions accurately, especially that the protocol is state-less. New protocols are being proposed to solve some of these problems.

- Web usage mining, right from the beginning, is part of a large privacy debate involving the personal data web servers retrieve from users when they browse for leisure or for buying in electronic commerce sites. It is obvious that for marketing purposes, electronic businesses track what users do on a site, when they access, and try to correlate it with who the users are. The marketers' argument is that users may benefit from this practice. By understanding surfing patterns and on-line buying habits, electronic businesses can provide better services and even personalized services. On-line businesses even maintain preferential treatment by serving the best customers first and excluding occasional users from special deals or services. Privacy advocates worry secondarily about the sophisticated profiles created, but are specially and primarily concerned with the proper use of this personal data accumulated. Several companies are sharing and even selling user information. This has lead to the Big Brother syndrome–many web users refrain from making business on-line out of a concern over the collection of personal data. Because of the stern warnings from privacy advocates, more and more on-line services and retailers are adopting policies for non-divulging transaction histories or releasing customer names. The issue is often more than just a simple orientation: marketer versus customer, but imposing or allowing anonymity in web transactions. Some even believe that posting anonymously in newsgroups and chat rooms should be allowed. Some services like Anonymizer.com and many others are making private surfing possible. However, for on-line shopping, retailers still have to get addresses to ship the purchased goods and credit cards are used for payments. Some digital cash schemes are being proposed that allow on-line payments that cannot be correlated back to any other on-line identity. It remains that the privacy issue has not been addressed in a meaningful way by the Internet community, and finding a good balance between personal privacy and marketing is paramount for a righteous growth for electronic commerce. It is important to note that privacy issues are not proper to web mining but are problematic for data mining at large. Freedom and privacy organizations are dealing with the problem at a national and international level trying to find solutions acceptable by all parties. It is sure though, that the dialectic will remain since privacy issues are a very delicate matter.

- We have demonstrated how WebML, our declarative language for mining the Web, can be very efficient and powerful for resource discovery and progressive information retrieval from the Web. We have also illustrated the effective capabilities of the language for data mining in the Web. However, the current design of the language does not emphasize web usage mining as it should be. The only information recovered from web log files in the VWV that the language takes advantage of, are access counts. This limits WebML in the expression of web usage mining tasks. For example, usage behaviour studies, clustering of users, correlation between requesters and resources, etc., can not be done with the current WebML.

- We have proposed the VWVs using MLDB structure with the extended relational model. The relational model was used for simplicity and efficiency. However, because of its flexible structure, the object-oriented model seems more suited for complex data types and to handle the unstructured and semi-structured data that can be found on the Internet. Besides, determining an effective and efficient design of data mining systems using object oriented databases is still a research issue. Little has been done so far on knowledge discovery from object-oriented databases. Because the object-oriented data model is more suited to support complex data types, it is natural to use object-oriented databases to store descriptors of Internet artifacts. However, new algorithms for attribute-oriented induction and generalization have to be developed for object-oriented databases. It is obvious that all algorithms pertaining to the construction and manipulation of the MLDB, as well as algorithms for data summarization and classification and association rule extraction, tailored for relational databases, do not operate in the object-oriented database context. The challenge is to find new algorithms appropriate for on-line analytical processing (OLAP) operations, data cube handling, and data mining with object-oriented databases. Integrating the object-oriented paradigm in the VWV design and investigating new data mining algorithms specific for object-oriented databases is one of our future objectives.

- The MultiMediaMiner prototype system we implemented, along with C-BIRD, our content-based image retrieval system, could be integrated in a Visual Asset Management application. Managing large video or image data banks, like in the newscast or film industry, is extremely difficult. We have subdivided videos in video sequences represented by some selected frames. Text caption accompanying videos could also be stored in the database. This hierarchical representation along with text caption, content-based retrieval, and OLAP capabilities, are an excellent advantage to investigate in a visual asset management context.

- Multimedia data are described with a large number of attributes, which makes multimedia mining applications highly dimensional. Depending upon the impending data mining task, most of these attributes are usually necessary. However, dealing with a large number of dimensions is problematic. This is called the curse of dimensionality. Each added dimension exponentially increases the search space by the number of distinct values of the related added attribute. Choosing which dimensions to drop and which to keep in order to reduce the size of the multi-dimensional data cube, is a problem in itself. Moreover, constructing the data cube is expensive in time complexity. We have briefly proposed a data cube materialization model, called Multi-Dimensional Database (MDDB), in which data cubes are conceptualized in a database and are materialized only when needed. The model allows a hierarchy of data cubes where materialized cuboids can generalize other cuboids. Further investigation and experiments with this model are necessary.

- We have presented and discussed the discovery of content-based multimedia association rules and association rules with spatial relationships. There are other data mining tasks calling for further research. Content-based descriptors of visual data could also be used for clustering and classifications of images with respect to the attributes of locales present in these images. There are a number of studies on classification and clustering. However, most of the algorithms do not overlap between classes. New algorithms for classification and clustering of multimedia objects, using image content descriptors, should be developed. These algorithms should allow multi-classification of the same objects in different classes.

7.3 Final Thoughts

Isaac Newton once said something like "We can see this far because we stand on the shoulders of those who went before." In this work we have put to use and built on top of research contributed and started by others. We expect that our contribution will be a starting point of many explorations, further study and probably debate in the field of information retrieval and data mining.

The global information network of tomorrow will neither be what the Internet is today nor just become the Memex that Vannevar Bush dreamed of. It will become an entity intelligently organized and used in a way beyond what we might try to foresee in the present.

We believe that the work in this thesis, no matter how insignificant it might be regarded in the future, will contribute in the building of the next generations of sophisticated virtual and physical global information networks.

We know what we are, but know not what we may be.

Shakespeare

The world is moving so fast these days that the man who says it can't be done is generally interrupted by someone doing it.

Elbert Hubbard

Appendix A

Information Retrieval from the Internet

When Johannes Gutenberg introduced the printing press in Europe more than five centuries ago, it spawned a revolution in publishing. Reproduction of publications was enabled on a large scale. Today the World-Wide Web is bringing forth a revolution in electronic publishing. We are witnessing phenomenal mass production of on-line publications of all sorts. New text documents, images, videos, sound files, programs are made available every single moment. However, with this overwhelming production of on-line resources it is very difficult to know what is available and where and how it can be accessed.

At the end of the Second World War, in 1945, Vannevar Bush wrote an article inviting scientists to join the effort in the massive task of building a system holding the sum of human knowledge, and of making this astounding store of information accessible[44]. In his paper, Bush wrote: "The difficulty seems to be, not so much that we publish unduly in view of the extent and variety of present-day interests, but rather that publication has been extended far beyond our present ability to make real use of the record." He described the "Memex," a hypothetical machine that allows a scientist to store thoughts and link available material in 'trails' of thoughts. The World-Wide Web technology and the convenient authoring tools partially mimic Bush's description of the Memex, and are available today not only to scientists, but to millions of users who are continuously contributing to this remarkable wealth of knowledge. However, one issue described by Bush still remains to be solved. Bush wrote: "A record, if it is to be useful... must be continuously extended, it must be stored, and above all it must be consulted." Today the web consents to the storage of an extremely large amount of multimedia and allows its continuous update. However, to make use of this accumulated asset, one has to find the information and retrieve it. An efficient and effective way has yet to be found.

Information retrieval has been a fertile research field. Many techniques have been proposed and implemented in successful and less prevailing applications[209, 142, 218, 115]. With the advent of the World-Wide Web, the appearance of a panoply of services and accumulation of a colossal aggregate of resources, information retrieval techniques have been adapted to the Internet, bringing forth indexing models and search engines. Finding appropriate documents pertinent to a given request is known as Resource Discovery. Resource discovery is the process of clarifying an information retrieval request and identifying and retrieving resources relevant to this request. This is exactly the function that search engines perform. However, the effectiveness of these tools is not satisfactory. The annoying results of current search engine technologies have invited researchers to tackle new challenges. Better indexing approaches, specialized information gathering agents, filtering and clustering methods, etc. have since been proposed.

A new trend in the research in information gathering from the Internet is known as Knowledge Discovery. Knowledge discovery from the Internet is the process of extracting knowledge, or facts, from discovered resources or the global network as a whole. Knowledge discovery on the Internet largely inherits from the data mining domain. This is not to be confused with the access of databases from the World-Wide Web[251, 191] or data mining in relational databases via the Internet. While the goal of resource discovery is to find explicit information, like text documents, multimedia objects, or even web sites, the goal of knowledge discovery is to bring to light implicit knowledge not necessarily stated in any resource, such as classification of web artifacts, correlation between document descriptors, relationships between resources or document content, summarization of resources, etc.

The Internet, since its inception, has evolved in a three dimensional space (Figure A.1). A myriad of services has emerged allowing a variety of resource exchange and protocols. Documents on the Internet have progressed from simple text documents, to semi-structured documents, fully structured documents and multimedia records. Using Artificial Intelligence, tools for information retrieval are becoming more skilled. Starting from manual sifting and tools for resource browsing, some tools now learn to exploit available services for information gathering, and others learn to deduce knowledge from document content. Tools

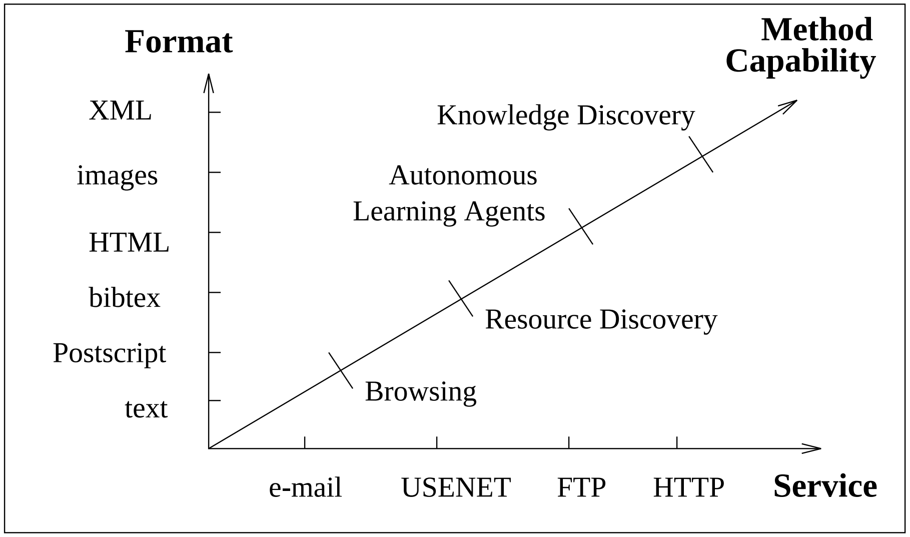


Figure A.1: Internet 3D space representation.

can be categorized into five classes. Tools for direct resource access or browsing are in the first class. Resource discovery tools like mega-indexes and search engines are in the second class. The third class encompasses guided and autonomous agents that, given a specified target or goal, would retrieve or filter information. More sophisticated agents are in the fourth class. These agents have the ability to adapt and learn from experience. They can infer user needs from a user profile and resource search and access patterns. Finally, Web mining tools that discover and extract knowledge from found resources constitute the fifth class. The axes in the figure A.1 represent the available document formats, the prevailing services, and the axis of retrieval capability along which methods are evolving. Each information retrieval method pertains to a set of services, performs on some types of documents, and enjoys a retrieval power. Retrieval methods can be categorized on this last axis depending upon their aptitude in retrieving resources and/or knowledge from the Internet. Etzioni in [82] has compared this evolution to a food chain organization where agents (carnivores) feed on search engines (herbivores) which graze on documents. In [201] Peterson uses the metaphor of biology taxonomy to compare information retrieval tools and their evolution - from kingdom to species. He uses cladistics (i.e. biological classification scheme) to explain the phenomenal rapid development of new tools on the Internet and foresee the outcome for such tools.

In this Appendix we survey techniques for resource discovery on the Internet, and the trends in the knowledge discovery field applied to the Internet. We also survey text retrieval technology in document databases, as the same technology or variations and adaptations of the technology are in the heart of resource discovery methods used on the Internet. A more thorough survey can be found in our publications [270, 271].

A.1 Information Retrieval Technology

Librarians were among the first to adopt computer technology. Early library science researchers dealt with the problem of manual indexing and text retrieval. The introduction of computer technology in library science permitted the automation of some indexing and retrieval. Library catalogues were automated and bibliography databases were created. During the past few decades, much work has been done in information retrieval for library systems[261, 212, 218, 197, 253, 209]. Automated document retrieval became important not only in libraries, but also in other areas like patent offices, law offices for jurisprudence text databases, business offices for computerized text retrieval from documents and memos, etc.

Information retrieval technology has drawn the attention of many researchers, and the phenomenal increase of machine-readable material and network communications has brought new challenges and has stimulated new research[131, 107, 64, 142, 115].

Information Retrieval (IR) is a term coined for the first time in a paper by Mooers in 1952. Today, it is a field that subsumes many topics ranging from data retrieval from databases to knowledge extraction, all related to information processing tasks. It is however regarded as being synonymous with document retrieval. In this section, we will focus more on the text retrieval aspect of the information retrieval field. Text retrieval, or document retrieval, has three distinct activities: indexing, searching and ranking. Indexing refers to the methods, or means, used to represent documents for retrieval purpose. Searching refers to the process that examines documents (or their representation) and attributes these documents to a search query. Finally ranking refers to the process of ordering documents retrieved with respect to their relevance to the search query. This section will examine these three activities for different document retrieval techniques.

Information Retrieval systems are evaluated based on effectiveness measures. Effectiveness is typically characterized by three statistics: Precision, Recall and Fallout. In a document retrieval system, after a search is completed, the document collection is divided into two groups consisting of the documents that are retrieved and the documents that are omitted. Each group is subdivided into those documents that are relevant to the search query and those that are not relevant. Figure A.2 shows a table presenting these groups of documents and defines precision, recall and fallout.

A good information retrieval systems minimizes c and b (silence and noise) and maximizes a and d (selection and rejection).

Precision is the fraction of the retrieved documents that are actually relevant to the query. A system usually tends to maximize the precision. It is obvious that precision could not be enough to evaluate an information retrieval system. Indeed, it is easy to maximize the precision by simply retrieving only one document and having that document relevant. The precision in that case would be 100%. The recall is the fraction of the actual set of relevant documents that are correctly retrieved by the system. In other words, the number of relevant documents retrieved from all relevant documents. An information retrieval system should also maximize the recall. It is again obvious that simply retrieving all documents in the collection maximizes the recall. All relevant documents are necessarily

	Relevant	*Non-relevant*	
Retrieved	**a**	**b**	(a+b) all retrieved documents
Not retrieved	**c**	**d**	(c+d) documents left out
	(a+c) all relevant documents in collection	(b+d) all non relevant documents in collection	(a+b+c+d) total collection

$$\boxed{\begin{aligned} precision &= \tfrac{\mathrm{a}\times 100}{(\mathrm{a}+\mathrm{b})} \\ recall &= \tfrac{\mathrm{a}\times 100}{(\mathrm{a}+\mathrm{c})} \\ fallout &= \tfrac{\mathrm{b}\times 100}{(\mathrm{b}+\mathrm{d})} \end{aligned}}$$

Figure A.2: Precision, Recall and fallout in IR.

in the collection. Therefore, precision and recall should be used together. A compromise to maximize them both should be found. When used together, precision and recall measure selection effectiveness. Because both precision and recall are insensitive to the total size of the collection, fallout is used to measure rejection effectiveness. Fallout is the fraction of the non-relevant documents that are retrieved. An information retrieval system should minimize the fallout. The calculation of recall requires knowledge of the total number of relevant documents in the collection (a+c). In a large collection, this is not always practical and even impossible. Often, the number of not retrieved relevant documents (c) is estimated. Sampling techniques and other methods like pooling are used to estimate this number.

It is important to note that precision, recall and fallout depend upon the definition of relevance. The relevance of each document-query pair is rated individually. Relevance rating is obtained by experts or users. Thus, this measure is highly subjective and represents a user need. To avoid a user-centered evaluation, a population of users (or experts) might be used. Nevertheless, with the same collection of documents, the same queries, and the same relevancy, precision, recall and fallout can be used to compare different information retrieval systems performance. In addition, time performance can also be used for system evaluation.

A.1.1 Conventional Document Retrieval

The most obvious way to look for documents that match a query string is to examine all documents one by one, and search for the string [148, 36, 240]. While this brute force method guarantees to locate all documents containing the exact string, it is certainly not scalable, and is redundant. However, this method generates no space overhead since no representation for documents is used except the documents themselves.

It was clear for the information retrieval research community that indexing was key for successful information retrieval systems. Researchers have devoted a lot of attention to the development of automated indexing techniques. A myriad of techniques have been implemented and tested. Some of them are still at the heart of modern techniques used for resource discovery from the Internet. As will be outlined in the following sub-sections, indexes differ significantly from one information retrieval method to the other. Each method has its own indexing technique and index structure. Indexes may contain terms, like for the inverted file technique, or a representation of some sort of these terms. Bit strings constitute the indexes for some information retrieval techniques, where a particular bit may indicate the presence or the absence of a term, for example. Not all words of a document are indexed. Words that occur with a high frequency are usually eliminated before the indexing process. A list of these words, called a stopword list[261], is consulted for each word encountered during the document indexing, or during the query parsing. Indeed, commonly occurring words are also discarded from queries when these are submitted in natural language.

Other means are also used to reduce the set of words for inclusion in the index. For example, plurals are converted to singulars. Different language normalization techniques borrow from natural language processing to reduce words with similar meanings to a generalized common concept. These are called conflation techniques. Stemming, for instance, reduces all words with the same root to a single form by stripping the root of its derivational and inflectional affixes (i.e. suffix, prefix, and infix). That is, all words are transformed to their canonical form. In the English language, stripping the suffixes usually suffices. Conflation techniques are usually language dependent and are often based on dictionaries, like dictionaries of common word endings, dictionaries of synonyms, etc. In general, there are three classes of word normalization:

1. morphological stemming: For example, for the term “retrieving”, the stem “retriev” would be extracted.

2. lexicon-based word normalization: For example, “retrieval” would be reduced to “retrieve”.

3. term clustering into synonymy classes or subsumption hierarchies: For example, “retrieve”, “recover”, “fetch”, “bring”, etc. could be grouped in the same class.

Word normalization not only reduces the size of the indexes, but it has also been proven to improve the selection effectiveness of information retrieval systems. Not all information retrieval systems use these natural language techniques for word normalization. Some systems are notorious in choosing to only eliminate some stopwords like articles and pronouns. For simplicity, they index the whole full text document content. This increases the size of the index and, more importantly, may decrease the recall.

When systems include phrases in their indexes, the phrase recognition phase is processed before stopwording and normalization in order to keep phrases in their original form. Many alternatives are used for phrase recognition; some based on syntactic recognition, some on word co-occurrence statistics, and some on manually built phrase dictionaries. Often a combination of these approaches is used.

Information retrieval systems normally involve the following:

1. Documents (or other records) are processed to record features about them that will assist in the matching with the queries. This is the indexing which either associates terms with the documents (keywords and phrases extraction) or generates a document representation.

2. Queries are processed in a similar manner as documents to extract features associated with them, such as phrases.

3. The document features and the query features are compared in such a way as to separate the documents that are relevant to the query from the others.

4. An ordering of the selected documents is prepared according to the relevancy.

In systems with relevance feedback, another step may be involved where the user may mark the relevance of the documents displayed and resubmit the ranking to the system. Using the judgment submitted, the system modifies the original query and re-evaluates it. Because it is difficult to formulate queries in a retrieval system when the document collection is unknown, relevance feedback yields progressive refinement and reformulation of queries. Reference feedback significantly improves retrieval performance [217].

Traditional information retrieval approaches used in library science are signature file methods and clustering methods which index documents by generating bit strings representations. While these systems are not as popular today as they were, hybrid variations of the technology still subsists. By and large, inverted file index is the most commonly used approach in information retrieval systems. These methods are outlined in the following subsections. Artificial intelligence-based methods like neural networks and genetic algorithms are not covered in this survey.

Document Clustering

The idea behind document clustering is that similar documents tend to be relevant to the same request. By grouping similar documents into clusters, the search space can be reduced and the search accelerated. Document clustering involves cluster generation, which is the indexing process, and the cluster search which is the query processing and matching. Clustering techniques operate on vectors. Each document is represented as t-dimensional vector, where t is the number of selected index terms. From all documents in the collection, t terms are chosen and are represented in the vector. "0" in the corresponding position in the document vector indicates the absence of a term in a document. The presence of a term is indicated by "1" or by a term weight which is either the occurrence frequency of the term in the document or a calculated term weight based on a relative occurrence of the term in the whole collection[89]. The vector model yields a representation of the documents in a t-dimensional space. Each document becomes a point in this space. Partitioning these points into groups constitutes the cluster formation. There have been many cluster formation methods proposed. Most are classified into three types of methods: similarity-based methods, iterative methods, and hybrid methods.

Methods based on similarity matrix[218]: These methods apply graph theoretic techniques and require a similarity function that measures how closely two documents are related. Many such document similarity measures have been proposed in the literature. The methods of this class are basically the following: Connect by an edge the two points representing two documents if the two documents have the similarity measure exceed a given threshold. When all documents are compared, the maximal cliques (i.e. the connected components) in the resulting graph are proposed clusters. If n is the number of documents in the collection, these methods require $O(n^2)$. Other methods using graph theory and requiring quadratic time have also been proposed[269].

Iterative methods[218]: These methods do not require a document-document similarity function, but they necessitate empirically determined thresholds like the number of clusters desired, the size of clusters, etc. Many iterative methods were proposed. They usually operate in the order $O(n \log n)$ or $O(n^2/\log n)$. The general approach is to determine an initial partitioning and iteratively re-assign documents to clusters until there are no assignments that can improve the clusters based on a document-cluster measure. These methods can allow overlap between clusters.

The same document clustering can be applied to terms. Co-occurring terms are usually related or relevant to each other. This grouping of terms into clusters is very useful in dimensionality reduction. Clusters are represented by concepts that can be used in the t-dimensional vectors as term representatives. Using concepts instead of terms reduces the keyword dimension.

Hybrid methods: These methods combine similarity matrix and iterative methods. Salton and McGill, for example, present in [218] a method that uses an iterative approach to generate rough partitioning of documents and then uses a graph-theoretic method to subdivide each partition into smaller clusters.

The cluster search starts by processing the query and representing it in a t-dimensional vector. A cluster to query similarity function is required. This function compares the query vector to the cluster centroids. The search is done only in the clusters which have a similarity with the query vector exceeding a certain threshold.

The vector model of the clustering method allows relevance feedback. After relevant documents among the retrieved ones are marked by the user, the system reformulates the query vector and restarts the cluster search. The effectiveness of the search has been shown to increase after two or three iterations.

Clustering techniques are still attracting much interest. In [163] Lee presents two similarity based approaches to solve the sparse data problem in clustering methods. The techniques applied for word classification in natural language processing are also used for document clustering and speech recognition error-rate reduction.

Document signatures

In the signature file document retrieval method, each document yields a bit string called a signature or filter. Signatures of all documents are collected in a signature file to form the index. Signatures are either stored sequentially in the same order as their corresponding documents in the document file, or are stored with a pointer to their corresponding document. A document signature is generated by transforming words in a document into bit strings using hashing, and superimposing or concatenating the coding. By performing similar transformations on the query words and comparing the resulting bit string to the document signatures, it is possible to determine the documents that contain the query words. Being smaller than the document collection, the signature file can be searched much faster. However, since hash functions are not precise, this method can generate noise (i.e. false positives: documents retrieved that are not relevant to the query).

The signature file method is more efficient when the distribution of the “1”s in the signatures is uniform. Hence the importance of selecting a good and effective hash function. Many techniques were proposed aiming at a uniform distribution in signatures[89]. Variations in the signature file method are based on the choice of hash function and the approach adopted for combining the word signatures into a document signature.

Stiassy proved that, for a given signature size, the probability of false positives occurring is minimized if the number of “1”s in the document signatures is equal to the number of “0”s[235]. Indeed, if there are too many “0”s there is space wasted in the signature, while if the number of “1”s is too high, most query-document pairs would match. To have each bit convey optimal information, it is necessary to have half the bits set to “1”. In [33] Bloom automatically selects a size for the document signature (bloom filter) that conveys a probability of 50% for a bit in the filter to be “1” or “0”. The size of the filter is calculated based on the distribution of numbers of words per document in the collection and the number of transformations performed on each word.

Two-level signature files as well as tree structured document signatures, and a clustering signatures approach have been proposed to improve the search speed of signature file techniques[89].

Despite its low accuracy rate (compared to other methods) the signature file method is popular because of the low space overload required. However, the method has another major drawback. Since the entire signature file has to be consulted for each query, the method is expensive in terms of disk input/output (I/O) operations when the number of documents is large. To reduce the I/O cost for retrieval, Roberts suggests in [212] to store signature files in "bit slices". Instead of organizing the signature files in rows of signature bits one signature per row, the file is organized one signature per column. In other words, the first bit of all document signatures are stored consecutively in the first row, then the second bit of all signatures in the second row and so on. This structure in a random access file can reduce the I/O operations drastically. However, while the updates are easy in the normal signature files, bit slice signature files are very costly in maintenance. Each time a document is updated or added, the whole signature file has to be changed.

Inverted Indexes

Just as the relational model is the most used model in commercial databases, inverted indexes are followed by most commercially available information retrieval systems [218]. An inverted index (or inverted file) is a file containing every term in a set of documents with the list of documents where they appear. The file is ordered in alphabetical order and each term appears only once. Terms can be words, compound words or phrases. In practice, terms are stored sorted in the inverted file along with a pointer to a posting file which contains the document accession numbers of the documents containing them. Figure A.3 shows an example of such structure: the term "retrieval" appears in four documents pda, pdb, pdc, and pdd. A pointer points to a record in the posting file where pda pointer is stored. pda, pdb, pdc and pdd pointers are stored consecutively in the posting file. They all point to the appropriate documents in the document file.

This structure is easily built by going over the document collection once and cumulating counters for selected terms. The search process simply locates documents containing a specific term by retrieving the document pointers in the posting file pointed by the term entry in the inverted file. By using a B^+tree structure on top of the sorted inverted file, the search process becomes extremely fast. Moreover, the inverted file structure permits easy processing of boolean expressions by using set operations on the document sets of the terms in the expression. For example, a simple intersection between a document set pointed by a term A and the set pointed by a term B would yield the documents satisfying A and B. The union of such sets yields the documents satisfying A or B. The not operation can also be processed in the same manner.

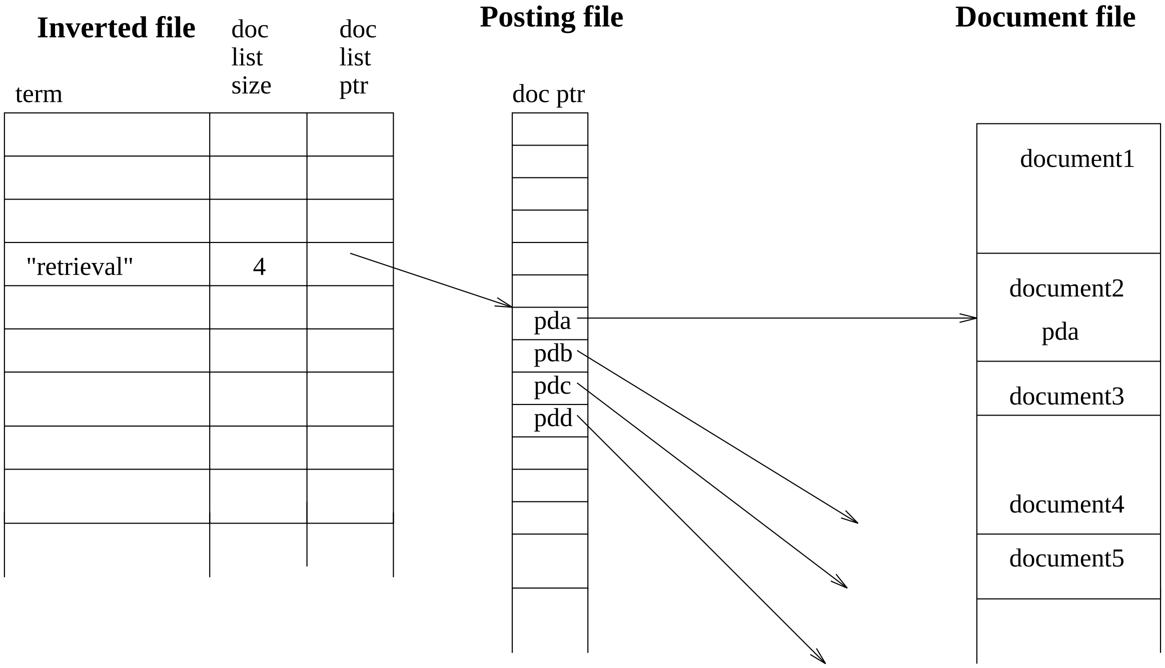


Figure A.3: Inverted Index File structure.

The most significant problem with inverted file structure is the storage overhead. Typically, an inverted file structure requires 50% to 300% of the original document collection size [131] depending upon how terms are chosen (words, phrases, overlapping phrases, synonyms etc.). Another disadvantage of this method is the maintenance cost. Adding new documents involves updating many document lists in the posting file, while adding new term may necessitate costly B$^+$tree structure updates. However, the easy implementation and the speed of the inverted file outweigh the disadvantages, making this method by and large the most popular, even in today's World-Wide Web search engine applications.

The simple structure of the inverted file can convey additional information that can be useful in the ranking of the documents retrieved. Relevance ranking of documents is usually based on heuristics using the number of terms from the query that appear in the documents, and the popularity of the terms in the collection. This information can be obtained from the inverted file structure. An enhanced information file structure has an additional file holding the positions of the terms in the documents. A change in the posting file schema allows adding pointers to the posting file. Figure A.4 shows such a structure. The term "retrieval" is used in four documents. A pointer from the inverted file points to a record in

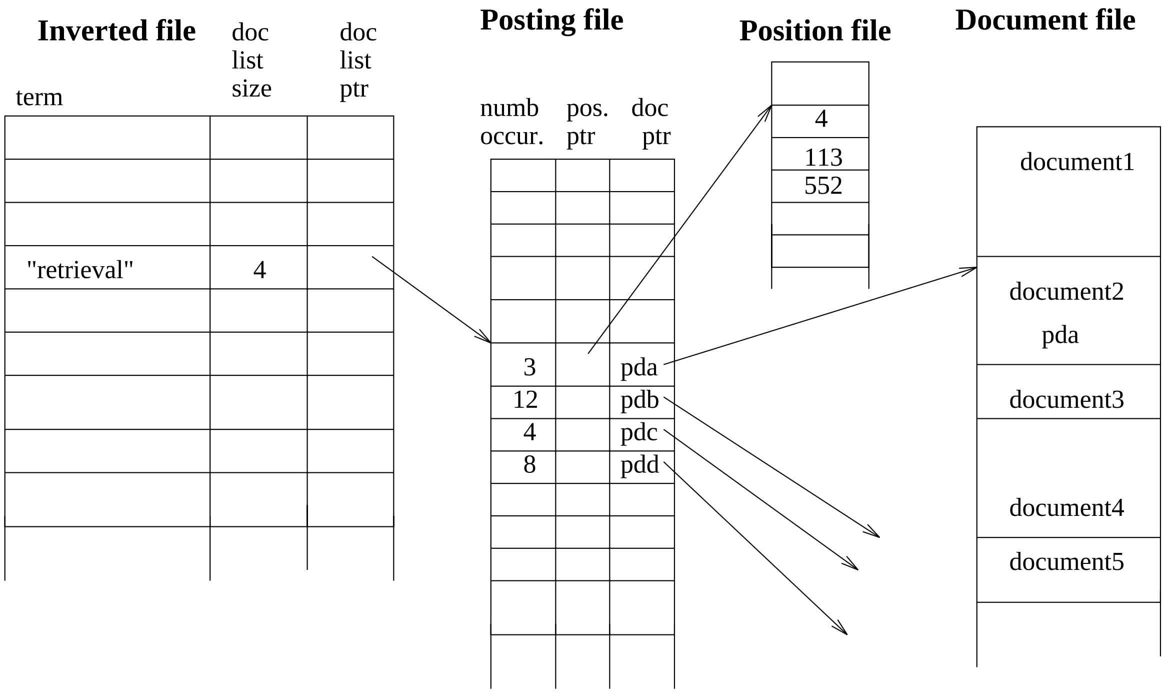


Figure A.4: Inverted Index File structure with position file.

the posting file which is the first of four consecutive records like in the structure in Figure A.3. A record in the posting file has a pointer to the corresponding document but also the number of occurrences of the term in the document and a pointer to a list of positions of this term in the document. The list is stored in the position file. For instance, “retrieval” appears three times in the document pda in positions 4, 113, and 552. With this new structure, the number of occurrences of a term in a document can be used in the relevancy of a document, and the relative position of search term in a document can also be used to rank the relevance of the document. Moreover, the proximity of terms (i.e. how close two terms are to each other) can be used in retrieval queries.

Term Weight Assignments

Terms in a document are not equally useful in content representation. Some terms are more important than others. Giving weight to terms according to their importance in the document can help rank documents according to their relevance to a query, hence, the term weighting approach. In order to assign high weight to terms deemed important and low weight to less important terms, frequency of terms in a document with respect to their frequency in the whole collection can be used. In other words, terms that are frequent in a document but do not appear in other documents are important, while terms that are repeated in all documents are not important. For a term t in a document D_i, its weight is equal to the term frequency in D_i times the inverse of the frequency of the term in the collection ($w_it = f_t(D_i) \times \frac{1}{\mathrm{f_t(C)}}$). This definition of weight favours terms with high frequency ($f_t(D_i)$) in a particular document D_i but with a low frequency overall in the collection ($f_t(C)$). When all documents in the collection are represented by weighted term vectors of the form $D_i = (w_i1, w_i2, ..., w_in)$ where w_it is the weight of the term t in the document D_i and n the number of distinct terms, a similarity measure can be calculated for each pair of vectors. This measure reflects the text similarity between the two documents. If queries are processed in the same manner and a weighted vector is generated, given a query Q_j, a similarity computation of the form $Similarity(D_i, Q_j) = \sum_{k=1}^{n} w_ik \times w_jk$ can be computed for any document D_i in the collection. Given this similarity function, a ranked list of documents in decreasing order of similarity can be obtained for any query or sample document.

A.1.2 Hypertext and multimedia

Because major resources on the Internet reside on the World-Wide Web and the World-Wide Web is using the hypertext paradigm, indeed, the World-Wide Web started as a hypertext project, it is important to provide a glimpse of the hypertext development.

Hypertext basic concepts were introduced by Ted Nelson, who coined the term "hypertext" in 1965 [188]. Following Vannevar Bush's Memex idea [44], Ted Nelson developed the Xanadu project[1] which aimed at placing the entire world's literary corpus on-line. Hypertext systems, also known as non-linear text systems, provide non-sequential access to information by incorporating the notion of navigation, annotation, and tailored presentation [31]. Hypertext has been defined as "an approach to information management in which data is stored in a network of nodes connected by links. Nodes can contain text, graphics, audio, video, as well as source code, or other forms of data" [230]. With multimedia, hypertext is called hypermedia.

The major original concept with hypertext is the concept of links. Links can either

[1] http://www.xanadu.com.au

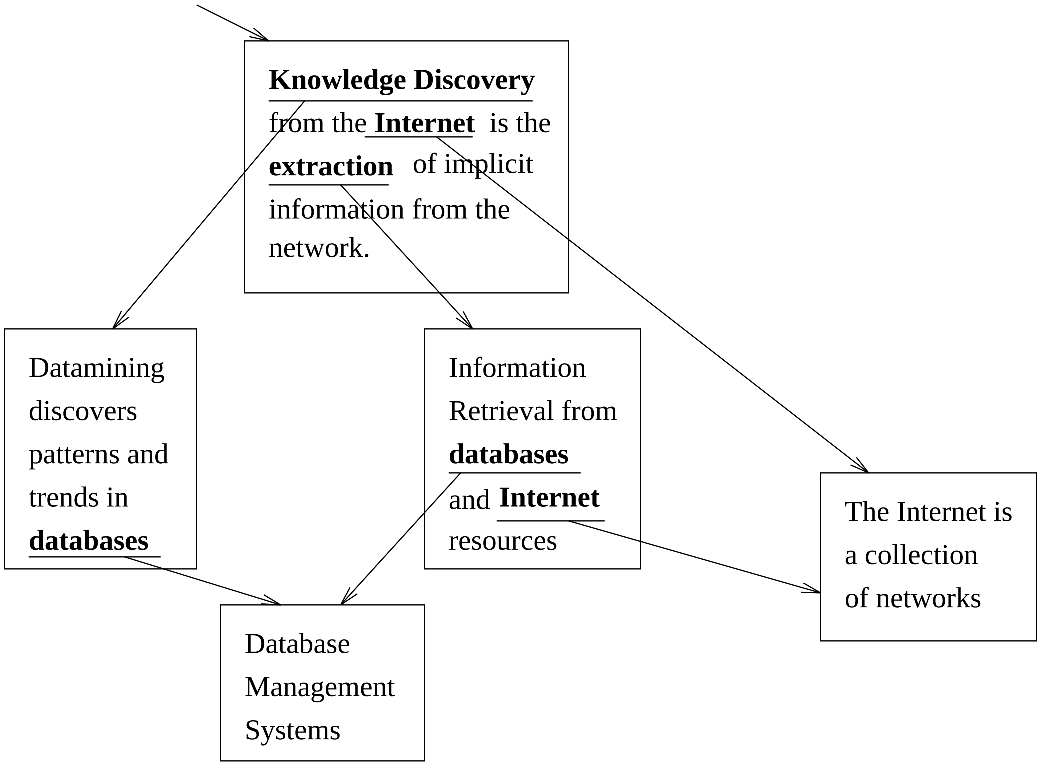

Knowledge Discovery from the **Internet** is the **extraction** of implicit information from the network.
Datamining discovers patterns and trends in **databases**
Information Retrieval from **databases** and **Internet** resources
The Internet is a collection of networks
Database Management Systems

Figure A.5: Example of a hypertext document.

be within a node, or between nodes. A document in a hypertext system is a collection of nodes interconnected by links, which can be unidirectional or bi-directional. Bi-directional links permit backward traversals of documents. It is the linking capability that allows the non-linear organization of text. Figure A.5 shows an example of a hypertext document. The node from which a link originates is called the reference, and the node at which the link points is called the referent. The starting point of a link is referred to as an anchor point, or a button. When reading a document, a user is presented with nodes which can be considered as document parts conveying the same theme. By clicking an anchor, the associated link is traversed, taking the user to the associated node. Obviously, the user interface is paramount in hypertext systems for navigating through large amounts of information. Authoring documents with such systems have been known to be somehow problematic, causing cognitive overhead, referred to as cognitive task scheduling problem. Authors need an additional effort and concentration to maintain several tasks and keep track of different links at one time[59]. User interface issues and hypermedia authoring issues have drawn much attention in the research community [112, 59, 31, 230]. The major information retrieval approach used for such a model is a query-less approach using simple browsing - navigating the document by traversing it node after node following its links. Browsing, however, has a major drawback: disorientation. Users have the tendency to lose their sense of location and direction in a non-linear document. This is known as the “Lost in Space” problem. Some aspects of this problem can be addressed by the user interface offered by the browser. Much effort has been put into this issue by Human Computer Interaction and User Interface researchers. Another solution for coping with disorientation is a query mechanism. By looking at nodes as records, or documents, as in the traditional information systems, standard information retrieval indexing techniques can be applied. Usual queries with boolean operations combining keywords allow a user to locate interesting nodes. Links, however, can give additional information that ranking heuristics can exploit for node relevance ranking.

A.2 Survey on Resource Discovery on the Internet

Many consider the moon landing to be the most significant event in our time, however, the advent of the Internet had, has, and will still have a greater impact on our society. Historians will not remember the 20th century as the atomic bomb century but rather as the century of the birth of the global communication network. The atomic bomb certainly struck an immense reaction among scientists, politicians and the *homo qualque*. The emotional impact on our society is unprecedented. However, hopefully, this means will never be used again and will be remembered as a bad experience in our civilization. On the other hand, the Internet and its collection of services will continue to grow to a level we cannot even imagine today. Already the Internet is a necessity we can not afford to lose. We use it to interact, learn, communicate, and entertain.

Many agree that the Internet started as the ARPANET, an initiative of the Advanced Research Projects Agency, during the Cold War in 1969. By building the ARPANET, the US Department of Defense wanted to explore the possibility of a communication network that could survive a nuclear attack. Initially the network connected four sites, but very rapidly, many research centres and universities were connected. The introduction of the communication protocol TCP/IP in the early 1980s helped interconnect various research networks, which resulted in the Internet with ARPANET as a backbone. During the same period, the National Science Foundation (NSF) established six supercomputer centres in the United States. In 1986, a dedicated network (NSFNET), funded by the NSF, connected these centres and became the new Internet backbone when ARPANET was dismantled. Since then, the growth in the number of hosts and the growth in packet traffic on the backbone have been increasing exponentially [185, 283]. The table in Figure A.6 and the graphs in Figures A.7, A.8 and A.9 show the growth of the Internet in host numbers, domain numbers and web site numbers. The data was collected and compiled from various sites on the Internet, among them [185, 283, 237]. The growth of the Internet accelerated even more with the advent of the World-Wide Web and the connection of commercial TCP/IP providers. Along with the growth of the Internet, there has been an important increase in the number of software tools to make use of the multitude of resources in the network. Moreover, different transfer protocols have been adopted each creating a “subspace” of resources within the Internet. File Transfer Protocol (FTP) for instance, which was the most frequently used service on the Internet until 1994 (in terms of data packet transferred), has a subspace of more than 3 million documents estimated at a few tera-bytes scattered on a few thousand sites[70]. Other subspaces like gopher- space, USENET space and World-Wide Web space contribute to the ever-growing size of the Internet. It is interesting to note that many attempts have been made to index on-line resources, all of which aim at one Internet subspace or another. Archie for example indexes FTP sites, and Veronica indexes Gopher space, while search engines cover the World-Wide Web. Most of recent World-Wide

Web indexes include USENET subspace and/or FTP or Gopher as well.

A.2.1 Internet tools for information Retrieval

This section outlines some of the widespread networked information retrieval tools and services. The number of tools available today is very large and still growing very quickly. The following classification is one possible ordering of some of the most popular services on the Internet aiding in the proliferation of resources or the retrieval of those resources. There are four categories of networked information retrieval tools or services. The first category encompasses communication services like electronic mail, news groups, telnet, synchronous chat tools, etc. The second category groups information storage and information exchange services, like FTP, Gopher, Alex, etc. Information indexing and information retrieval services, like Archie, Veronica, WAIS, WHOIS and Netfind, are in the third category. Finally, the interactive multimedia information delivery service, namely the World-Wide Web and its indexes, is in the fourth category.

Communication Services

Electronic mail (e-mail): E-mail has been one of the main uses of the Internet and was one of its first applications. E-mail is a way to send messages from a user on a computer to a recipient user on a destination machine, both user machines being connected via a network. The message can contain any text such as a business memo or a personal letter. Today, e-mail can contain multimedia like sound, images or video. Although e-mail is still one of the major uses of the Internet, the documents generated by this service are usually private and do not contribute largely to the proliferation of public on-line resources on the Internet. However, it is common, on corporate Intranets, to archive office memos exchanged by e-mail. Such archives or personal electronic mailboxes grow very rapidly and necessitate information retrieval techniques to locate particular e-mail messages, or even data mining techniques to summarize message contents.

USENET (network news groups): USENET is a collection of hosts that receive network news groups, which are discussion groups or forums about a variety of topics. Network news is a mechanism for broadcasting messages from one host to a large number of hosts. A message or article sent to a news group is received by a host on USENET. People who access the USENET hosts can read the messages. This connection simulates the message

Date	*Hosts*	*Domains*	*Web sites*
09/1969	4		
04/1971	23		
06/1974	62		
03/1977	111		
08/1981	213		
05/1982	235		
08/1983	562		
10/1984	1,024		
10/1985	1,961		
02/1986	2,308		
11/1986	5,089		
12/1987	28,174		
07/1988	33,000		
10/1988	56,000		
01/1989	80,000		
07/1989	130,000	3,900	
10/1989	159,000		
10/1990	313,000	9,300	
01/1991	376,000		
07/1991	535,000	16,000	
10/1991	617,000	18,000	
01/1992	727,000		
04/1992	890,000	20,000	
07/1992	992,000	16,300	50
10/1992	1,136,000	18,100	
01/1993	1,313,000	21,000	
04/1993	1,486,000	22,000	
07/1993	1,776,000	26,000	150
10/1993	2,056,000	28,000	
01/1994	2,217,000	30,000	650
07/1994	3,212,000	46,000	3,000
10/1994	3,864,000	56,000	
01/1995	4,852,000	71,000	10,000
07/1995	6,642,000	120,000	25,000
01/1996	9,472,000	240,000	100,000
07/1996	12,881,000	488,000	300,000
01/1997	16,146,000	828,000	650,000
07/1997	19,540,000	1,301,000	1,200,000
01/1998	29,670,000	2,500,000	2,450,000
07/1998	36,739,000	4,371,603	

Figure A.6: Growth of the Internet from 1969 to 1998. (Compiled from [185, 283] and other sites)

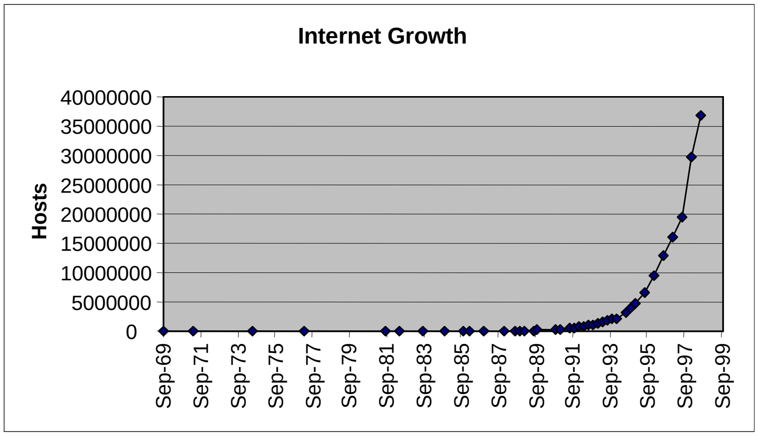


Figure A.7: Growth of the Internet in terms of number of unique hosts.

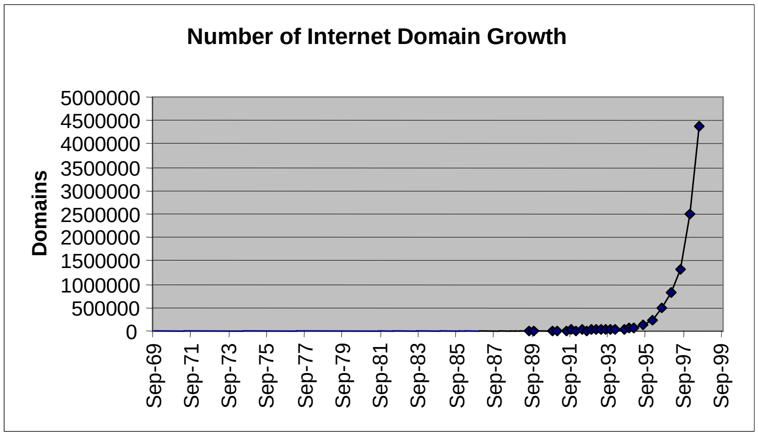


Figure A.8: Growth of the Internet in terms of number of unique Internet domains.

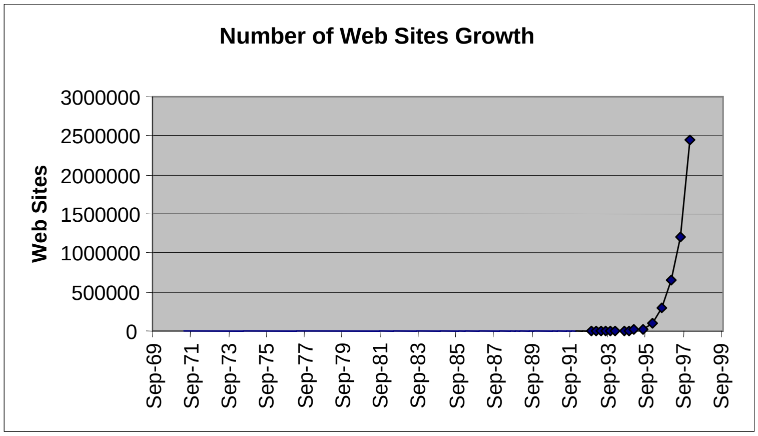
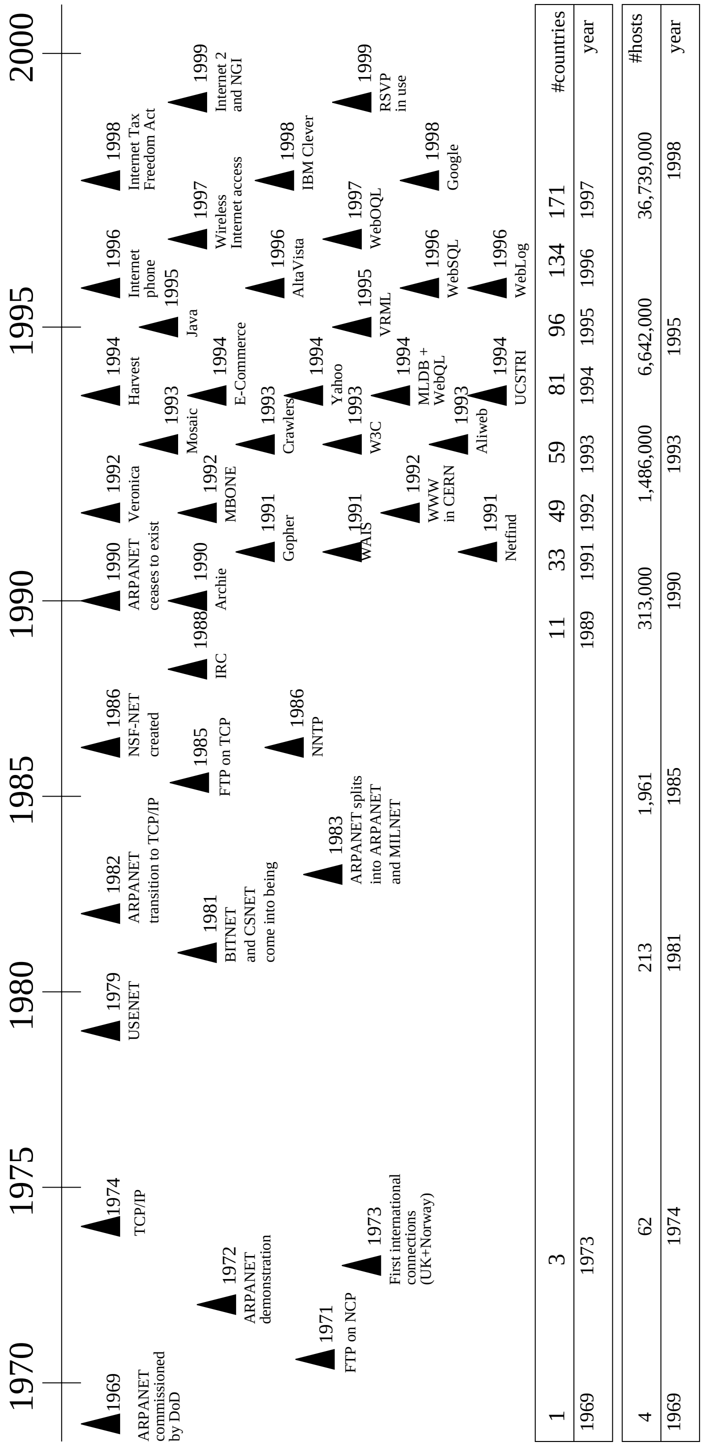


Figure A.9: Growth of the Internet in terms of number of Web Sites.



#countries	1	3	11	33	49	59	81	96	134	171
year	1969	1973	1989	1991	1992	1993	1994	1995	1996	1997

#hosts	4	62	213	1,961	313,000	1,486,000	6,642,000	36,739,000
year	1969	1974	1981	1985	1990	1993	1995	1998

Figure A.10: Timeline of the Internet.

broadcast. People who access news groups can browse and read articles, post new messages, or reply to particular messages. News group messages are archived and are accessible on-line. There are thousands of news groups on different topics from scientific or technical, to political or esoteric. The large collection of articles posted daily on USENET contributes significantly to the rapid expansion of the size of the on-line resource accumulation.

Internet Relay Chat (IRC): Real-time textual chatting has been very popular since the early days of the Internet with Unix facilities like "talk". This service allows real-time interactive character-based communication between two or more remote users. IRC sessions are usually not archived. However, in a business context, chat session may be saved on a corporate Intranet for later retrieval and reference.

With the growing popularity of the Internet, thanks to the World-Wide Web, new communication services are added and enhanced regularly. Newcomers like net telephony, net fax, and video conferencing, will significantly contribute to the large number of on-line resources.

Information Storage and Information Exchange Services

Anonymous FTP: File Transfer Protocol (FTP) allows the access of resources, mainly files, on remote computers. An FTP server provides a portion of its file directory structure and allows exchange of files. Before the World-Wide Web became widely used, anonymous FTP archive sites (i.e. publicly accessible FTP sites) were the most widely accessible source of information on the Internet. Each FTP site usually offers files related to one or more topics of interest. These files can be document files, executable programs, data files, or any file a computer can store. FTP archive sites are usually maintained as a volunteer effort. Each file directory accessible by anonymous FTP, contains a "readme file" explaining the content of each file in the directory. There are no standards about the content or the structure of the readme file or even the name of the file[2]. Usually, free text is used to compendiously describe the files. Thus, it is necessary to manually browse the FTP server directory structure and read the readme files in order to find sought for resources. Techniques like Archie[69] and UCSTRI[250] were implemented to help find resources in anonymous FTP archive sites. Archie and UCSTRI are described later in the following subsection.

An analysis of the files available at FTP sites was presented in [70]. On 1044 anonymous

[2] Files can be called "index", "readme", "read.me", "dir", etc.

FTP sites, there were 3.1 million files, 44% of which were textual documents. Clearly, anonymous FTP and FTP sites contribute to the wealth of information available on the Internet. Today, there are thousands of FTP sites with a total of a few million files, and roughly a tera-byte of data.

Gopher: The Internet Gopher[255] is a distributed document delivery system originally developed in an effort to provide the University of Minnesota students and staff with a flexible campus-wide information system for disseminating news, announcements, and other kinds of information to the university community. Its simple single menu-driven interface made it very popular and helped its rapid adoption by many sites across the Internet. Gopher evolved from a system primarily intended to distribute documents to an environment for providing access to many different types of files, and network services through gateways. Despite its initial popularity and the proliferation of Gopher documents, gopher space is shrinking and is gradually being "translated" to World-Wide Web documents. Veronica, an indexing and retrieving system for gopher documents, has been developed. Veronica is presented in following subsection.

Alex: Alex[13] is a file system that provides transparent read access to files in distributed anonymous FTP sites on the Internet. Rather than accessing FTP files by logging in to remote hosts and copying files locally, Alex allows the user to see FTP files as part of the local file system. Transparently, file directories of FTP sites are "mounted" to the local file system with */alex* as root directory. The Internet domain of an FTP site is coded as sub-directories in the Alex structure. The Simon Fraser University FTP site for example would be */alex/ca/sfu*, while FTP Berkeley would be */alex/edu/berkeley*. Local Unix tools like *grep* or *find* can be used to find and retrieve documents from remote sites without having to make local copies or keep track of remote file updates. Any document or file put into any anonymous FTP site becomes available via Alex.

Information Indexing and Information Retrieval Services

Archie: Until 1995, when the World-Wide Web became the most used service on the Internet in terms of data packets transferred on the NSF backbone, FTP was the most used service on the Internet. FTP still accounts for a large amount of data transferred on the Internet. Hundreds of FTP sites offer reports, documents, raw data, images, programs, etc., however, in order to find a file, one has to know the FTP site where the file might be archived then browse the directory structure of the FTP site, download readme files and eventually find the file sought for. Archie is an electronic directory service implemented at McGill University by Peter Deutsh, Alan Emtage, and Bill Heelan[69], that allows locating files archived in anonymous FTP sites. The automatic cataloging system periodically retrieves listings of file names from anonymous FTP sites by recursively browsing directories via anonymous FTP. Listings are combined into a searchable database which can be accessed from the Internet via telnet, Archie clients, or various information system gateways. Archie database contains only file names, modification dates and FTP sites and paths where files can be found. Its index is produced once a month and is mirrored in 13 sites to reduce traffic. The problem with Archie is that only file names are indexed. This means that in order to locate a file, its name or portion of its name has to be known. It is not possible to find a file with Archie when only a description of the file or the file topic is known. Despite its limitations, Archie has been extremely popular, and new World-Wide Web Archie-like systems have been implemented like FTPSEARCH[3] or CNET[4] which provide indexes of free software on FTP and World-Wide Web servers organized by topic.

Veronica (Very Easy Rodent-Oriented Net-wide Index to Computerized Archives): The Internet Gopher system provides a simple menu-driven user interface. Its protocol is a simple client-server stateless protocol. When a server gets a request it returns a document or a menu and closes the connection. This makes an automated traversal of gopher space easy. When searching for a document manually, one has to browse a hierarchy of menus to eventually locate the resource. Veronica, developed at the University of Nevada, makes it easy to search for items in gopher space by title. Veronica periodically traverses gopher space by recursively requesting menus from different Gopher servers starting from a set of registered menus. It indexes all titles in each text menu received. A search with a term results is a menu with links to known gopher menus containing the search term as a menu entry. Veronica has been very popular among Gopher users, but since gopher space is fading, Veronica is becoming useless.

WAIS (Wide Area Information Servers): WAIS, developed by Thinking Machines Corporation, is a system that allows users to deploy, search and retrieve documents and different types of resources from indexed databases. WAIS was developed in collaboration with Apple Computer and Dow Jones initially for use by business executives. In contrast to Archie

[3] http://ftpsearch.ntnu.no/ftpsearch

[4] http://www.cnet.com

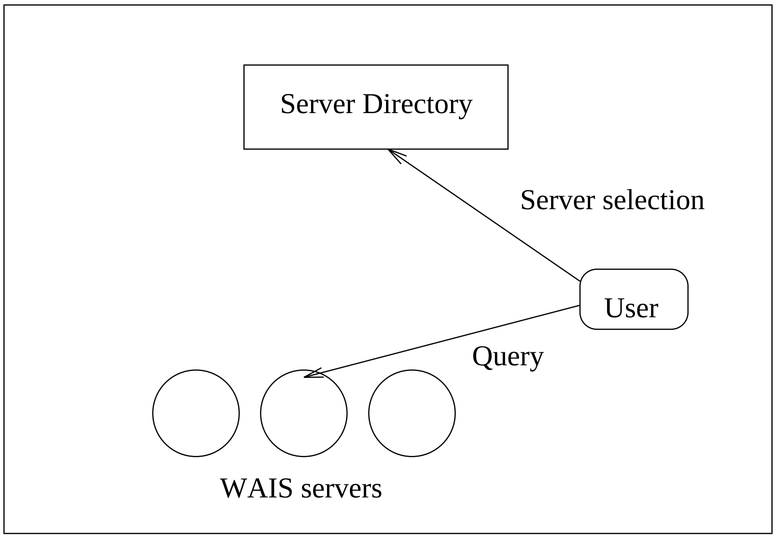


Figure A.11: WAIS general architecture.

which indexes only file names, WAIS indexes contain keywords from the content of textual documents, bibliographical databases, and even descriptors from graphical files. While Archie or Veronica centralize their index in a single global index which is mirrored, WAIS addresses scalability with a better approach decentralizing its index. WAIS divides its indexes among the servers that provide information. Each server indexes its information locally and registers its repository and index in a directory operated by Thinking Machines. The directory is a top-level index that classifies servers and provides knowledge about information available on each WAIS server. Servers are thus specialized and each usually contains resources on a particular topic. When searching for a resource, users connect to the directory of servers and select a particular server to search. Queries are submitted in natural language to the selected server. After eliminating stopwords, the conjunction of the remaining words and phrases is applied to the full text index to find relevant documents. Documents are ranked based on heuristics using word weighing algorithms. Relevant documents can be fed back to the server to refine the search (i.e. relevance feedback). Successful searches can be automatically run to alert the user when new information becomes available. WAIS is used by many systems as the underlying indexing scheme. The first World-Wide Web search engines, for example, used WAIS to index information retrieved by their spiders (see section A.2.3). Figure A.11 shows the general architecture of the WAIS approach with distributed indexes.

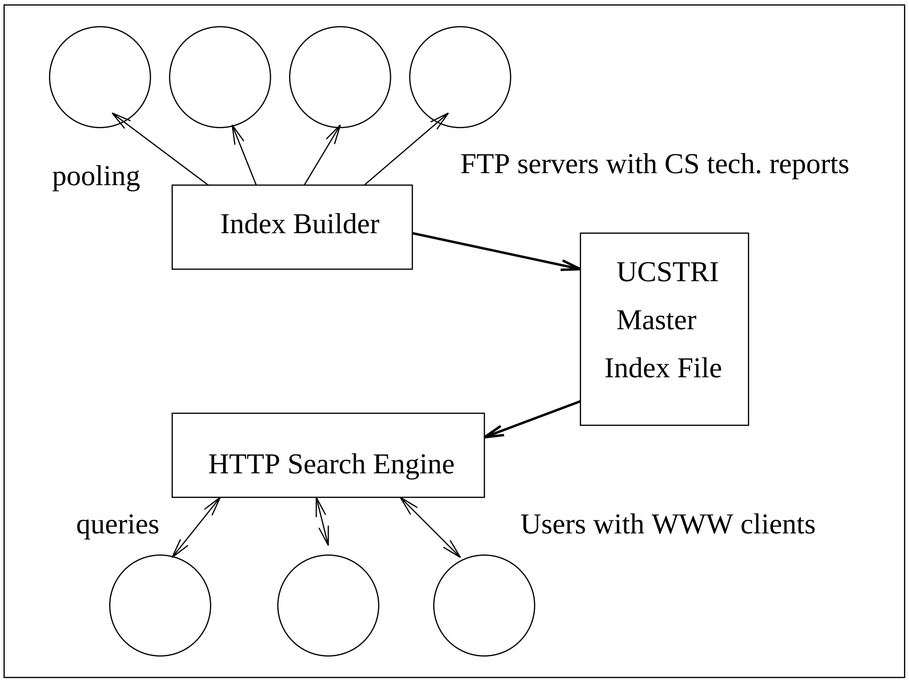


Figure A.12: UCSTRI general architecture.

UCSTRI: The Unified Computer Science Technical Report Index[250], built by Marc VanHeyningen in Indiana University in 1994, is a service which provides a searchable index over existing technical reports, theses, and other documents broadly related to computer science and stored in anonymous FTP sites. UCSTRI index pulls information from a large number of registered anonymous FTP sites each with its own format for the readme files. UCSTRI regularly visits the registered FTP sites and automatically downloads the readme files it finds. All these readme files with different free text formats are parsed and merged in a large master index. 6,000 technical reports are indexed by UCSTRI from about 120 different FTP sites. UCSTRI uses file names and descriptions found in the readme files to index technical reports and academic papers. The system has the advantage of being completely automated. However, the maintenance of the index is cumbersome because readme files are written in free text. Moreover, the name of the readme file differs from site to site and can be named README, read.me, dir, about, etc. Each time a new FTP site is added to the registered FTP site list, the UCSTRI index builder has to be rewritten or updated in order to take into account the structure of the readme files of the new site. Figure A.12 shows the general architecture of the UCSTRI system.

Netfind: Netfind is an Internet directory service developed by the Networked Resource Discovery Project at the University of Colorado. Despite the fact that Netfind does not locate documents in the Internet, it is a service worth mentioning for the exemplary methodology used for resource discovery. Netfind attempts to locate electronic mail addresses and other information about Internet users dynamically. Rather than developing a protocol to collect and register every user on the Internet in a database, which would be very difficult to maintain, the Netfind approach is to use already existing network services to locate likely machines where the sought for user might reside and address these machines with a finger service. This strategy has been adopted by many intelligent agents on the Internet (see section A.2.4). Netfind regularly browses USENET archives and other services to retrieve unique e-mail addresses and build an Internet domain-based hierarchy of addresses that help future searches. A search query is submitted by providing the name or login identification, and some keywords describing the institution or location where the user sought for might reside. Netfind uses a set of heuristics to locate hosts on which the desired user may have an account or mailbox. The query can be refined by the user selecting among the different locations Netfind guesses. The subset of domains selected by the user is searched in parallel, again taking advantage of existing network services. Netfind has been very successful and scalable despite the incredible growth of the Internet. Using existing network services and resources proved to be a viable and prevailing method of supplying a new service on the Internet.

Interactive Multimedia Information Delivery Service

The World-Wide Web is the interactive multimedia information delivery service on the Internet. Many hypertext and text retrieval systems, still in use, were available before the advent of the World-Wide Web. Researchers in hypertext were very active, but most focussed on user interface and authoring issues. Apart from the Xanadu project, no hypertext project was emphasizing wide area distribution and global system access. In 1992, an internal project at CERN in Switzerland, led by Tim Berners-Lee and Robert Caillau, attempted to make distribution of information easy for physicists working around the world and exchanging data. The World-Wide Web was born out of this project and was very rapidly adopted by Internet users for its ease of use, interactivity and multimedia support. The World-Wide Web relies on hypertext technology[27, 28]. Documents are formatted using HTML (HyperText Markup Language), and hypertext links within the documents are used to travel from current documents to others. HTML allows annotating documents using hypertext links and colours, and intermixing text with images and other media in the same document. When the World-Wide Web browser Mosaic was released in 1993, the World-Wide Web became widely used thanks to the "point and click" interface of Mosaic. No Internet service used to share information across the Internet allows simple browsing and ease of use like the World-Wide Web, which allows users to browse large collections of information across the network without having to log in or know in advance where to look for information. The World-Wide Web became the technology of choice to deploy information on the Internet and was the major reason for the tremendous growth of the Internet in terms of the amount of published information and in terms of use of the network. Figure A.9 shows the exponential increase of numbers of web sites on the Internet. Because gateways to most Internet services have been developed for the World-Wide Web, the World-Wide Web is becoming for many a synonym of the Internet. Because of the huge amount of data rapidly cumulated in the World-Wide Web space, "surfing" the web to find information became cumbersome. Finding real information is often a hit-and-miss process. While browsing to look for a particular information, a user may drift and end up reading other possibly interesting pages which may be irrelevant to the original quest, a phenomenon similar to browsing an encyclopedia. The following sections highlight the evolution of information retrieval techniques applied to the World-Wide Web.

A.2.2 Catalogues and Directories

Because the number of web pages available on-line was reasonably small at the start of the World-Wide Web project, browsing through a list of web servers maintained at the CERN sufficed in looking for on-line resources. The very large number of web servers and available web pages today prohibits this method for information retrieval from the World-Wide Web. Many started building lists of interesting links (i.e. bookmarks) and made these lists publicly available. NetServices list and NCSA Meta Index were maintained lists of references. These lists were very useful because users could access references to on-line resources without having to collect them themselves browsing the web. However, the rapid increase of web sites and web pages made these lists obsolete. In an attempt to make these lists more up-to-date, automatic collection of on-line references was introduced. The "CUI W3 catalog" at the Centre Universitaire d'Informatique at the University of Geneva was based on an automatic retrieval of references from a fixed set of documents like the "NCSA What's new list" that reported new sites on the web, or automatic sifting through news groups articles. These automatically collected lists of references to on-line resources were comprehensive and reasonably up-to-date, however, their size rapidly became too large to be a valuable resource. To make the lists more useful despite their sizes, lists were made searchable. A search key would then allow a user to find entries in the reference list that could be interesting. No real indexes were constructed but searches were processed in real time going though the list text and matching strings. The response time was acceptable, however, the dynamic nature of the World-Wide Web made these lists stale and thus impractical. Catalogues and directories built this way could not be representative enough of the wealth of resources on the web given the rapid growth of the Internet. Maintenance of these lists is too expensive and not scalable.

The only successful on-line catalogue of web pages is Yahoo[5], an initiative from two graduate students from Stanford University. Yahoo is a classification of topics built in a hierarchical tree. A node in the tree is a category visualized by a menu of sub-categories (i.e. arcs in the tree structure coming out of that node) and leaves are links to web resources. Resources are collected, reviewed and classified manually by editors. Authors submit the URL (Unified Resource Locator) of their web page (or web site) along with a title and a description of the resource to Yahoo editors. If the submission is accepted (only 33% are[237]), the reference to the new resource is classified and added to the Yahoo directory. A searchable index on Yahoo directory has been implemented. Contrary to what many might believe, Yahoo is not a search engine for web resources at large. Yahoo searchable index contains only resources catalogued by Yahoo editors. Moreover, the index does not contain keywords from the resources' content. The index entries are words Yahoo category names, resource titles and resource descriptions submitted by the authors. Apart from the classification of resources that allow the users to browse the hierarchy to find information related to a topic of interest, the originality of Yahoo is the fact that resources are not necessarily individual web pages like automated discovery indexes might have. Submissions that describe entire web sites are also accepted. The content of the web pages is never used for indexing. When users search for resources, they can either browse the directory by topic or submit terms to search. In the case of a search, matches are resources in categories containing the search terms and resources with titles or descriptions containing the search

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

terms. Documents in the result are ranked in priority based on the terms appearing in their category, their title, and finally their description.

Because of the success of Yahoo, others have tried to copy their concept and present a friendly user interface with a reviewed directory of sites. The LookSmart[6] approach, for example, is to present to the user an initial list of 10 topics. After a topic is chosen, a new classification of 10 topics is presented with the original list remaining on the screen. After a second choice, a third sub-classification is displayed, in addition to retaining the first two lists. This continues on with each further selection until a document list is displayed. This allows the user to easily see the path taken to reach the resource in a concept hierarchy like classification. LookSmart is a hybrid search engine. It allows a search on its catalogue like Yahoo, but also combines the result with the result of a search on the World-Wide Web space like other search engines.

Other popular types of catalogues or directories, are specialized directories; directories that contain only references to resources pertaining to a particular topic. For example, a directory for triathlon contains only references to web pages about triathlon or related to triathlon. These directories are either created manually (i.e. author's submission) or generated from the result of a query submitted to a search engine. Building a search engine on top of such a specialized directory creates a specialized search engine.

Rings are another type of specialized directory [104]. Resources in such directories are inter-linked in a circular list. Maintainers of web sites pertaining to a common topic collaborate to insert a bi-directional hypertext link in each web site to link to the next and previous site in the circle. Site S_i for example is linked to S_{i+1} and S_{i-1}. The last site in the list and the first site are inter-linked to form a ring. A centralized "authority" orchestrates the ring by accepting newcomers and maintaining on a server a list of links to all sites in the ring. When browsing, a user can jump from one site to the other in the ring following the links, or randomly access any site by requesting a link from the centralized list on the server. Figure A.13 shows an example of ring with six web sites interconnected.

A.2.3 Robots and Search Engines

Since simple browsing, listing, and even automatic collection of resources did not solve the problem for information retrieval on the web, automatic discovery became necessary

[6] http://www.looksmart.com

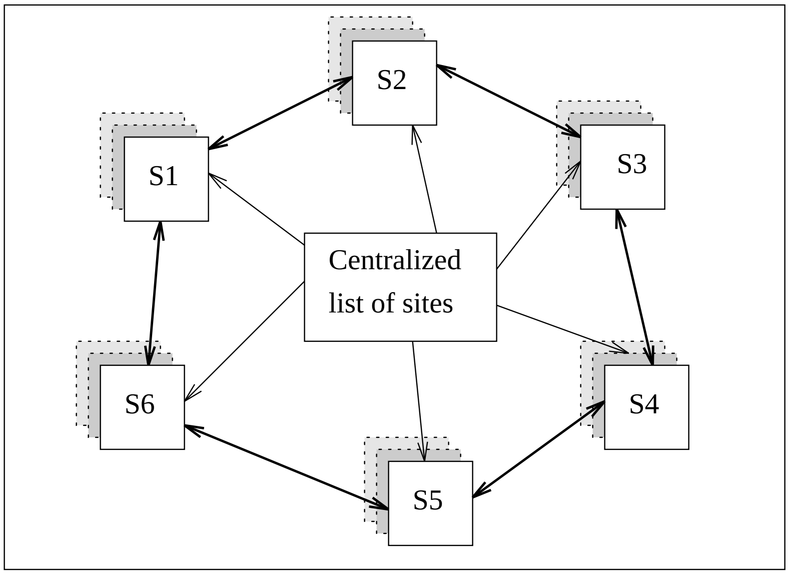


Figure A.13: Example of a ring with 6 sites.

to make effective use of this wealth of information in the World-Wide Web space. There are programs that traverse the World-Wide Web, download documents, analyze them, and store some information about them in an index that can be queried. Documents are found by exploring the graph that hypertext links form between documents. From an initial Web page, all its links are extracted and added to a queue of URLs (Unified Resource Locator). The recursive process repeatedly selects a URL from the queue, retrieves the page and extracts its links. These programs are known as robots, crawlers, spiders, or wanderers [157]. Robots may have different purposes. They can crawl the Web for indexing, filtering, mirroring documents, or for statistical intent, like calculating the size of the World-Wide Web space (it is estimated today at 1.5 TB.) or the number of web sites.

Perhaps the first academic work describing these robots is the work of Jonathan Fletcher at Stirling University in Scotland, who implemented JumpStation [98], and the work of Oliver McBryan at the University of Colorado, who developed the World-Wide Web Worm (WWWW) [179] (today the Goto.com search engine). A paper by Eichmann revealed the anatomy of RBSE spider [77], built for the Repository-Based Software Engineering Program funded by NASA. While the WWWW indexed only text that appeared in the title and headers of HTML documents, RBSE spider did a full text indexing in the hope of improving precision and recall of the system. The architecture of these systems has three major components: a spider, a parser and indexer, and the index itself. The spider is the program that traverses the known, or visible World-Wide Web and downloads the documents found. It is important to note that the visibility of a spider is limited to the connected graph of documents commencing at the starting point. Starting from a particular document, there is no guarantee that all documents in the World-Wide Web will be somehow indirectly connected to it. Thus, the need for a set of starting points to ensure a better coverage. Directories of web sites are usually good starting points. Spiders are usually the most elaborate component of the search engine and manipulate large databases of URLs [40, 55]. With respect to the visibility of the World-Wide Web space, it is also important to note that most spiders do not traverse frame-based web pages and can not access dynamically generated web pages or authenticated ones. Dynamic web pages are web pages that do not exist on the web server and are generated on the fly after a web-based form is filled and submitted. Usually these dynamic web pages contain answers to database queries.

The indexer (and parser) receives a document downloaded by the spider and parses it to extract new URLs from the links found in the document. These new references are passed back to the spider for download. When parsing the document, text is also analyzed to extract terms for indexing. The extraction of terms differs from one system to the next. Some may index the full text, some may select terms depending on their location in the document (i.e. title, header, etc.), while others may choose to analyze only the beginning of the document. The third component is the index itself, which is the result of the automatic, and recursive traversal. The index depends largely upon the information retrieval technology chosen. While some systems use inverted file technology, others use the vector space model or the probabilistic indexing model [177, 213]. What is called a search engine is in fact the program that sifts through the index to answer user requests. In other words, it is the interface between the user and the index. This automatic and autonomous exploration of the World-Wide Web structure to build a searchable index seems a simple and elegant solution for information retrieval in the World-Wide Web. However, it involves ethical concerns relating to its resulting impact. Spiders continuously retrieve entire objects and then discard them after retaining some of their content. It seems that robots are generating substantial load on web servers, and are generally increasing the traffic on the Internet backbone, especially with the proliferation of such spiders [156]. To index the whole World-Wide Web, the entire World-Wide Web content has to be downloaded. With the plethora of search engines doing the same exercise, the entire corpus is circulating continuously

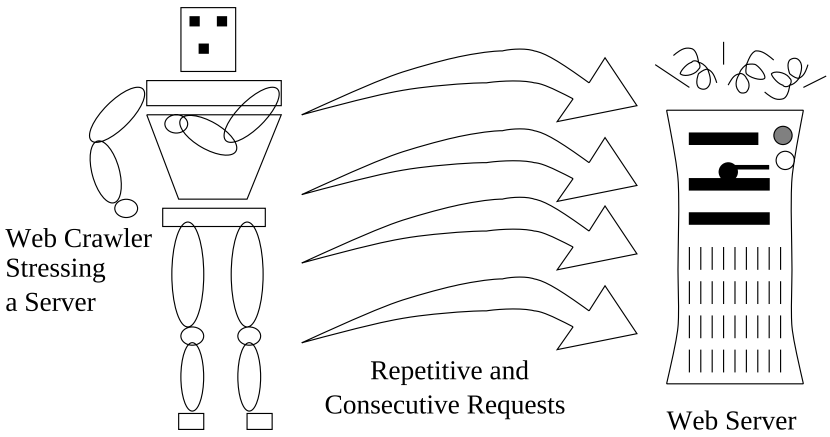


Figure A.14: Crawlers overload Web Servers.

on the Internet. The number of crawlers doing this procedure today is estimated at 400 [237]. This means that more than 400 processes are independently downloading the entire visible content of the World-Wide Web to build their own indexes. Since the content is continuously changing, indexes need to be updated in an uninterrupted manner. For each pass, the entire World-Wide Web content, estimated today at more than one tera-byte, is sent through the network more than 400 times, again and again. It is true that crawlers usually download only text documents (i.e. HTML files) and not images and other large artifacts in the Web, but there exist more and more specialized crawlers (like the one described in Chapter 4) that also download images, Videos, VRLM objects (i.e. Virtual Reality), pdf and postscript documents, software, etc. At the rate the content of the World-Wide Web is increasing (behaving similarly to an exponential growth) and the number of new crawlers are developed, this approach is definitely not scalable and even not viable in the long term even if the network bandwidth is enlarged by one order of magnitude.

Moreover, spiders can unnecessarily overload web servers. When parsing a web page, or a set of web pages from the same site, to extract hyperlinks to follow, these hyperlinks often end up pointing to pages localized in a common web site. By continuously and consecutively requesting all documents pointed by those hyperlinks, spiders overload web servers and can flood them and prevent them from serving other users. It is desirable to spread the request to the same server in time to give the server the opportunity to attend to other users (or spiders). Furthermore, spiders tend to download all possible documents; all the documents that the spider comes across are requested. This is not always necessary, and sometimes even undesirable. Indexing a web page can sometimes be meaningless if the web page is duplicated, under construction, or simply not meant by its author to be indexed. This has precipitated controversy. Based on these concerns, guidelines for implementing spider-like programs were proposed [154, 155]. Breadth-first traversal, instead of depth-first traversal of the Web was suggested, for example, to alleviate some of the load on the servers. The suggested standards for robot exclusion specify some mechanisms to indicate to spiders which part of a server can be indexed, and which part should not be accessed, and suggest to avoid submitting too many requests to the same server at any given time. Spiders that follow these recommendations exclude superfluous web pages and avoid bombarding servers by sending consecutive requests to the same server in a short period of time. This is done by randomly selecting, rather than sequentially selecting, a page to visit from the list of all pages still to visit. Alternatively, a list of servers visited can be maintained with the time of last visit. This list could be consulted when a page to request is chosen from the list of pages still to visit, avoiding calling upon the same server too quickly.

Other researchers advocate spider-less techniques to index the World-Wide Web. Aliweb [153] is particularly interesting in that it does not use a spider to create its index. Instead, web page authors write their own descriptions of their pages, and register with Aliweb, which indexes these summaries. This process removes the problems of server overload that web crawlers in general are causing. Moreover, only pertinent information is indexed since it is submitted by the authors or web site managers. However, it does require extra work on the part of the pages' authors to describe their resources and notify Aliweb about their existence. The Harvest system [34] and the Multiple Layered Database approach (MLDB) [127, 276] are supportive of a distributed index architecture. Harvest solves the problem of server load and network traffic by moving the indexing task to the information provider site. The system comprises two major components: a gatherer and a broker. A gatherer collects information from a provider for indexing purposes, like a spider would do, while a broker provides an interface to the index. The originality of the solution is the variety of possible configurations. Many gatherers and brokers can co-exist on the network. While brokers can specialize by providing customized indexes, gatherers can reside on the same host as the information server, and be executed during off-peak periods. The Multiple Layered Database approach [127], presented in Chapter 2, is to build a distributed repository of metadata describing artifacts on the Internet. Specialized tools, similar to Essence [129], are executed on information provider sites by the site maintainer during off-peak times

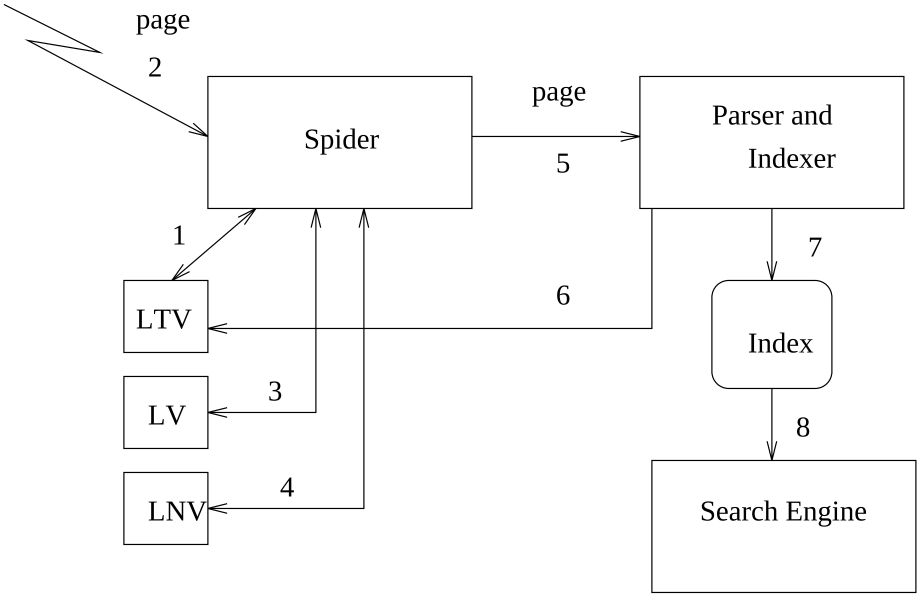


Figure A.15: The spider-based search engine general architecture.

to extract pertinent data from the documents, and build a local structured database. By successive transformations, these distributed databases can be generalized to form a multi-layered structure where each layer generalizes the concepts held in the lower level. This architecture offers to users the possibility to browse the hierarchy of layers and interactively drill-down to the pertinent resources. Moreover, using structured databases, this approach takes advantage of the high precision and recall that database technology presents. In addition, the Multiple Layered Database architecture has knowledge discovery potential that no other techniques offer.

How search engines work

Figure A.15 shows the general architecture of a search engine with four components: the spider, the indexer, the index, and the search engine. The spider component, as well as the indexer can be replicated to parallelize the crawling and indexing. Therefore, the index might be distributed or centralized. Other components are different URL lists managed in a shared local database. In the simplest version, there are three lists: the first list, LTV, is a list of URLs to visit. Initially, it contains the starting points, then all found URLs are also stored in this list for processing. When a page is visited, its URL is removed from LTV and stored in LV, the list of visited pages. This list avoids revisiting the same page again. The third list, LNV, is a list of pages not to visit. This is to conform to the robot exclusion guidelines and to avoid revisiting known problematic sites. The spider selects a URL from LTV (1) and downloads the corresponding page (2), then moves the URL from LTV to LV (3). LNV is updated, if necessary (4). If the page is downloaded successfully, it is passed to the indexer (5). The indexer parses the page to extract new URLs, and stores them in LTV (6), if they do not already exist in LV or LNV, then extracts terms from the page for indexing purposes (7). Once LTV is empty, the entire "visible" web is indexed. The index is used by the search engine to answer user requests (8).

Because the index rapidly becomes stale, it is necessary to rebuild the index often by restarting the web traversal with LTV initialized with a starting point list. The starting point list can be augmented with new URLs submitted by users who want their sites indexed.

The described scenario is a simplistic one. There are other considerations when "spidering" the Web. The execution of a web spider is a challenging task that involves performance issues as well as "net social" issues. Despite the high parallelism in the implementation of Web crawlers, due to the very large size of the Web it takes weeks, if not months, to index the World-Wide Web [237]. Hence the existence of different strategies for URL selection from LTV. For example, spiders may store modification dates of pages visited in order to revisit pages that are updated regularly more often than static pages. Particular domains or popular sites may also be revisited more often than others. In addition, when selecting a URL to visit, the spider has to take into account the fact that it should not consecutively request several pages from the same server. In other words, the selection of URLs from LTV is not a sequential selection, but may involve complex heuristics.

In a recent paper[55] Cho, Garcia-Molina and Page present a new technique for URL selection based on backlink counts. URLs that are referenced more often by other pages are selected and visited more often by the spider. Priority in the selection from LTV and the re-iteration through LTV is given to "cyber-popular" pages.

Relevancy and Ranking of Documents

Due to the profusion of commercial search engines, the competition pushed spiders to crawl and index as many resources as possible in order to assert the most complete index. Today, some search engines claim to have indexed more than 100 million documents [237]. However, completeness is not the most important factor. As long as the search result is adequately presented, indexing the whole Internet or a portion of it wouldn't matter. Indeed, a set of 8000 matching documents or a set of 1000 matching documents does not make a big difference for a user. The user still has to go over this large set to identify the documents fitting the quest. It has been demonstrated in [223] that any single web search engine provides the user with only 15-42 % of the relevant documents. Because of the over-abundance of resources on the Internet, any search query would yield hundreds, if not thousands, of matches. Search engines, like traditional information retrieval systems, rank the resources found by relevancy and present them ordered with the most relevant first. Apart from the size of the index, the model chosen for indexing and the user interface, the most important difference between search engines is the ranking mechanism. Relevance is a very subjective term. Moreover, it is very difficult to ascertain with high confidence the real need of the user from a query. Queries present little context, and search engines do not learn from past experience (yet).

Most search engines would analyze how often terms appear in relation to other terms in the document. The higher the frequency, the more relevant the document is considered. The location of the term is also taken into consideration. Being semi-structured, HTML documents can disclose information regarding the location of a word or phrase. For example, if a term appears in the title or in a header of the document, it is considered more relevant than a word in the body of the document. Frequency and location are not the only factors used to determine relevancy. Each search engine uses different stimuli to influence the relevancy. Since relevance ranking is capital for the success of search engines, these algorithms are securely guarded and rarely disclosed. Usually, the factors used to calculate the relevance of a document are the following. These are heuristics that may not have large support.

Frequency: Documents with a higher frequency of keywords are ranked better than documents with a low keyword frequency. The frequency is calculated using the percentage of keywords in the document in order to normalize documents of different sizes.

Location: When counting the frequency of keywords, words can be weighted based on their location in the document or their location with respect to each other. Words that appear in the title are generally considered more important. Other locations considered are the headers, the URL of the document, the anchors (or hyperlinks), and the beginning of the document. A word is considered more important if it appears in the first paragraph (or paragraphs) for example. The closeness of keywords to each other can also be a factor to increase the word weight. Some search engines may also disadvantage isolated words by favouring keywords that appear in complete sentences.

Entirety: The number of matched keywords in a query is an important factor in determining the relevancy. The more terms from the query match words in the document, the more the document is relevant to the query. In other words, if all search terms appear, it is considered better than if only some of the terms do. Documents can be ranked by **entirety** to create clusters in which documents can be ranked by **frequency**.

Size: It seems that some search engines favour smaller documents over larger ones[237].

Age: The date of the last modification of a document can be accessed easily. This date can be used to favour old documents for instance. By keeping track of modification dates of documents, search engines "learn" the update frequency of documents and favour in their ranking documents that are updated regularly.

Directory: Hybrid search engines (i.e. those that also maintain a directory of sites) usually favour documents that appear in their catalog. Since the sites appearing in their catalogs are reviewed, they are considered more important (i.e. relevant).

Links: Web pages that are referenced by many other pages seem to be more important than others. This popularity of web pages can be estimated by counting the number of links leading to a page. When ranking pages, search engines favour pages that have many links pointing at them. Links can also be weighed based on the popularity of the web page they come from. Hence the notion of link quality.

Metadata: The Dublin Metadata workshop has stressed the importance of metadata (i.e. document descriptors) in networked documents to facilitate resource discovery [259, 258]. Extensions to the HTML specifications include new tags allowing the description of keywords and content summary inside the HTML document. Figure A.16 shows an excerpt of an HTML document example using these tags (see Appendix D for more information on the HTML META tag extensions). Obviously these tags have to be entered by the document author. However, when present, these keywords may reflect better the real content of the document. During the relevance ranking of web pages, some search engines may favour pages that have keywords from the meta tag match search terms before those that have matches only with terms in the document body. Thus, terms extracted from the meta tags (i.e. keywords and description) get a higher weight than others.

Domain: The Internet domain from which web pages are retrieved may play a role in the ranking. Domains like ".com" or ".org" can be ranked before others.

```
<HTML>
<HEAD>
<META NAME=”description” CONTENT=”This is a survey about search engine tricks”>
<META NAME=”keyword” CONTENT=”Spider,Indexing,relevance ranking,spamming”>
<TITLE>Search Engine Tricks</TITLE>
</HEAD>
<BODY>
...
```

Figure A.16: A snippet from an HTML document with META tags.

$Money$: Unfortunately, some search engines may allow companies to pay to have their pages ranked first when a match with their pages occurs. Being ranked among the 10 or 20 first documents is important since rare are the users who dig deeper than two or three pages of document lists. Statistics show that less than 7% of users go beyond the fist three pages of results[237].

Knowing these ranking factors, some web pages authors try to “cheat” to see their web pages better ranked. This is commonly called spamming. Spamming consists of adding for example the same keyword over and over in the same page. These keywords are usually invisible to users (using comment tags or using same colour for text and background), but clearly visible to web crawlers. One of the most common techniques is also to put keywords in the meta tags that do not relate to the page’s actual content, like general keywords that are frequently used in queries, or keywords copied from meta tags of sites that rank high by search engines. There are techniques and heuristics to detect spamming. Some search engines penalize documents, when detecting spamming, by ranking them lower.

Some new techniques for ranking retrieved documents were recently unveiled: Direct Hit by Direct Hit Technologies Inc.[7], CLEVER by IBM Almaden Research Labs[8], and PageRank by Stanford University[9].

Direct Hit is a relevance feedback approach. Users’ selections from the usual search result are recorded. For any given query, the documents that are selected the most become

[7] http://www.directhit.com

[8] http://www.almaden.ibm.com/cs/k53/clever.html

[9] http://google.stanford.edu

the most relevant the next time the same query is submitted. Obviously, this technology is useful only if the query is frequent. Direct Hit needs to gather information about users' selections per query. When enough information is gathered, the Direct Hit ranking can proceed.

Direct Hit does not exclude the other ranking strategies but is a supplement that can be added on any ranking strategy.

This idea is not completely novel. MetaCrawler presented in section A.2.4 exploits user chosen references as metrics for their search result ranking. This metric takes also into account the origin of the resource selected [224].

PageRank used in the experimental search engine at Stanford University is based on links[40]. The basic component of its ranking metric is link popularity. A large link graph of the web, called citation graph, is constructed. It represents all present links between web pages. This graph is used to attribute the PageRank weight to each web page, which is later used for ranking pages when appearing in a search result. The mechanism to calculate PageRank is the following: Each Page P has a number of links coming out of it $C(P)$ (C for citation), and a number of pages pointing at it $P_1, P_2, ..., P_n$. PageRank of P $PR(P) = (1-d) + d \times (\sum_{k=1}^{n} \frac{\mathrm{PR(P_k)}}{\mathrm{C(P_k)}})$. d is a dumping factor between 0 and 1. Intuitively, PageRank represents the probability to choose a page after a random browsing[40, 55].

CLEVER also is heavily based on link frequency. It ranks documents by measuring links between them. The main purpose of CLEVER, as stated in [46], is not to simply rank documents in a search result, but to find authoritative resources in a pool of web pages. The objective is to build directories like Yahoo directories but automatically find the authoritative entries in each category. Starting from categories (i.e. category hierarchy), queries are generated and sent to standard search engines. For each given query, the result that is gathered from the search engines constitutes a pool of documents. In order to rank these documents and keep the most authoritative in the topic of the query (i.e. current directory category in process), all pages pointing to and all pages pointed by the documents in the pool are retrieved and add to the pool of documents. All pages collected are then weighted by iteratively calculating the weight of links pointing to each page, and carry more weight from pages that cumulate more weight in each iteration. This way, pages that are pointed by important pages get more weight and become the “authority”. CLEVER uses also page content, like text in hyperlinks, to add more weight to corresponding links if the search terms appear in the text. By sorting the pages by weight, the more authoritative are selected to represent the catalog category. CLEVER can also be used for real-time search result ranking, however, the calculation of weight may take time depending upon the size of the root set. One might classify CLEVER as a knowledge discovery approach since it finds networked information based on relationships between resources, rather than finding resources on the Internet.

Concept-Based Retrieval

Furnas, et al. [111] show that due to widespread synonymy and polysemy in natural languages, indexing methods based on the occurrence of single words do not perform adequately. Concept-based methods match words with similar meanings rather than string patterns. Concept-based search tools can find related documents even when they do not contain any of the search words specified in the query. By searching for documents about "fruit jam", one can retrieve documents with "blueberry jam" for example (i.e. subsumption). Concept siblings and synonyms are also possible. One way to broaden the scope of a search to include such concepts, is to use thesauri. A thesaurus correlates terms and helps search engines to conjoin the search terms with related terms. Thesauri can either be static based on a language, or built dynamically by statistically tracking cross-references for words commonly appearing together in queries or documents. Relevancy ranking usually favours matches with search terms before matches with similar concepts.

Stemming is also a common technique used for concept-based retrieval. It consists of extracting the stem from search terms and concatenating suffixes to generate new terms. This approach, however, may generate poor precision in many cases. The stem "tea" for example, may generate words like "tear", "teak", "team", "teal", "teach", etc. Stemming nevertheless has advantages for substring matching.

The document similarity search can also be considered a type of concept-based retrieval. The word weighting technique generating term weight vectors representing documents can be used to compare documents with a similarity function similar to the function presented in section A.1.1. By starting with a document about kangaroos, a similarity search could find documents about wombats, even though, these documents do not contain the term kangaroo. Since kangaroos and wombats are both marsupials living in Australia, documents about these two leaping herbivorous animals may share many similar words, hence the vector similarity. Research in knowledge representation and natural language processing is paving the way for automatic paraphrasing[262]. Techniques for generating paraphrases of queries can lead to more precise content retrieval.

Some other approaches use concept hierarchy to match concepts. These concept hierarchies are either built by experts like in [127] and may contain synonyms from different languages allowing multi-lingual querying, or built by automatic subject extraction using assumption grammars[273, 272].

A.2.4 Agents for Information Retrieval

While spiders are a type of information gathering agent, there are other types of agents specialized in information retrieval. Agent based information retrieval has stimulated great interest over the past several years. While the definition of an agent is still in question and generating much debate, research in related fields has been very productive. Heterogeneous database access, knowledge representation, cognitive science, learning algorithms, communication protocols, distributed query processing, are all fields of interest for agent-based or multiagent-based information gathering systems[138, 38].

Desirable characteristics for an “intelligent” agent were proposed or alluded to by many researchers. No single agent or agent prototype today compounds all the desirable characteristics yet. However, many of the agents include most of them. The commonly advisable characteristics which distinguish agents form other software components are:

Autonomy: An agent should be able to take initiatives and have a certain control over its actions like modifying high-level requests, dialoguing with the user to clarify requests or even invoking help from other agents.

Adaptability: An agent should not be static but able to change the sequence of its actions in response to its environment. Ideally, it should learn from its interaction with users and other agents and customize itself to the preferences of its users.

Communicability: An agent should be able to communicate and engage in involving communications with the user and other agents. Dialogues with the user are necessary to disambiguate some requests or priorities, while communication and collaboration with other agents or users is paramount in information gathering and task delegation.

Personality: An agent should have a perceivable and plausible behavioural and emotional state. This well-defined agent character is significant for effective communication and conversation like for avatars representing or acting on behalf of users in virtual spaces.

Mobility: Agents typically run on one host and retrieve information scattered around the Internet. This may make them look like moving, however, only the resources are transmitted. Real mobile agents (i.e. agents that move, transport themselves or duplicate themselves, to execute on different hosts on the web) exist nevertheless. While the usefulness of agent mobility is still disputed, itinerant agents can be very useful for information gathering. When the amount of information on servers is greater than can reasonably be transmitted to a client for processing, mobile agents can be sent to execute on servers for information gathering and filtering. Mobile information gathering agents are of great importance when client machines are disconnected or lack the necessary processing power and resources to perform the filtering.

In the information retrieval context, an agent can be seen as a program that learns about user needs adapts to the needs, and reacts to them by collecting and retrieving resources or even knowledge that could satisfy those needs.

Networked information retrieval agents are personal assistants that act on behalf of users on the World-Wide Web, often relying on tools and services already available. After users state their needs, agents determine where to find the information on the Internet and how to retrieve it. One such simple agent is the MetaCrawler[224] which alleviates the burden of resubmitting requests to different search engines by automatically simultaneously sending the user query to several of the most popular search engines. The agent knows specific features of each search engine and adapts the search query accordingly. Users need not remember these specific characteristics and peculiarities of the different query interfaces. Moreover, the agent merges the different results and eliminates unnecessary duplicates. It even downloads the web pages in the result list to scan their content for search terms. This allows further filtering and the presentation of concise relevant result list. SavvySearch[135] does a similar job, but rather than using always the same resources it selects the search engines to query. SavvySearch learns to identify which search engines are most appropriate for particular queries by tracking long-term performance of these search engines. The agent keeps a compendium of search experience and uses it to rank specialized search engines to query. SavvySearch does not adapt to the users but rather adapts to the available resources. The approach is very similar to the information agent matchmaking presented in [158], where an intelligent facilitator (agent) matches information providers with consumers' needs. Other agents adapt to the user by observing the users activity (i.e. browsing, querying, etc.) and may recommend resources or queries that other users on the system with similar profile and needs have retrieved or submitted. The agent manages user profiles that it updates regularly based on the user interest and activity on the World-Wide Web. WebWatcher[18] developed at Carnegie Mellon University observes users' browsing activity and provides advice on which hyperlink might lead to preset goals. Some commercially available agents, based on user behaviour observation, predict user needs and automatically download and locally cache web pages potentially interesting to the user. If the user, for example, regularly accesses a particular set of pages at a particular time in the day (i.e. news, stock market, etc.), the agent would automatically retrieve the pages a few minutes before the usual access time and cache them locally in preparation for the user's on-line browsing.

One popular task for agents is information filtering. Information filtering in the Internet context is the extraction of relevant information from large volumes of dynamically generated documents[192]. USENET news and electronic mail are the most commonly used dynamic large document collections for information filtering in the Internet. The dynamic nature of the document repository is a binding requirement for information filtering systems. A continuously running process sifts through incoming e-mail messages for instance, and filters out messages that satisfy the user's requirement or interest. A similar agent can listen on news groups channels and alert the user when interesting topics are being discussed. The agent could even summarize articles of interest. SIFT[267], an agent developed at Berkeley, uses keywords describing users topics of interest to filter new netnews articles and send by e-mail to users articles or summaries of articles that are of interest to these users. The DICA agent[3] applies this approach on the World-Wide Web and monitors previously found relevant pages for any changes. The DICA agent only reports interesting changes.

A new breed of agents go beyond information filtering or gathering from the World-Wide Web. Some shopping agents access vendors' web pages and look for bargains by comparing prices offered. Suggestions are presented to the user or automatic ordering of the product is initiated. Netbot Jango[10] one such web shopping agent, compares product prices offered on-line, creates accounts on users' behalf, and when instructed, can initiate the purchasing.

Networked agents for information retrieval from the Internet can be classified into three categories: *Exploiter* agents, *Watchdog* agents, and *Apprentice* agents. Exploiters take advantage of existing networked services (or other agents) to find resources. Watchdogs monitor given resources for interesting changes. They learn about the environment in which

[10] http://www.jango.com

they act. Apprentices learn from users behaviour and infer new needs. An overlap between these categories is possible and is desirable. Ideally, a good agent should be an apprentice, a watchdog and an exploiter. In other words, it should learn about its users, learn about its environment, and take advantage of existing services.

Many of the information gathering agents are at the same time in the information retrieval domain and the data-mining domain. By returning information or knowledge from within documents, like comparison-shopping agents[71] rather than returning resource references, agents can be perceived as web mining processes.

A.3 Summary and Conclusion

This survey outlines major information retrieval approaches used on the Internet and pinpoints some of their weaknesses. Alas, the currently most successful and widely practiced approaches are using “spidering” techniques to index documents on the Internet, crawling the Web from one artifact to the other. While heavily “parallelizing” the process in different machines, like some business applications do, may partially solve the scalability problem, it remains that the whole content of the Web has to be continuously downloaded to build the indexes and keep them current. Despite the parallel indexing solutions, and the smart methods for updating indexes, given the tremendous growth of the Internet size and the great number of the independently built indexes, spidering the Web as it is done today will always overload the Web traffic; this confirms our first thesis regarding the Web overload (see Chapter 1). Spidering the Web could be viable if it is done in a collaborative way where the resources to index could be divided geographically or other, and the indexes shared at a higher level. Small indexes could be built locally and sent to a central body for sharing and merging with generally visible index. This avenue is explored in Chapter 2 with Virtual Web Views that could centralize, at a high level, indexes gathered from different channels. The high level indexes are updated only when the changes are propagated from the different channels.

Another major problem with current technology is the relevancy issue. The relevance of documents in a search result is far from satisfactory, making current search engines almost useless. Research effort is underway to develop new techniques, from concept-based retrieval to web mining. This survey reports the initial undertaking. The research may necessitate contribution and collaboration from different communities such as artificial intelligence, data mining, information retrieval, natural language processing, etc.

Web Mining, knowledge discovery from the World-Wide Web, is becoming a new research trend. Chapter 2 covers a brief survey on Web Mining.

There are two ways of being creative. One can sing and dance.
Or one can create an environment in which singers and dancers flourish.
Unknown

Appendix B

Web Mining Language (WebML) Grammar

We present a simplified extended Backus-Naur Form (BNF) grammar for WebML. We make use of the following conventions: "[ ]" represent zero or one occurrence; "{ }" represent zero or more occurrences; upper case words represent WebML keywords; characters between quotes are characters or symbols accepted in WebML; words starting with upper case are variable; words in italic and starting with lower case are terminal categories. The terminal categories are: Type (*integer*, *float*, *string*), Identifier (*identifier-Tab*, *identifier-Var*, *identifier-Attrib*, *identifier-AttSet*), and Function (*function-user*, *function-boolean*, *function-compare*). Types are self-explanatory. *identifier-Tab* is the name of a relation; *identifier-Var* is a variable name; *identifier-Attrib* is the name of an attribute; and *identifier-AttSet* is the name of a set-valued attribute (example: set of links, set of authors, etc.). Functions are user defined functions: *function-boolean* returns a boolean 0 or 1; *function-compare* takes two arguments, A and B, and returns -1 (if $A < B$), 0 (if $A = B$) or 1 (if $A > B$); and *function-user* has no restrictions on the value it can return.

WebML-Query
    ::=   Select-clause [INTO *identifier-Tab*]
           From-clause
           Pertinence-clause
           Where-clause
           [Ordering-clause]

[Ranking-clause]
[Grouping-clause]
[Threshold-clause]

Select-clause
::= Retrieval-header | Mining-header

Retrieval-header
::= Selection-header Attribute-name-list

Selection-header
::= SELECT | LIST

Attribute-name-list
::= ∗ | Attribute-name {, Attribute-name}

Attribute-name
::= [*identifier-Var.*]*identifier-Attrib*

From-clause
::= FROM Relation-list

Relation-list
::= Relation-reference {,Relation-reference}

Relation-reference
::= *identifier-Tab* [*identifier-Var*] [LEVEL *integer*]

Pertinence-clause

::= [RELATED TO Name-list] [IN Location-list]

Name-list

::= Name-reference{, Name-reference}

Location-list

::= Name-reference{, Name-reference}

Name-reference

::= [*identifier-Var.*]*identifier-Attrib*

Where-clause

::= WHERE Where-predicate

Where-predicate

::= Predicate-term | Where-predicate OR Predicate-term

Predicate-term

::= Predicate-factor | Predicate-term AND Predicate-factor

Predicate-factor

::= [NOT] Predicate-condition

Predicate-condition

::= Condition | (Where-predicate)

Condition

::= Compare-condition
| In-condition
| Like-condition
| Between-condition
| Exist-condition
| Null-condition

Compare-condition

::= Scalar-expression Comparison Scalar-expression
| Scalar-expression Comparison (Select-expression)

Comparison

::= "=" | "! =" | "<" | ">" | "<=" | ">="

In-condition

::= Scalar-expression [NOT] IN Set-expression

Set-expression

::= Set-of-scalar | *string* (Link-path) | (Link-path) *string*

Link-path

::= Link-element{Link-element} [OR Link-path]

Link-element

::= Link[$*$]

Link

::= $\mapsto|\rightarrow|\Rightarrow$

Set-of-scalars

::= *identifier-AttSet* | (Constant-list) | (Select-expression)

Constant-list

::= Constant{, Constant}

Constant

::= String-item | *integer* | *float*

String-item

::= *string* | SUBSTRING(*string*)

Select-expression

::= Retrieval-header
From-clause
Pertinence-clause
Where-clause
[Ordering-clause]
[Ranking-clause]
[Grouping-clause]

Like-condition

::= [Quantifier] Like-element [NOT] Like-operator Set-of-scalars

Like-element

::= Attribute-name | Set-of-scalars

Like-operator

::= LIKE | CLOSE TO | COVERED BY | COVERS | *function-boolean*

```
Quantifier
        ::=   ONE OF | ALL

Between-condition
        ::=   Attribute-name [NOT] BETWEEN Scalar-expression AND Scalar-expression

Exist-condition
        ::=   [NOT] EXISTS (Select-expression)

Null-condition
        ::=   Attribute-name IS [NOT] NULL

Scalar-expression
        ::=   EXACT Constant
        |     Scalar-term
        |     Scalar-expression "+" Scalar-term
        |     Scalar-expression "-" Scalar-term
        |     Scalar-expression "|" Scalar-term

Scalar-term
        ::=   Scalar-factor
        |     Scalar-factor "*" Scalar-factor
        |     Scalar-factor "/" Scalar-factor

Scalar-factor
        ::=   ["+" | "-"] Scalar-primary

Scalar-primary
        ::=   Constant | Attribute-name | (Scalar-expression) | Aggregate-function(Scalar-expression)
```

Aggregate-function

::= COUNT | AVG | SUM | MIN | MAX | *function-user*

Mining-header

::= Describe-header | Classify-header | Associate-header

Describe-header

::= DESCRIBE IN RELEVANCE TO Attribute-name-list

Classify-header

::= CLASSIFY ACCORDING TO Attribute-name{, Attribute-name}
IN RELEVANCE TO Attribute-name-list

Associate-header

::= ASSOCIATE IN RELEVANCE TO Attribute-name{, Attribute-name}

Ordering-clause

::= ORDER BY Order-item-list

Order-item-list

::= Order-attribute{, Order-attribute}

Order-attribute

::= Attribute-name [ASC | DESC]

Ranking-clause

::= RANK BY Ranking-function

Ranking-function

::= ACCESS | INWARD | OUTWARD | *function-compare*

Grouping-clause

::= GROUP BY Attribute-name{, Attribute-name}

Threshold-clause

::= Threshold-specification{, Threshold-specification}

Threshold-specification

::= Threshold-name THRESHOLD "=" Threshold-value [FOR Attribute-name]

Threshold-name

::= SUPPORT | CONFIDENCE | CLASSIFICATION

Threshold-value

::= *integer* | *float*

Note that the syntax presented above is very permissive. It allows the generation of some constructs that are not legal or semantically incorrect in WebML. For example, there is no distinction between numeric and string expressions in *Scalar-expression*. We have simplified the grammar for clarity, to avoid complicated production rules due to the context-sensitivity of WebML. For clarity, we have also allowed redundancy, like for *Name-list* and *Location-list*.

Language is called the garment of thought: however, it should rather be,
language is the flesh-garment, the body, of thought.
Thomas Carlyle

Appendix C

Data Mining Query Language (DMQL) Grammar

We present a simplified extended Backus-Naur Form (BNF) grammar for the Data Mining Query Language DMQL. We make use of the following conventions: "[ ]" represent zero or one occurrence; "{ }" represent zero or more occurrences; upper case words represent WebML keywords; characters between quotes are characters or symbols accepted in DMQL; words starting with upper case are variable; words in italic and starting with lower case are terminal categories. The terminal categories are: Type (*integer*, *float*, *string*) and Identifier (*identifier-DB*, *identifier-Tab*, *identifier-Var*, *identifier-Attrib*, *identifier-Cube*). Types are self-explanatory. *identifier-DB* is the name of a database; *identifier-Tab* is the name of a relation; *identifier-Var* is a variable name; *identifier-Attrib* is the name of an attribute; and *identifier-Cube* identifies a multi-dimensional data cube.

DMQL-statement
    ::= DMQL-hierarchy-manipulation | DMQL-query

DMQL-hierarchy-manipulation
    ::= Hierarchy-definition | Hierarchy-insertion | Hierarchy-deletion

Hierarchy-definition
    ::= DEFINE HIERARCHY FOR Concept-hierarchy [Hierarchy-name":"] Hierarchy-expression

Hierarchy-expression

::= Schema-hierarchy | Group-hierarchy

Schema-hierarchy

::= "{" Schema-attribute-list "}" ">" "{" Schema-attribute-list "}"

Group-hierarchy

::= "{" Constant-list "}" ">" "{" Constant-list "}"

Hierarchy-name

::= *string*

Concept-hierarchy

::= Schema-attribute-name | Attribute-name "}"

Schema-attribute-name-list

::= Schema-attribute-name{, Schema-attribute-name}

Schema-attribute-name

::= [*identifier-Tab.*]*identifier-Attrib*

Constant-list

::= Constant{, Constant}

Constant

::= *string* | *integer* | *float*

```
Hierarchy-insertion
    ::=  INSERT Concept-name UNDER Concept-name
         TO HIERARCHY [Hierarchy-name “:”] FOR Concept-hierarchy


Concept-name
    ::=  Constant | Attribute-name


Hierarchy-deletion
    ::=  DELETE Concept-name UNDER Concept-name
         FROM HIERARCHY [Hierarchy-name “:”] FOR Concept-hierarchy


Attribute-name
    ::=  [identifier-Var.]identifier-Attrib


DMQL-query
    ::=  Summarizer-query
    |    Comparator-query
    |    Associator-query
    |    Classifier-query
    |    Clusterer-query
    |    Predictor-query


Summarizer-query
    ::=  [ Use-db-clause ]
         [ Hierarchy-clause {, Hierarchy-clause} ]
         Summarizer-clause
         From-clause
         Where-clause
         [ S-Threshold-clause ]
         [ Show-clause ]
```

```
Comparator-query
        ::=  [ Use-db-clause ]
             [ Hierarchy-clause {, Hierarchy-clause} ]
             Comparator-clause
             From-clause
             Where-clause
             [ S-Threshold-clause ]
             [ Show-clause ]


Associator-query
        ::=  [ Use-db-clause ]
             [ Hierarchy-clause {, Hierarchy-clause} ]
             Associator-clause
             From-clause
             Where-clause
             [ A-Threshold-clause ]
             [ Show-clause ]


Classifier-query
        ::=  [ Use-db-clause ]
             [ Hierarchy-clause {, Hierarchy-clause} ]
             Classifier-clause
             From-clause
             Where-clause
             [ C-Threshold-clause ]
             [ Show-clause ]


Clusterer-query
        ::=  [ Use-db-clause ]
             [ Hierarchy-clause {, Hierarchy-clause} ]
             Clusterer-clause
```

```
          From-clause
          Where-clause
          [ T-Threshold-clause ]
          [ Show-clause ]


Predictor-query
    ::=   [ Use-db-clause ]
          [ Hierarchy-clause {, Hierarchy-clause} ]
          Predictor-clause
          From-clause
          Where-clause
          [ T-Threshold-clause ]
          [ Show-clause ]


Use-db-clause
    ::=   USE [ DATABASE ] identifier-DB


Hierarchy-clause
    ::=   USE HIERARCHY Hierarchy-name FOR Attribute-name


From-clause
    ::=   FROM Source-list


Source-list
    ::=   Relation-list | Cube-reference


Relation-list
    ::=   Relation-reference {,Relation-reference}
```

Relation-reference

 ::= *identifier-Tab* [*identifier-Var*]

Cube-reference

 ::= *identifier-Cube*

Where-clause

 ::= WHERE Where-predicate

Where-predicate

 ::= Predicate-term | Where-predicate OR Predicate-term

Predicate-term

 ::= Predicate-factor | Predicate-term AND Predicate-factor

Predicate-factor

 ::= [NOT] Predicate-condition

Predicate-condition

 ::= Condition | (Where-predicate)

Condition

 ::= Compare-condition
 | In-condition
 | Like-condition
 | Between-condition
 | Exist-condition
 | Null-condition

```
Compare-condition
    ::=   Scalar-expression Comparison Scalar-expression
    |     Scalar-expression Comparison (Select-expression)


Comparison
    ::=   “=” | “! =” | “<” | “>” | “<=” | “>=”


In-condition
    ::=   Scalar-expression [NOT] IN (Set-of-scalar)


Set-of-scalars
    ::=   Constant-list | Select-expression


Select-expression
    ::=   Select-clause
          From-clause
          Where-clause
          [ Ordering-clause ]
          [ Grouping-clause ]
          [ Having-clause ]


Like-condition
    ::=   Attribute-name [NOT] LIKE Constant


Between-condition
    ::=   Attribute-name [NOT] BETWEEN Scalar-expression AND Scalar-expression


Exist-condition
    ::=   [NOT] EXISTS (Select-expression)
```

```
Null-condition
      ::=   Attribute-name IS [NOT] NULL


Scalar-expression
      ::=   EXACT Constant
      |     Scalar-term
      |     Scalar-expression “+” Scalar-term
      |     Scalar-expression “-” Scalar-term
      |     Scalar-expression “|” Scalar-term


Scalar-term
      ::=   Scalar-factor
      |     Scalar-factor “*” Scalar-factor
      |     Scalar-factor “/” Scalar-factor


Scalar-factor
      ::=   [“+” | “-”] Scalar-primary


Scalar-primary
      ::=   Constant | Attribute-name | (Scalar-expression) | Aggregate-function-reference


Aggregate-function-reference
      ::=   COUNT( ∗ )
      |     Aggregate-function ([ALL] Scalar-expression)
      |     Aggregate-function(DISTINCT Attribute-name)


Aggregate-function
      ::=   COUNT | AVG | SUM | MIN | MAX
```

Select-clause

::= SELECT [ALL | DISTINCT] Attribute-name-list

Attribute-name-list

::= $*$ | Attribute-name {, Attribute-name}

Ordering-clause

::= [ORDER BY Order-item-list]

Order-item-list

::= Order-attribute{, Order-attribute}

Order-attribute

::= Attribute-name [ASC | DESC]

Grouping-clause

::= GROUP BY Attribute-name{, Attribute-name}

Having-clause

::= HAVING Where-predicate

Summarizer-clause

::= SUMMARIZE Attribute-name-list WITH RESPECT TO Attribute-name-list AS *string*

Comparator-clause

::= COMPARE Target-name Where-clause Contrasting-classes
WITH RESPECT TO Attribute-name-list

Contrasting-classes
::= IN CONTRAST TO Contrast-name Where-clause
{IN CONTRAST TO Contrast-name Where-clause}

Target-name
::= *string*

Contrast-name
::= *string*

Associator-clause
::= MINE ASSOCIATION AS *string* WITH RESPECT TO Attribute-association-list
[Meta-rule-clause]

Attribute-association-list
::= * | Attribute-name [AS *string*]{, Attribute-name [AS *string*]}

Classifier-clause
::= MINE CLASSIFICATION AS *string* FOR Attribute-name
WITH RESPECT TO Attribute-name-list

Clusterer-clause
::= MINE CLUSTERING AS *string* FOR Attribute-name
WITH RESPECT TO Attribute-name-list

Predictor-clause
::= MINE PREDICTION AS *string* FOR Attribute-name
WITH RESPECT TO Attribute-name-list

```
S-Threshold-clause
    ::=  S-Threshold-specification{, S-Threshold-specification}


S-Threshold-specification
    ::=  [SET] SUPPORT [THRESHOLD] Threshold-value
    |    [SET] DISTINCT-VALUE [THRESHOLD] Threshold-value [FOR Attribute-name]


A-Threshold-clause
    ::=  A-Threshold-specification{, A-Threshold-specification}


A-Threshold-specification
    ::=  [SET] A-Threshold-name [THRESHOLD] Threshold-value


A-Threshold-name
    ::=  SUPPORT | CONFIDENCE | INTERESTING


Meta-rule-clause
    ::=  MATCHING Predicate-list "=>" Predicate-list


Predicate-list
    ::=  Predicate {AND Predicate}


Predicate
    ::=  string(identifier-VAR, Constant)


C-Threshold-clause
    ::=  C-Threshold-specification{, C-Threshold-specification}
```

C-Threshold-specification

::= [SET] C-Threshold-name [THRESHOLD] Threshold-value

C-Threshold-name

::= CLASSIFICATION | NOISE | TRAINING-SET

T-Threshold-clause

::= [SET] TRAINING-SET [THRESHOLD] Threshold-value

Threshold-value

::= *integer* | *float*

Show-clause

::= SHOW Output

Output

::= RULES | TABLES | GRAPHS

Note that the syntax presented above is very permissive. It allows the generation of some constructs that are not legal or semantically incorrect in DMQL. We have simplified the grammar for clarity, to avoid complicated production rules due to the context-sensitivity of DMQL.

Knowledge is a rich storehouse for the glory of the Creator and the relief of man's estate.

Francis Bacon

Knowledge comes, but wisdom lingers

Alfred Lord Tennyson

Appendix D

Defining Metadata for the Internet

The most common definition of the term "metadata" is *data about data*. Metadata for documents would greatly help in the indexing process and improve relevancy of information retrieval. Most on-line documents today have no specific metadata with them. However, in HTML 2.0, new tags were introduced to allow web developers and authors to identify document creators, keywords, description, and even specify web agent behaviours regarding some documents. In the first section, we enumerate some of these tags and variables that allow to define metadata for on-line documents.

The second section gives a document type definition for XML-based documents using the **Dublin Core Metadata element set**.

D.1 META tags in HTML

HTML 2.0 defines the META element, a tag type which allows a limited ability to describe a particular document. META tags have two possible attributes:

1. <META HTTP-EQUIV="*name*" CONTENT="*content*">

2. <META NAME="*name*" CONTENT="*content*">

The corresponding structure of the META element is defined by the following DTD (Document Type Definition):
<!ELEMENT META - O EMPTY>
<!ATTLIST META

HTTP-EQUIV NAME #IMPLIED
NAME NAME #IMPLIED
CONTENT CDATA #REQUIRED >

META tags are placed in the head of the HTML document, between the <HEAD> and </HEAD> tags. The major difference between HTTP-EQUIV and NAME is that HTTP-EQUIV is to define variable destined to be part of the HTTP response header[1]. Variable defined in this form have an equivalent effect as when they are specified directly in the HTTP header. Some Web servers actually translate HTTP-EQUIV META tags into actual HTTP headers automatically. META tags with a NAME attribute are used for other types which do not correspond to HTTP headers. The CONTENT element contains the associated data to the variable named in NAME or HTTP-EQUIV.

D.1.1 Examples of HTTP-EQUIV META Tags in HTML

Here are some examples of the variables defined in the META tag with HTTP-EQUIV attribute (conform to RFC1945 and RFC2068):

- **Content-Language** may be used to declare the natural language of a document, example: <META HTTP-EQUIV="Content-Language" CONTENT="en">
- **Content-Length** is used to specify the size of a document in bytes.
- **Content-Location** is used for the URL of the resource.
- **Content-Type** specifies the media type and can be extend to give the character set, example:
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-2022-JP">
- **Content-Version** may be used to indicate the version of the evolving document.
- **Expires** is used to declare the expiry date of the document. This date is used to control caching and update indexes, example:
  <META HTTP-EQUIV="Expire" CONTENT="Fri, 26 Mar 1999 09:30:57 GMT">
- **Last-Modified** indicates the date the document was last modified.

[1] HTTP headers are defined in RFC1945 (HTTP/1.0) and RFC2068 (HTTP/1.1).

- **Link** is used to indicate relationships to other resources.

- **PICS-Label** is for document content labeling. PICS stands for Platform for Internet Content Selection[2] and the tag is used to declare a document's rating in terms of adult content (sex, violence, etc.), for example: <META HTTP-EQUIV='PICS-Label" CONTENT="(PICS-1.1 "http://www.picsservice.org/v1.0" labels on "1999.01.05T09:30-0500" for "http://www.site.com/mypage.html" ratings (s 0 v 0 g 0))'>

- **Pragma** allows the document content to stay current by preventing caching by browsers and web agents, example: <META HTTP-EQUIV="Pragma" CONTENT="no-cache">

- **Refresh** specifies the time in seconds before a Web browser (or agent) reloads the document automatically. It can also specify an alternative URL to load, example: <META HTTP-EQUIV="Refresh" CONTENT="5; URL="http:www.somewhere.ca/document.html">

D.1.2 Examples of NAME META Tags in HTML

Here are some examples of the variables defined in the META tag with NAME attribute. Except for the Dublin Core elements, these are not standardized and are suggested by a variety of companies and organization.

1. Dublin Core: there are 15 elements in the Dublin Core. They can be used in the HTML META tags by adding the DC prefix to their label: **DC.TITLE, DC.CREATOR, DC.SUBJECT, DC.DESCRIPTION, DC.PUBLISHER, DC.CONTRIBUTOR, DC.DATE, DC.TYPE, DC.FORMAT, DC.IDENTIFIER, DC.SOURCE, DC.LANGUAGE, DC.RELATION, DC.COVERAGE,** and **DC.RIGHTS**. See Section 2.3.1 in Chapter 3 for more details.

2. Other variables

    - **Robots** controls crawling robots on a per-page basis. Normally, well behaved crawling robots consult the *robots.txt* file to verify indexing permission on a site. This tab allows to specify to visiting crawlers how they should behave with regard to the document. Values can be *all, none, index, noindex, follow* and *nofollow.* For example, to allow a crawler to index the document but not to follow its

[2] see PICS standard at http://www.w3c.org/PICS.

hyperlinks, the following tag is needed:
<META NAME="Robots" CONTENT="index | nofollow">

- **Keywords** is used to enumerate important keywords and synonyms associated with the document. They can be used and are given priority by search engines for indexing or given weights to keywords, for example:
  <META NAME="Keywords" CONTENT="Information Retrieval, Data Mining">
- **Description** can be used to describe the content of the document in plain text. Search engines index it and display it as a snippet related to the document.
- **Author** is used to qualify the document's author.
- **Contact** is used to specify the authors e-mail address.
- **Location** can be used to specify the geographical location like country, province, city, etc.
- **Note** is used to add any supplemental information in plain text.
- **Copyright** indicates the document copyright statement.

There are many other variables for the META NAME tag suggested by different applications such Microsoft Office (**Generator, Editor, Language, Office, Publisher, Project, Status, Subject, Date-Completed**, etc.) and others. It is also important to notice that there are many different variables with the same semantics, for example **DateofLastModification, Date-Completed, LastUpdated** and **timestamp** specify the date of the last update.

D.2 Example of Web Document Described with Dublin Core

As an example of adding metadata in web documents, we make use of the Dublin Core elements to describe the information pertaining to the document. These elements are intended to support access to information on the World Wide Web or other digital libraries. The excerpt shown bellow, contains some descriptive metadata, invisibly embedded in the HTML using the <META> tag in the <HEAD> section of the document.
<META NAME="package" CONTENT="(TYPE=begin)(VERSION=0.1) Dublin Core">
<META NAME="DC.title" CONTENT="Resource and Knowledge Discovery From the Internet and Multimedia Repositories">

```
<META NAME=”DC.subject” CONTENT=”Data Mining, WWW, Knowledge Discovery, Visual
Data, Content-Based Retrieval”>
<META NAME=”DC.creator” CONTENT=”(TYPE=name) Osmar R. Zaiane”>
<META NAME=”DC.creator” CONTENT=”(TYPE=email) zaianecs.sfu.ca”>
<META NAME=”DC.creator” CONTENT=”(TYPE=affiliation) Simon Fraser University”>
<META NAME=”DC.creator” CONTENT=”(TYPE=homepage) http://www.cs.sfu.ca/~zaiane”>
<META NAME=DC.Description CONTENT=”In this thesis, the inefficiency and inadequacy of
the current information retrieval technology applied on the Internet is demonstrated. A framework,
called Virtual Web Views, for intelligent interactive information retrieval and knowledge discovery
from global information systems is proposed, and a query language, WebML, for resource discovery
and data mining from the Web using the virtual web views is put forward. it is also illustrate how
descriptors collected for virtual web view building can be exploited for content-based image retrieval,
and how to carry out on-line analytical processing and data mining on visual data from the World-
Wide Web, or other multimedia repositories.”>
<META NAME=”DC.date” CONTENT=”(TYPE=creation) (SCHEME=ISO31) 1999-01-12”>
<META NAME=”DC.form” CONTENT=”(SCHEME=imt) text/html”>
<META NAME=”DC.identifier” CONTENT=”(TYPE=url) http://www.cs.sfu.ca/~zaiane/thesis.html”>
<META NAME=”DC.language” CONTENT=”(SCHEME=iso639) en-GB”>
<META NAME=”DC.Rights” CONTENT=”http://www.cs.sfu.ca/~zaiane/rights.html”>
<LINK REL=SCHEMA.dc HREF=”http://purl.org/metadata/dublin_core/elements#title”>
<LINK REL=SCHEMA.dc HREF=”http://purl.org/metadata/dublin_core/elements#subject”>
<LINK REL=SCHEMA.dc HREF=”http://purl.org/metadata/dublin_core/elements#creator”>
<LINK REL=SCHEMA.dc HREF=”http://purl.org/metadata/dublin_core/elements#date”>
<LINK REL=SCHEMA.dc HREF=”http://purl.org/metadata/dublin_core/elements#form”>
<LINK REL=SCHEMA.dc HREF=”http://purl.org/metadata/dublin_core/elements#Description”>
<LINK REL=SCHEMA.imt HREF=”http://sunsite.auc.dk/RFC/rfc/rfc1521.html”>
<LINK REL=SCHEMA.dc HREF=”http://purl.org/metadata/dublin_core/elements#identifier”>
<LINK REL=SCHEMA.dc HREF=”http://purl.org/metadata/dublin_core/elements#language”>
<META NAME=”package” CONTENT=”(TYPE=end)(VERSION=0.1) Dublin Core”>
```

D.3 Dublin Core XML DTD Fragment

The Dublin Core is a set of 15 key metadata elements, the form of which has been agreed upon and has become the basis of a standard for metadata on the WWW. The 15-element metadata element set is intended to facilitate discovery of electronic resources. The 15 elements are: CREATOR, TITLE, SUBJECT, DESCRIPTION, PUBLISHER, CONTRIBUTOR, DATE, TYPE, FORMAT, IDENTIFIER, SOURCE, LANGUAGE, RELATION, COVERAGE, and RIGHTS. An explanation of these elements can be found in Section 2.3.1 of Chapter 3.

The following is a DTD (document type definition) for XML documents using the Dublin Core elements.

<!– Beginning of metadata element declarations –>

<!– $Id: XML.dtd,v 2.0 1999/01/12 Osmar Zaiane, modified from v 1.1 jon Exp$ –>

<!– The METAPACKAGE element is intended to be the outer grouping element for a whole set of related metadata from a single metadata schema (such as Dublin Core). It implicitly forms an AND grouping of its child elements. The default schema attribute value should be taken by applications as "DublinCore" and the default version as "1.0". –>

<!ELEMENT METAPACKAGE (ANDGROUP* | ORGROUP* | CREATOR* | TITLE* | SUBJECT | DESCRIPTION* | PUBLISHER* | CONTRIBUTOR* | DATE* | TYPE* | FORMAT* | IDENTIFIER* | SOURCE* | LANGUAGE* | RELATION* | COVERAGE* | RIGHTS* | META*)>

<!ATTLIST METAPACKAGE SCHEMA CDATA #IMPLIED VERSION CDATA #IMPLIED>

<!– The ANDGROUP element is used to explicitly form a conjunction between its child elements. –>

<!ELEMENT ANDGROUP (ANDGROUP* | ORGROUP* | CREATOR* | TITLE* | SUBJECT | DESCRIPTION* | PUBLISHER* | CONTRIBUTOR* | DATE* | TYPE* | FORMAT* | IDENTIFIER* | SOURCE* | LANGUAGE* | RELATION* | COVERAGE* | RIGHTS* | META*)>

<!– The ORGROUP element is used to explicitly form a disjunction between the child elements within it –>

<!ELEMENT ORGROUP (ANDGROUP* | ORGROUP* | CREATOR* | TITLE* | SUBJECT | DESCRIPTION* | PUBLISHER* | CONTRIBUTOR* | DATE* | TYPE* | FORMAT* | IDENTIFIER* | SOURCE* | LANGUAGE* | RELATION* | COVERAGE* | RIGHTS* | META*)>

<!– Now we get the actual metadata elements themselves. Here the elements are named after the Dublin Core v1.0 element set names. Note that the all take the same set of attributes (relating to their qualifiers) _except_ for contributor that also requires a role attribute as well. –>

<!ELEMENT TITLE (#PCDATA)>

<!ATTLIST TITLE SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET CDATA

```
#IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT CREATOR (#PCDATA)>
<!ATTLIST CREATOR SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET
CDATA #IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT SUBJECT (#PCDATA)>
<!ATTLIST SUBJECT SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET
CDATA #IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ATTLIST DESCRIPTION SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET
CDATA #IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT PUBLISHER (#PCDATA)>
<!ATTLIST PUBLISHER SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET
CDATA #IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT DATE (#PCDATA)>
<!ATTLIST DATE SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET CDATA
#IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT TYPE (#PCDATA)>
<!ATTLIST TYPE SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET CDATA
#IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT FORMAT (#PCDATA)>
<!ATTLIST FORMAT SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET
CDATA #IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT IDENTIFIER (#PCDATA)>
<!ATTLIST IDENTIFIER SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET
CDATA #IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT SOURCE (#PCDATA)>
<!ATTLIST SOURCE SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET CDATA
#IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT LANGUAGE (#PCDATA)>
<!ATTLIST LANGUAGE SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET
CDATA #IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT RELATION (#PCDATA)>
<!ATTLIST RELATION SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET
CDATA #IMPLIED LANGUAGE CDATA #IMPLIED ROLE CDATA #IMPLIED>
<!ELEMENT COVERAGE (#PCDATA)>
<!ATTLIST COVERAGE SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET
CDATA #IMPLIED LANGUAGE CDATA #IMPLIED>
```

```
<!ELEMENT RIGHTS (#PCDATA)>
<!ATTLIST RIGHTS SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET CDATA #IMPLIED LANGUAGE CDATA #IMPLIED>
<!ELEMENT CONTRIBUTOR (#PCDATA)>
<!ATTLIST CONTRIBUTOR SCHEME CDATA #IMPLIED TYPE CDATA #IMPLIED CHARSET CDATA #IMPLIED LANGUAGE CDATA #IMPLIED ROLE CDATA #IMPLIED>
<!– This is similar to the HTML 2.0/3.2 META element definition and is included for some limited backwards compatibility and use with non-DC like metadata schemes. –>
<!ELEMENT META EMPTY>
<!ATTLIST NAME CDATA #IMPLIED HTTP-EQUIV CDATA #IMPLIED CONTENT CDATA #IMPLIED>
<!– End of metadata declarations –>
```

Bibliography

[1] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The lorel query language for semistructured data, 1997. http://www-db.stanford.edu/~abitebou/pub/jodl97.lorel96.ps.

[2] S. Abitibout. Querying semi-structured data. In *Int. Conf. on Database Theory*, 1997.

[3] Mark S. Ackerman, Brian Starr, and Michael Pazzani. The do-i-care agent: Effective social discovery and filtering on the web. In *Proc. RIAO'97(Computer Assisted Information Searching on the Internet)*, Montreal, Canada, 1997.

[4] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Proc. 4th Int. Conf. Foundations of Data Organization and Algorithms*, Chicago, October 1993.

[5] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An interval classifier for database mining applications. In *Proc. 18th Int. Conf. Very Large Data Bases*, pages 560–573, Vancouver, Canada, August 1992.

[6] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data*, pages 207–216, Washington, D.C., May 1993.

[7] R. Agrawal and J. C. Shafer. Parallel mining of association rules: Design, implementation, and experience. *IEEE Trans. Knowledge and Data Engineering*, 8:962–969, 1996.

[8] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 487–499, Santiago, Chile, September 1994.

[9] A.V. Aho and M.J. Corasick. Fast pattern matching: an aid to bibliographic search. *Communications of ACM*, 18(6):333–340, June 1975.

[10] AI Magazine, 18(2). *Intelligent Systems on the Internet*, 1997.

[11] AI Magazine, 19(2). *Intelligent Agents*, 1998.

[12] P. Aigrain, H. Zhang, and D. Petkovic. Content-based representation and retrieval of visual media: A state-of-the-art review. *Int. J. Multimedia Tools and Applications*, 3:179–202, November 1996.

[13] Alex ftp filesystem. ftp://alex.sp.cs.cmu.edu/usr0/anon/www/alex.html.

[14] K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. In *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining (KDD'97)*, Newport Beach, California, August 1997.

[15] R. Alonso, D. Barbara, and H. Garcia-Molina. Data caching issues in an information retrieval system. In *ACM Transactions on Database Systems*, pages 359–384, 1990.

[16] American National Standards Institute. *Database Language SQL*, ansi x3.135-1992 edition, 1992.

[17] Yigal Arens, Chun-Nan Hsu, and Craig A. Knoblock. Query processing in the sims information mediator. In *ARPA/Rome Laboratory Knowledge-Based Planning and Scheduling Initiative Workshop*, 1996.

[18] Roberts Armstrong, Dayne Freitag, Thorsten Joachims, and Tom Mitchell. WebWatcher: A learning apprentice for the world wide web. In *AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, March 1995.

[19] Gustavo O. Arocena and Alberto O. Mendelzon. WebOQL: Restructuring documents, databases and webs. In *Proc of ICDE Conf.*, Orlando, Florida, USA, February 1998.

[20] P. Atzeni, G. Mecca, and P. Merialdo. Semistructured and structured data in the web: Going back and forth. In *Proc. Workshop on Semi-structured Data*, Tucson, Arizona, May 1997.

[21] J.R. Bach, C. Fuller, A. Gupta, and et al. The Virage image search engine: An open framework for image management. In *SPIE Storage and Retrieval for Image and Video Databases IV*, February 1996.

[22] D. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[23] D.H. Ballard and C.M. Brown. *Computer Vision.* Prentice Hall, 1982.

[24] E. Baralis and G. Psaila. Designing templates for mining association rules. *Journal of Intelligent Information Systems*, 9:7–32, 1997.

[25] R. Beckwith, C. Fellbaum, D. Gross, K. Miller, G.A. Miller, and R. Tengi. Five papers on WordNet. *Special Issue of Journal of Lexicography*, 3(4):235–312, 1990. ftp://ftp.cogsci.princeton.edu/pub/wordnet/5papers.ps.

[26] Gordon Bell and Jim Gemmell. On-ramp prospects for the information superhighway dream. *Communications of the ACM*, 39(7):55–61, 1996.

[27] Tim Berners-Lee. The world wide web initiative. CERN, http://info.cern.ch/hypertext/WWW/TheProject.html.

[28] Tim Berners-Lee, Robert Caillau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret. The world-wide web. *Communication of the ACM*, 37(8):76–82, August 1994.

[29] Phil Bernstein, Michael Brodie, Stefano Ceri, David DeWitt, Mike Franklin, Hector Garcia-Molina, Jim Gray, Jerry Held, Joe Hellerstein, H. V. Jagadish, Michael Lesk, Dave Maier, Jeff Naughton, Hamid Pirahesh, Mike Stonebraker, and Jeff Ullman. The asilomar report on database research. *ACM Sigmod Record*, 27(4), December 1998. also available at: http://www.acm.org/sigmod/record/issues/9812/asilomar.html.

[30] I. Bhandari, E. Colet, J. Parker, Z. Pines, and R. Pratap. Advanced scout: Data mining and knowledge discovery in NBA data. *Data Mining and Knowledge Discovery*, 1(1):121–125, 1997.

[31] Michael Bieber. Providing information systems with full hypermedia functionality. In *Proc. 26th Hawaii Int. Conf. on System Sciences*, 1993.

[32] Eric Bina, Vicki Jones, Rob McCool, and Marianne Winslett. Secure access to data over the internet. http://bunny.cs.uiuc.edu/CADR/winslett/pubs/SecureDBAccess.ps.

[33] B. Bloom. Space time tradeoffs in hash coding with allowable errors. *Communications of ACM*, 13(7):422–426, 1970.

[34] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS-732-94, University of Colorado, 1994. ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Harvest.FullTR.ps.Z.

[35] C. Mic Bowman, Peter B. Danzig, Udi Manber, and Michael F. Swartz. Scalable internet resource discovery: Research problems and approaches. *Communication of the ACM*, 37(8):98–107, August 1994.

[36] R.S. Boyer and J.S. Moore. A fast string searching algorithm. *Communications of ACM*, 20(10):762–772, October 1977.

[37] P.M.E. De Bra and R.D.J. Post. Information retrieval in the world-wide web: Making client-based searching feasible.

[38] Jeffrey M. Bradshaw. *Software Agents*. AAAI Press / The MIT Press, 1997.

[39] S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. In *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data*, pages 265–276, Tucson, Arizona, May 1997.

[40] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *7th Int. Conf. WWW*, Brisbane, Australia, April 1998.

[41] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proc. ACM SIGMOD Conf. on Management of Data*, 1996.

[42] Peter Buneman. Semistructured data, tutorial in principal of database systems conference. http://www.cis.upenn.edu/~db, 1997.

[43] P.J. Burt. Smart sensing within a pyramid vision machine. *Proceedings of IEEE*, 76(8):1006–1015, 1988.

[44] Vannevar Bush. As we may think. *The Atlantic Monthly*, 176(1):101–108, July 1945.

[45] Y. Cai, N. Cercone, and J. Han. Attribute-oriented induction in relational databases. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 213–228. AAAI/MIT Press Also in Proc. IJCAI-89 Workshop Knowledge Discovery in Databases, Detroit, MI, August 1989, 26-36., 1991.

[46] S. Chakrabarti, B. Dom, D. Gibson, S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in topic distillation. In *ACM SIGIR workshop on Hypertext Information Retrieval on the Web*, Melbourne, Australia, 1998.

[47] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record*, 26:65–74, 1997.

[48] S. Chaudhuri, S. Ghandeharizadeh, and C. Shahabi. Avoiding retrieval content for composite multimedia objects. In *Proc. 21st Int. Conf. on Very Large Data Bases (VLDB)*, pages 287–298, 1995.

[49] Hsinchun Chen, Andrea Houston, Jay Nunamaker, and Jerome Yen. Toward intelligent meeting agent. *IEEE Computer*, 29(8):62–70, August 1996.

[50] M. S. Chen, J. S. Park, and P.S. Yu. Data mining for path traversial patterns in a web environment. In *Proc. 16th Int. Conf. Distributed Computing Systems*, pages 385–392, May 1996.

[51] D.W. Cheung, J. Han, V. Ng, A. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In *Proc. 1996 Int. Conf. Parallel and Distributed Information Systems*, pages 31–44, Miami Beach, Florida, Dec. 1996.

[52] D.W. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proc. 1996 Int. Conf. Data Engineering*, pages 106–114, New Orleans, Louisiana, Feb. 1996.

[53] D.W. Cheung, V. Ng, A. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. *IEEE Trans. Knowledge and Data Engineering*, 8:911–922, 1996.

[54] S. Chien, F. Fisher, H. Mortensen, E. Lo, and R. Greeley. Using artificial intelligence planning to automate science data analysis for large image databases. In *Proc. Third Int. Conf. on Knowledge Discovery and Data Mining*, pages 147–150, 1997.

[55] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. Efficient crawling through URL ordering. In *7th Int. Conf. WWW*, Brisbane, Australia, April 1998.

[56] W. W. Chu, Q. Chen, and R. Lee. Cooperative query answering via type abstraction hierarchy. In S.M Dee, editor, *Cooperating Knowledge Based System*, pages 271–292. Now York: Elsevier, 1990.

[57] Kimberly C. Claffy, Hans-Werner Braun, and George C. Polyzos. Tracking long-term growth of the NSFNET. *Communication of the ACM*, 37(8):34–45, August 1994.

[58] E. F. Codd. A relational model for large shared data banks. *Communication of the ACM*, 13(6):377–387, June 1970.

[59] Jeff Conklin. Hypertext: An introduction and survey. *IEEE ComputerDatabase Engineering*, 20(9):17–41, September 1987.

[60] Dan Connolly and Jon Bosak World-Wild Web Consortium. Extensible markup language-xml, April 1997. http://www.w3c.org/XML.

[61] W. Bruce Croft and Howard Turtle. A retrieval model for incorporating hypertext links. In *Proc. Hypertext89*, pages 213–224, November 1989.

[62] F. Cuppens and R. Demolombe. Cooperative answering: A methodology to provide intelligent access to databases. In *Proc. 2nd Int. Conf. Expert Database Systems*, pages 621–643, Fairfax, VA, April 1988.

[63] A. Czyzewski. Mining knowledge in noisy audio data. In *Proc. Second Int. Conf. on Knowledge Discovery and Data Mining*, pages 220–225, 1996.

[64] Jody J. Daniels and Edwardina L. Rissland. A case-based approach to intelligent information retrieval. In *Proc. ACM SIGIR'95 Conf.*, Seattle, WA, USA, 1995.

[65] Jody J. Daniels and Edwina L. Rissland. A case-based approach to intelligent information retrieval. In *Proc. SIGIR 95 Conf.*, pages 238–245, Seattle, WA, USA, 1995.

[66] P. Danzig, K. Obraczka, D. DeLucia, and N. Alam. Massively replicating services in autonomously managed wide-area internetworks. Technical report, Technical report, 1994. Available from *ftp://catarina.usc.edu/pub/kobraczk/ToN.ps.Z.*

[67] P. B. Danzig, R. S. Hall, and M. F. Schwartz. A case for caching file objects inside internetworks. In *Proc. SIGCOMM'93*, pages 239–248, ACM, New York, September 1993.

[68] B. de Ville. Applying statistical knowledge to database analysis and knowledge base construction. In *Proc. 6th Conf. on Artificial Intelligence Applications*, pages 30–36, Santa Barbara, CA, 1990.

[69] Deutsch, Emtage, and Heelan. Archie - an electronic directory service for the internet. ftp://archie.ans.net/pub/archie/doc/whatis.archie.

[70] Martin Dillon, Erik Jul, Mark Burge, and Carol Hickey. Assessing information on the internet: Toward providing library services for computer-mediated communication. *Internet Research*, 3(1), Spring 1993.

[71] R. Doorendos, O. Etzioni, and D. Weld. A scalable comparison-shopping agent for the world-wide web. In *Proc. Autonomous Agents ACM*, 1997.

[72] M.S. Drew, J. Wei, and Z.N. Li. Illumination–invariant color object recognition via compressed chromaticity histograms of color–channel–normalized images. In *Proc. Int. Conf. on Computer Vision (ICCV '98)*, pages 533–540, 1998.

[73] E.H. Durfee, D.L. Kiskis, and W.P. Birmingham. The agent architecture of the University of Michigan digital library. *IEEE Software Engineering*, 144(1):61–71, February 1997.

[74] Mark T. Maybury Editor. *Intelligent Multimedia Information Retrieval.* The AAAI Press/The MIT Press, 1997.

[75] Max J. Egenhofer. *Spatial Query Languages.* PhD thesis, University of Maine, 1989.

[76] Max J. Egenhofer and Jayant Sharma. Topological relations between regions in r^2 and z^2. In *Advances in Spatial Databases (SSD'93)*, Singapore, 1993.

[77] David Eichmann. The RBSE spider - balancing effective search against web load. In *Proc. 1st WWW Conf.*, May 1994.

[78] David Eichmann, Terry McGregor, and Dann Danley. Integrating structured databases into the web: The MORE system. http://rbse.jse.nasa.gov:81/.

[79] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems, 2nd ed.* Bemjamin/Cummings, 1994.

[80] Hans Erickson. Mbone: The multicast backbone. *Communication of the ACM*, 37(8):54–60, August 1994.

[81] Oren Etzioni. The wold-wide web: Quagmire or gold mine? *Communications of the ACM*, 39(11):65–68, 1996.

[82] Oren Etzioni. Moving up the information food chain: Deploying softbots on the world wide web. *AI Magazine*, 18(2), 1997.

[83] Oren Etzioni, Neal Lesh, and Richard Segal. Building softbot for unix, November 1992. Preliminary report.

[84] Oren Etzioni and Daniel Weld. A softbot-based interface to the internet. *Communications of the ACM*, 37(7):72–76, 1994.

[85] Oren Etzioni and Daniel Weld. Intelligent agents on the internet: Fact, fiction, and forecast, May 1995.

[86] Andrew Fall. *Reasoning with Taxonomies*. PhD thesis, School of Computing Science, Simon Fraser University, December 1996.

[87] Andrew Fall. The foundations of taxonomic encoding. *Computational Intelligence*, 14(4):598–642, 1998.

[88] Christos Faloutsos. Access methods for text. *Computing Surveys*, 17(1):49–74, March 1985.

[89] Christos Faloutsos and Douglas W. Oard. A survey of information retrieval and filtering methods. Technical Report CS-TR-3514, University of Maryland, August 1995. http://www.enee.umd.edu/medlab/filter/papers/survey.ps.

[90] U. Fayyad and P. Smyth. Image database exploration: Progress and challenges. In *Proc. Knowledge Discovery in Databases Workshop*, pages 14–27, Washington, D.C, 1993.

[91] U. M. Fayyad, S. G. Djorgovski, and N. Weir. Automating the analysis and cataloging of sky surveys. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 471–493. AAAI/MIT Press, 1996.

[92] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.). *Advances in Knowledge Discovery and Data Mining.* AAAI/MIT Press, 1996.

[93] R. Feldman and I. Dagan. Knowledge discovery in textual databases (KDT). In *Proc. 1st Int. Conf. Knowledge Discovery and Data Mining*, pages 112–117, Montreal, Canada, Aug. 1995.

[94] R. Feldman and H. Hirsh. Mining associations in text in the presence of background knowledge. In *Proc. 2st Int. Conf. Knowledge Discovery and Data Mining*, pages 343–346, Portland, Oregon, Aug. 1996.

[95] Mary Fernandez, Daniela Florescu, Jaewoo Kang, Alon Levy, and Dan Suciu. Catching the boat with strudel: Experiences with a web-site management system. In *Proc. ACM SIGMOD Conf. on Management of Data*, Seattle, WA, June 1998.

[96] D. Fisher. Improving inference through conceptual clustering. In *Proc. 1987 AAAI Conf.*, pages 461–465, Seattle, Washington, July 1987.

[97] David Flater and Yelena Yesha. Alibi: A novel approach to resource discovery. *Journal of Internet research*, 1995.

[98] Jonathon Fletcher. Internet robots - structure from anarchy?, 1994. JumpStation Front Page: http://www.stir.ac.uk/jf1bin/js.

[99] M. Flickner, H. Sawhney, W. Niblack, and et al. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

[100] Daniela Florescu, Alon Levy, and Alberto Mendelzon. Database techniques for the world-wide web: A survey. *ACM SIGMOD Record*, 27(3):59–74, September 1998.

[101] C. Frankel, M. Swain, and V. Athitsos. Webseer: An image search engine for the World Wide Web. Technical Report TR-96-14, CS Department, Univ. of Chicago, 1996.

[102] C. Frankel, M. J. Swain, and V. Athitsos. Webseer: An image search engine for the world wide web. Technical Report 96-14, University of Chicago, Computer Science Department, August 1996.

[103] Jeff Frentzen. Meta tags can index, organize your web pages. *PCWeek Online*, 1996.

[104] Jeff Frentzen. Web rings: A novel way to get around the net. *PCWeek Online*, 1996.

[105] Y. Fu and J. Han. Meta-rule-guided mining of association rules in relational databases. In *Proc. 1st Int. Workshop Integration of Knowledge Discovery with Deductive and Object-Oriented Databases (KDOOD'95)*, pages 39–46, Singapore, Dec. 1995.

[106] Yiangjian Fu. *Discovery of Multiple-level Rules from Large Databases*. PhD thesis, School of Computing Science, Simon Fraser University, July 1996.

[107] Norbert Fuhr and Ulrich Pfeifer. Probalistic information retrieval as a combination of abstraction, inductive learning, and probalistic assumptions. *ACM Transactions on Information System*, 12(1):92–115, 1994.

[108] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data*, pages 13–23, Montreal, Canada, June 1996.

[109] R. Fuller and J. de Graaff. Measuring user motivation from server log files. In *http://www.microsoft.com/usability/webconf/fuller/fuller.htm*, 1997.

[110] B.V. Funt and G.D. Finlayson. Color constant color indexing. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 17:522–529, 1995.

[111] G. Furnas, T.K. Landauer, L.M. Gomez, and S. Dumais. The vocabulary problem in human-system communications. *Communications of the ACM*, 30:964–971, 1987.

[112] Athula Ginige, David B. Lowe, and John Robertson. Hypermedia authoring. *IEEE Multimedia*, pages 24–34, 1995.

[113] J. Graham-Cumming. Hits and miss-es: A year watching the web. In *Proc. 6th Int. World Wide Web Conf.*, Santa Clara, California, April 1997.

[114] Andrew S. Grimshaw and Wm.A. Wulf. The legion vison of a worldwide virtual computer. *Communications of the ACM*, 40(1):39–45, 1997.

[115] David A. Grossman and Ophir Frieder. *Information Retrieval: Algorithms and Heuristics.* Kluwer Academic Publishers, 1998.

[116] V. Gudivada and V. Raghavan. Content-based image retrieval systems. *IEEE Computer*, 28(9):18–22, 1995.

[117] Ralf Hartmut Güting. In introduction to spatial database systems. *The VLDB Journal*, 3(4):357–399, 1994.

[118] P. Gvozdjak and Z.N. Li. From nomad to explorer: Active object recognition on mobile robots. *Pattern Recognition*, 31(6):773–790, 1998.

[119] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. Knowledge and Data Engineering*, 5:29–40, 1993.

[120] J. Han, J. Chiang, S. Chee, J. Chen, Q. Chen, S. Cheng, W. Gong, M. Kamber, G. Liu, K. Koperski, Y. Lu, N. Stefanovic, L. Winstone, B. Xia, O. R. Zaïane, S. Zhang, and H. Zhu. DBMiner: A system for data mining in relational databases and data warehouses. In *Proc. CASCON'97: Meeting of Minds*, pages 249–260, Toronto, Canada, November 1997.

[121] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. 1995 Int. Conf. Very Large Data Bases*, pages 420–431, Zurich, Switzerland, Sept. 1995.

[122] J. Han and Y. Fu. Exploration of the power of attribute-oriented induction in data mining. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 399–421. AAAI/MIT Press, 1996.

[123] J. Han, Y. Fu, Y. Huang, Y. Cai, and N. Cercone. DBLearn: A system prototype for knowledge discovery in relational databases. In *Proc. 1994 ACM-SIGMOD Conf. Management of Data*, page 516, Minneapolis, MN, May 1994.

[124] J. Han, Y. Fu, and R. Ng. Cooperative query answering using multi-layered databases. In *Proc. 2nd Int. Conf. Cooperative Information Systems*, pages 47–58, Toronto, Canada, May 1994.

[125] J. Han, Y. Fu, W. Wang, J. Chiang, W. Gong, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, B. Xia, and O. R. Zaïane. DBMiner: A system for mining knowledge in large relational databases. In *Proc. 1996 Int. Conf. Data Mining and Knowledge Discovery (KDD'96)*, pages 250–255, Portland, Oregon, August 1996.

[126] J. Han, Y. Fu, W. Wang, K. Koperski, and O. R. Zaïane. DMQL: A data mining query language for relational databases. In *Proc. 1996 SIGMOD'96 Workshop Research Issues on Data Mining and Knowledge Discovery (DMKD'96)*, pages 27–34, Montreal, Canada, June 1996.

[127] Jiawei Han, Osmar R. Zaïane, and Yongjian Fu. Resource and knowledge discovery in global information systems: A multiple layered database approach. Technical Report TR94-24, School of Computing Science, Simon Fraser University, November 1994.

[128] Jiawei Han, Osmar R. Zaïane, and Yongjian Fu. Resource and knowledge discovery in global information systems: A scalable multiple layered database approach. In *In Proc. Conf. on Advances in Digital Libraries*, Washington, DC, May 1995.

[129] D. Hardy and M. F. Schwartz. Essence: A resource discovery system based on semantic file indexing. In *Proc. of the USENIX Winter Conf.*, pages 361–374, Berkeley, CA, 1993. ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Essence.Conf.ps.Z.

[130] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data*, pages 205–216, Montreal, Canada, June 1996.

[131] R.L. Haskin. Special-purpose processors for text retrieval. *Database Engineering*, 4(1):16–29, September 1991.

[132] K. Hirata, Y. Hara, N. Shibata, and F. Hirabayashi. Media-based navigation for hypermedia systems. In *Proceedings of ACM Hupertext'93 Conf.*, pages 159–173, Seattle, WA, 1993.

[133] N. Hirzalla and A. Karmouch. Detecting cuts by understanding camera operations for video indexing. *Journal of Visual Languages and Computing*, 6:385–404, 1995.

[134] T.H. Hong and A. Rosenfeld. Compact region extraction using weighted pixel linking in a pyramid. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-6(2):222–229, 1984.

[135] Adele E. Howe and Daniel Dreillinger. Savvysearch: A metasearch engine that learns which search engine to query. *AI Magazine*, 18(2), 1997.

[136] W. Hsu, T.S. Chua, and H.K. Pung. An integrated color-spatial approach to content-based image retrieval. In *Proc. ACM Multimedia '95*, pages 305–313, 1995.

[137] Michael N. Huhn, Munindar P. Singh, and Tomasz Ksiezyk. Global information management via local autonomous agents. In *Proc. ICOT Int. Symposium on Fifth Generation Computer Systems: Workshop on Heterogeneous Cooperative Knowledge Bases*, 1994.

[138] Michael N. Huhns and Munindar P. Singh. *Readings in Agents.* Morgan Kaufmann Publishers, 1998.

[139] The inktomi technology behind hotbot - a white paper. http://www.inktomi.com, 1996.

[140] International Organization for Standardization. *Syntactic Metalanguage – Extended Backus-Naur Form*, iso/iec 14977:1996(e) edition, 1996.

[141] Sonny H.S. Chee Jiawei Han, Osmar R. Zaïane and Jenny Y. Chiang. *Electronic Commerce Technologies: Challenges and Opportunities*, chapter Towards On-Line Analytical Mining on the Internet for Electronic Commerce. Prentice Hall, 1999.

[142] Karen Sparck Jones and Peter Willett. *Readings in Information Retrieval.* Morgan Kaufmann Publishers, 1997.

[143] I. Jurisica. Literature review: Information retrieval and hypertext systems, September 1995.

[144] S. Khoshafian and A. B. Baker. *Multimedia and Imaging Databases.* Morgan Kaufmann Publishers, 1996.

[145] M. Kifer, G. Lausen, and J. Wu. Logical foundations for object-oriented and frame-based languages. *Journal of ACM*, 1995.

[146] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proc. 3rd Int. Conf. Information and Knowledge Management*, pages 401–408, Gaithersburg, Maryland, Nov. 1994.

[147] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. 1998 Int. Conf. Very Large Data Bases*, pages 392–403, New York, NY, August 1998.

[148] D.E. Knuth, J.H. Morris, and V.R. Pratt. Fast pattern matching in strings. *SIAM J. Comput*, 6(2):323–350, June 1977.

[149] David Konopnicki and Oded Shmueli. W3qs: A query system for the world-wide web. In *Proc. 21st Int. Conf. on Very Large Data Bases (VLDB)*, pages 54–65, Zurich,Switzerland, 1995.

[150] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *Proc. 4th Int. Symp. Large Spatial Databases (SSD'95)*, pages 47–66, Portland, Maine, Aug. 1995.

[151] Krzysztof Koperski. *A Progressive Refinement Approach for Spatial Data Mining.* PhD thesis, School of Computing Science, Simon Fraser University, April 1999.

[152] H. F. Korth and A. Silberschatz. *Database System Concepts, 2ed.* McGraw-Hill, 1991.

[153] Martijn Koster. Aliweb - archie like indexing the web. In *Proc. 1st Int. Conf. on the World Wild Web*, May 1994. http://www.nexor.com/public/aliweb/.

[154] Martjin Koster. Guidelines for robot writers. Nexor Corp, http://web.nexor.co.uk/mak/doc/robots/guidelines.html.

[155] Martjin Koster. A proposed standard for robot exclusion. Nexor Corp, http://web.nexor.co.uk/mak/doc/robots/norobots.html.

[156] Martjin Koster. The web robots database. Nexor Corp, http://info.webcrawler.com/mak/projects/robots/active.html.

[157] Martjin Koster. World wide web wanderers, spiders and robots. Nexor Corp, http://web.nexor.co.uk/mak/doc/robots/robots.html.

[158] Daniel Kuokka and Larry Harada. Matchmaking for information agents. In *Proc. Of the 14th Joint Conf. On AI*, 1997.

[159] Daniel Kuokka and Larry Harada. Matchmaking for information agents. In *Proc. 14th Int. Joint Conf. on AI*, pages 672–678, 1997.

[160] L. Lakshmanan, F. Sadri, and I. Subramanian. On the logical foundations of schema integration and evolution in heterogeneous database systems. In *Proc. 3rd Int. Conf. on Deductive and Object-Oriented Databases (DOOD'93)*, December 1993.

[161] L. Lakshmanan, F. Sadri, and I. Subramanian. A declarative language for querying and restructuring the web. In *Proc. 6th Int. Workshop on Research Issues in data Engineering*, New Orleans, 1996.

[162] Ora Lassila and Ralph R Swick. Resource description framework (rdf) model and syntax specification. W3C Working Draft, October 1998. http://www.w3c.org/TR/1998/WD-rdf-syntax-19981008/.

[163] Lillian Jane Lee. *Similarity-Based Approaches To Natural Language Processing*. PhD thesis, Harvard University, 1997. also TR-11-97.

[164] B. Lent, A. Swami, and J. Widom. Clustering association rules. In *Proc. 1997 Int. Conf. Data Engineering (ICDE'97)*, pages 220–231, Birmingham, England, April 1997.

[165] Michael Lesk. The seven ages of information retrieval. http://community.bellcore.com/lesk/ages/ages.html.

[166] Alon Y. Levy, Avi Silberschatz, Divesh Srivastava, and Maria Zemankova. Challenges for global information systems. In *Proc. 20th VLDB Conf.*, Santiago, Chile, 1994.

[167] Ze-Nian Li, Osmar R. Zaïane, and Zinovi Tauber. Illumination invariance and object model in content-based image and video retrieval. *Journal of Visual Communication and Image Representation*, 1998. Accepted for publication.

[168] Ze-Nian Li, Osmar R. Zaïane, and Bing Yan. C-bird: Content-based image retrieval from image repositories using chromaticity and recognition kernel. Technical Report TR1998-03, School of Computing Science, Simon Fraser University, February 1998.

[169] Ze-Nian Li, Osmar R. Zaïane, and Bing Yan. C-BIRD: Content-based image retrieval in digital libraries using illumination invariance and recognition kernel. In *Proc. Int. Workshop on Storage and Retrieval Issues in Image and Multimedia Databases, in 9th Int. Conf. on Database and Expert Systems (DEXA'98)*, Vienna, Austria, August 1998.

[170] Z.N. Li and B. Yan. Recognition kernel for content-based search. In *Proc. IEEE Conf. on Systems, Man, and Cybernetics*, pages 472–477, 1996.

[171] Z.N. Li, B.G. Yao, and F. Tong. Linear generalized Hough transform and its parallelization. *Image and Vision Computing*, 11(1):11–24, 1993.

[172] F. Liu and R.W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(7):722–733, 1996.

[173] W. Lu, J. Han, and B. C. Ooi. Knowledge discovery in large spatial databases. In *Far East Workshop on Geographic Information Systems*, pages 275–289, Singapore, June 1993.

[174] M. Antonini, et al. Image coding using wavelet transform. *IEEE Trans. on Image Processing*, 1(2):205–221, 1992.

[175] Udi Manber and Peter A. Bigot. Connecting diverse web search facilities. *Bulletin of the IEEE*, 1998.

[176] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *Proc. AAAI'94 Workshop Knowledge Discovery in Databases (KDD'94)*, pages 181–192, Seattle, WA, July 1994.

[177] M.E. Maron and J.L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of ACM*, 7, 1960.

[178] M. L. Mauldin. *Lycos: Hunting WWW Information.* CMU, 1994. Available from *http://lycos.cs.cmu.edu/.*

[179] Oliver A. McBrian. GENVL and WWWW: Tools for taming the web. In *Proc. 1st WWW Conf.*, May 1994. http://www.cs.colorado.edu/home/mcbryan/mypapers/www94.ps.

[180] Sean McGrath. *XML by Example, Building E-Commerce Applications.* Prentice Hall PTR, 1998.

[181] Kenneth A. Megill. *The Corporate Memory, Information Management in the Electronic Age.* Browker-Saur, 1997.

[182] A. Mendelzon and T. Milo. Formal models of web queries. In *ACM SIGACT-SIGMOD Symp. on Principles of Database Systems*, 1997.

[183] Alberto Mendelzon, George Mihaila, and Tova Milo. Querying the world wide web. In *Proc. PDIS'96*, Miami, December 1996.

[184] R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. In *Proc. 1996 Int. Conf. Very Large Data Bases*, pages 122–133, Bombay, India, Sept. 1996.

[185] Merit data. ftp://ftp.merit.edu/statistics/nsfnet.

[186] R.J. Miller and Y. Yang. Association rules over interval data. In *Proc. 1997 ACM-SIGMOD Int. Conf. Management of Data*, pages 452–461, Tucson, Arizona, May 1997.

[187] Sougata Mukherjea, Kyoji Hirata, and Yoshinori Hara. Towards a multimedia world wide web information retrieval engine. In *Proc. Sixth Int. WWW Conference*, Santa Clara, CA, 1997.

[188] Ted Nelson. A file structure for the complex, the changing and the interminate. In *ACM 20th National Conference*, 1965.

[189] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proc. 1994 Int. Conf. Very Large Data Bases*, pages 144–155, Santiago, Chile, September 1994.

[190] R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data*, pages 13–24, Seattle, Washington, June 1998.

[191] Tam Nguyen and V. Srinivasan. Accessing relational databases form the world wide web. In *Proc. ACM SIGMOD conf. on the Management of Data*, Montreal, Canada, June 1996.

[192] Doug Oard and Jinmook Kim. Information filtering resources. http://www.clis.umd.edu/dlrg/filter/, 1998.

[193] Daniel E. O'Leary, Daniel Kuokka, and Robert Plant. Artificial intelligence and virtual organization. *Communications of the ACM*, 40(1):52–59, 1997.

[194] J. Ostermann, E.S. Jang, J. Shin, and T. Chen. Coding of arbitrarily shaped video objects in MPEG-4. In *Proc. Int. Conf. on Image Processing (ICIP '97)*, pages 496–499, 1997.

[195] B. Özden, A. Biliris, R. Rastogi, and A. Silberschatz. A low-cost storage server for movie on demand databases. In *Proc. 20th Int. Conf. on Very Large Data Bases (VLDB)*, pages 594–605, 1994.

[196] B. Özden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Proc. 1998 Int. Conf. Data Engineering (ICDE'98)*, pages 412–421, Orlando, FL, Feb. 1998.

[197] Roger C. Palmer. *Online Reference and Information Retrieval.* Libraries Unlimited, 2 edition, 1987.

[198] J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. In *Proc. 1995 ACM-SIGMOD Int. Conf. Management of Data*, pages 175–186, San Jose, CA, May 1995.

[199] A. Pentland, R.W. Picard, and S. Sclaroff. Photobook: Tools for content-based manipulation of image databases. In *SPIE Storage and Retrieval for Image and Video Databases II*, volume 2, 185, pages 34–47, San Jose, CA, 1994.

[200] M. Perkowitz and O. Etzioni. Adaptive sites: Automatically learning from user access patterns. In *Proc. 6th Int. World Wide Web Conf.*, Santa Clara, California, April 1997.

[201] Richard Einer Peterson. The biology of the internet: A cladistic taxonomy. http://www2.hawaii.edu/˜rpeterso/biology.htm, 1996.

[202] G. Piatetsky-Shapiro, U. Fayyad, and P. Smith. From data mining to knowledge discovery: An overview. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–35. AAAI/MIT Press, 1996.

[203] G. Piatetsky-Shapiro and W. J. Frawley. *Knowledge Discovery in Databases.* AAAI/MIT Press, 1991.

[204] Jeff Prosise. Crawling the web: A guide to robots, spiders, and other shadowy denizens of the web. *PC Magazine*, 1996.

[205] D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. Querying semistructured heterogeneous information, 1994. ftp://db.stanford.edu/pub/quass/1994/querying-full.ps.

[206] S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. In *Proc. 1998 Int. Conf. Very Large Data Bases*, pages 368–379, New York, NY, August 1998.

[207] A.R. Rao and G.L. Lohse. Towards a texture naming system: identifying relevant dimensions of texture. In *Proc. IEEE Conf. on Visualization*, pages 220–227, San Jose, 1993.

[208] R.L. Read, D.S. Fussell, and A. Silberschatz. A multi-resolution relational data model. In *Proc. 18th Int. Conf. Very Large Data Bases*, pages 139–150, Vancouver, Canada, Aug. 1992.

[209] E. Riloff and L. Hollaar. *Text Databases and Information Retrieval. In Handbook of Computer Science.* A.B. Tucker (ed), CRC Press, 1996.

[210] Edwina L. Rissland and Jody J. Daniels. Using CBR to drive IR. In *Proc. IJCAI 95 Conf.*, pages 400–407, Montreal, Canada, August 1995.

[211] R.M. Haralick, et al. Texture features for image classification. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-3(6):610–621, 1973.

[212] C.S. Roberts. Partial-match retrieval via the method of superimposed codes. *Proc. IEEE*, 67(12):1624–164, December 1979.

[213] S.E. Robertson. The probibility ranking principle in IR. *Journal of Documentation*, 33, 1977.

[214] K. Ross and D. Srivastava. Fast computation of sparse datacubes. In *Proc. 1997 Int. Conf. Very Large Data Bases*, pages 116–125, Athens, Greece, Aug. 1997.

[215] G. Salton. An automatic phrase matching. In D. Hays, editor, *Readings in Automatic Language Processing.* American Elsevier Publishing Company Inc., New York, 1966.

[216] G. Salton. Automatic information retrieval. *IEEE Computer Mag.*, 13(9):41–56, 1980.

[217] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41:288–297, 1990.

[218] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[219] Carsten Schlichting and Erik Nilsen. Signal detection analysis of www search engines. In *Design for the Web: Empirical Studies*, Seattle, WA, USA, October 1996. http://www.microsoft.com/usability/webconf.htm.

[220] Kyle Schurman. How the net has grown. *Guide to going Online*, 5(3):57–61, 1997.

[221] Kyle Schurman. Search engines and web directories. *Guide to going Online*, 5(3):77–79, 1997.

[222] Michael F. Scwartz, Alan Emtage, Brewster Kahle, and B. Clifford Neuman. A comparison of internet resource discovery approach. *Computing Systems*, 5(4), 1992.

[223] E. Selberg and O. Etzioni. Multi-service search and comparison using the metacrawler. In *4th International World Wide Web Conference*, 1996.

[224] Erik Selberg and Oren Etzioni. Multi-engine search and comparison using the metacrawler. In *Fourth International World Wide Web Conference*, pages 195–208, Boston, 1995.

[225] S.F. Chang, et al. VideoQ: an automated content based video search system using visual cues. In *Proc. ACM Multimedia 97*, pages 313–324, 1997.

[226] J. Shakes, M. Langheinrich, and O. Etzioni. Ahoy! the home page finder. In *Proc. Sixth World Wide Web Conf.*, Santa Clara, CA, USA, April 1997.

[227] W. Shen, K. Ong, B. Mitbander, and C. Zaniolo. Metaqueries for data mining. In U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 375–398. AAAI/MIT Press, 1996.

[228] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts, 3ed.* McGraw-Hill, 1997.

[229] Avi Silberschatz, Mike Stonebraker, and Jeff Ullman. Database research: Achivements and opportunities into the 21st century. Report on an NSF Workshop on the Future of Database Systems Research, May 26-27 1995.

[230] John B. Smith and Stephen F. Weiss. An overview of hypertext. *Communications of ACM*, 31(7), July 1988.

[231] J.R. Smith and S.F. Chang. Visually searching the web for content. *IEEE Multimedia*, 4(3):12–20, 1997.

[232] T.R. Smith. A digital library for geographically referenced materials. *IEEE Computer*, 29(5):54–60, 1996.

[233] Ellen Spertus and Lynn Andrea Stein. A hyperlink-based recommender system written in squeal. In *Proc. ACM CIKM'98 Workshop on Web Information and Data Management (WIDM'98)*, pages 1–4, Washington DC, November 1998.

[234] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data*, pages 1–12, Montreal, Canada, June 1996.

[235] S. Stiassy. Mathematical analysis of various superimposed coding methods. *American Documentation*, 11(2):155–169, February 1960.

[236] P. Stolorz, H. Nakamura, E. Mesrobian, R.R. Muntz, E.C. Shek, J.R. Santos, J. Yi, K. Ng, S.Y Chien, C.R. Mechoso, and J.D. Farrara. Fast spatio-temporal data mining of large geophysical datasets. In *Proc. First Int. Conf. On Knowledge Discovery and Data Mining*, pages 300–305, August 1995.

[237] Danny Sulivan. Search engine watch. http://searchenginewatch.com/.

[238] Danny Sullivan and Richard Karpinski. Supercharge your web searches. *NetGuide Magazine*, pages 63–84, May 1997.

[239] T. Sullivan. Reading reader reaction : A proposal for inferential analysis of web server log files. In *Proc. 3rd Conf. Human Factors & the Web*, Denver, Colorado, June 1997.

[240] D.M. Sunday. A very fast substring search algorithm. *Communications of ACM*, 33(8):132–142, August 1990.

[241] M.J. Swain and D.H. Ballard. Color indexing. *Int. J. Computer Vision*, 7(1):11–32, 1991.

[242] H. Tamura, S. Mori, and T. Yamawaki. Texture features corresponding to visual perception. *IEEE Trans. on Systems, Man, and Cybernetics*, 8(6):460–473, 1978.

[243] Y. Taniguchi, A. Akutsu, and Y. Tonomura. PanoramaExcerpts: extracting and packing panoramas for video browsing. In *Proc. ACM Multimedia 97*, pages 427–436, 1997.

[244] Gary Taubes. Indexing the internet. http://www.edoc.com/aaas/computers/webindex.html.

[245] L. Tauscher and S. Greenberg. How people revisit web pages: Empirical findings and implications for the design of history systems. *International Journal of Human Computer Studies, Special issue on World Wide Web Usability*, 47:97–138, 1997.

[246] A.M. Tekalp. *Digital video processing.* Prentice Hall PTR, 1995.

[247] L. Teodosio and W. Bender. Salient video stills: Content and context preserved. In *Proc. First ACM Int. Conf. on Multimedia*, pages 39–46, 1993.

[248] V. Tucakov and R. Ng. Identifying unusual spatio-temporal trajectories from surveillance videos. In *Proc. of 1998 SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98)*, Seattle, Washington, June 1998.

[249] Elizabeth Tudhope. Query based stremming. Technical Report CS-96-31, University of Waterloo, 1996.

[250] Marc D. VanHeyningen. The unified computer science technical report index: Lessons in indexing diverse resources. In *Proc. Second Int. WWW Conf.*, Chicago, October 1994. http://www.cs.indiana.edu/ucstri/paper/paper.html.

[251] C. Varela, D. Nekhayev, P. Chandrasekharan, C. Krishnan, V.Govindan, D. Modgil, S.Siddiqui, and M. Winslett D. Lebedenko. Browsing object-oriented databases over the web. In *Fourth International World Wide Web Conference*, pages 209–220, Boston, 1995.

[252] Ronald J. Vetter, Chris Spell, and Charles Ward. Mosaic and the world-wide web. *IEEE Computer*, 27(10):49–57, October 1994.

[253] Peter J. Vigil. *Online Retrieval Analysis and Strategy.* John Willey and Sons, 1988.

[254] Jian Wang. Motion-based object segmentation from digital video. Master's thesis, School of Computing Science, Simon Fraser University, 1998.

[255] Lynn Ward. Exploring the power of the internet gopher. *UIUCnet, Dec. 1992 - Jan. 1993*, 6(1), 1993. ftp://ftp.cso.uiuc.edu/doc/net/uiucnet/vol6no1.txt.

[256] J. Wei, M.S. Drew, and Z.N. Li. Illumination invariant video segmentation by hierarchical robust thresholding. In *Proc. IS&T/SPIE Symp. on Electronic Imaging '98, Storage & Retrieval for Image and Video Databases VI, SPIE Vol. 3312*, pages 188–201, 1998.

[257] Jie Wei. *Foveate Wavelet Transform and its Applications in Digital Video Processing, Acquisition, and Indexing.* PhD thesis, School of Computing Science, Simon Fraser University, November 1998.

[258] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin core metadata for resource discovery. Request for Comments rfc2413, September 1998. http://info.internet.isi.edu/in-notes/rfc/files/rfc2413.txt.

[259] Stuart Weibel, Jean Godly, Eric Miller, and Ron Daniel. OCLC/NCSA metadata workshop report (the essential element of network object description), March 1995. http://www.oclc.org:5046/conferences/metadata/dublin_core_report.html.

[260] Daniel Weld and Oren Etzioni. The first law of robotics (a call to arms). In *Proc. AAAI 94 Conf.*, 1994.

[261] William F. Williams. *Principles of Automated Information Retrieval.* The Business Press, 1965.

[262] W.A. Woods. Important issues in knowledge representation. *Proc. of the IEEE*, 74(10), October 1986.

[263] WordNet - a lexical database for english. http://www.cogsci.princeton.edu/~wn/, 1998.

[264] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulas, 2nd ed.* Wiley, New York, 1982.

[265] Bing Yan. Content based search in multimedia databases. Master's thesis, School of Computing Science, Simon Fraser University, 1997.

[266] Tak W. Yan and Hector Garcia-Molina. The electronic library of the future: Accessing worldwide information.

[267] T.W. Yan and H. Garcia-Molina. SIFT - a tool for wide-area information dissemination. In *Proc. of the 1995 USENIX Technical Conference*, pages 177–186, 1995.

[268] Budi Yuwono, Savio L.Y. Lam, Jerry H. Ying, and Dik L. Lee. A world wide web resource discovery system. In *Fourth International World Wide Web Conference*, pages 145–158, Boston, 1995.

[269] C.T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. on Computers*, 20(1):68–86, January 1971.

[270] Osmar R. Zaïane. From resource discovery to knowledge discovery on the internet. Technical Report TR1998-13, School of Computing Science, Simon Fraser University, August 1998.

[271] Osmar R. Zaïane. From resource discovery to knowledge discovery on the internet. *ACM Computing Surveys*, 1998. submitted for publication.

[272] Osmar R. Zaïane, Andrew Fall, Stephen Rochefort, Veronica Dahl, and Paul Tarau. Concept-based retrieval using controlled natural language. In *Proc. Workshop on Applications of Natural Language to Information Systems (NLDB97)*, Vancouver, Canada, June 1997.

[273] Osmar R. Zaïane, Andrew Fall, Stephen Rochefort, Veronica Dahl, and Paul Tarau. On-line resource discovery using natural language. In *Proc., RIAO'97:Computer-Assisted Searching on the Internet*, Montreal, Canada, june 1997.

[274] Osmar R. Zaïane, Eli Hagen, and Jiawei Han. Data cleaning and hierarchy building for data mining and information retrieval from the internet. publication in progress, 1999.

[275] Osmar R. Zaïane, Eli Hagen, and Jiawei Han. Word taxonomy for on-line visual asset management and mining. In *Fourth International Workshop on Application of Natural Language to Information Systems (NLDB99)*, Klagenfurt, Austria, June 1999.

[276] Osmar R. Zaïane and Jiawei Han. Resource and knowledge discovery in global information systems: A preliminary design and experiment. In *Proc. First Int. Conf. On Knowledge Discovery and Data Mining*, Montreal, Canada, 1995.

[277] Osmar R. Zaïane and Jiawei Han. Webml: Querying the world-wide web for resources and knowledge. In *Proc. ACM CIKM'98 Workshop on Web Information and Data Management (WIDM'98)*, pages 9–12, Washington DC, November 1998.

[278] Osmar R. Zaïane and Jiawei Han. Mediating virtual web views. In *Fourth IFCIS Conference on Cooperative Information Systems (CoopIS'99)*, Edinburgh, Scotland, September 1999. submitted for review.

[279] Osmar R. Zaïane, Jiawei Han, Ze-Nian Li, Jenny Y. Chiang, and Sonny Chee. Multimedia-miner: A system prototype for multimedia data mining. In *Proc. 1998 ACM-SIGMOD Conf. on Management of Data*, pages 581–583, Seattle, Washington, June 1998.

[280] Osmar R. Zaïane, Jiawei Han, Ze-Nian Li, and Jean Hou. Mining multimedia data. In *CASCON'98: Meeting of Minds*, Toronto, Canada, November 1998.

[281] Osmar R. Zaïane, Jiawei Han, and Hua Zhu. Association rules with recurrent items for multimedia artifacts. In *Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'99)*, San Diego, CA, August 1999. submitted for review.

[282] Osmar R. Zaïane, Man Xin, and Jiawei Han. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In *Proc. Advances in Digital Libraries ADL'98*, pages 19–29, Santa Barbara, CA, USA, April 1998.

[283] Robert H'obbes' Zakon. Hobbes' internet timeline. available http://www-personal.umd.umich.edu/~nhughes/htmldocs/timeline.html.

[284] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Proc. ACM SIGIR'98*, 1998.

[285] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. Fast and intuitive clustering of web documents.

[286] Hua Zhu. On-line analytical mining of association rules. Master's thesis, School of Computing Science, Simon Fraser University, December 1998.