2001

Advanced Web search based on formal concept analysis.

Bong-Seop. Kim
University of Windsor

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

ADVANCED WEB SEARCH BASED ON FORMAL CONCEPT ANALYSIS

By

Kim, Bong-Seop

A thesis

Submitted to the Faculty of Graduate Studies and Research

through School of Computer Science

in Partial Fulfillment of the Requirements for

the Degree of Master of Science at the

University of Windsor

Windsor, Ontario, Canada

2001

ABSTRACT

Explosive increase of global wide web using has led to "web searches" as one of the most important tasks in our daily lives. However, most web users discover that only a few results are valuable among thousands of results returned. The purpose of this thesis is to derive a new and improved way of web searching. To that end, research aimed at solving the above mentioned problems and surveying the methodologies or approaches adapted to them was found first. Some research papers give hints that help analyze user requirements for web searches. This survey on background research papers and user requirement analyses is one basis of my thesis. Another research area related to Formal Concept Analysis that is the theoretical background of my thesis is also surveyed. Following the research papers surveyed, a new searching methodology, *the advanced web searching methodology based on formal concept analysis*, is proposed. A lattice derived from the Formal Concept Analysis gives dynamic and interactive aspects to the new niche search engine.

KEY WORDS: Formal Concept Analysis, Lattice, Web, Search, Dynamic, Interactive, Advanced

DEDICATION

I want to thank my wife, Eun-Young, for her devoted patience, support, and guidance throughout my educational journey in a foreign land. Her understanding with great love always gave me the courage to move far beyond my ability.

Many thanks to my daughter Ji-Yoon(Holly) who thought her father(me!) was the best in the world and to my son Do-Heon(Kenneth) who was born here and brought up well.

나의 사랑하는
부인 은영,
딸 지윤,
아들 도헌에게 바칩니다.

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Dr. Park, who is currently at Bradley University in America, for his invaluable help throughout the last 2 years. He accepted me as his student gladly and inspired me in every aspect.

A special thanks goes to Dr. Suh and Dr. Li for their kind advice throughout my thesis work.

Many thanks are Dr. Tsin's allotment for his scrupulous job on my background reading.

There have been three special people with whom I had good friendship over the last two years. Jong-Seok Kim, Taek-Sueng Jang, and Han-Soo Ahn are they. One of the things that I will miss the most will be the smoking time at the Library.

I want to thank my parents for their endless love.

The final thanks goes to the Korean Government which sponsored me for the last 2 years.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Current situation of web searching

Explosive increase of web using.

As more of the population goes on-line, and as more tasks are performed on the web, the need for better search services is becoming increasingly important [43]. The internet has grown rapidly since its inception in Dec. 1969 and is anticipated to expand 1,000% over the next few years [4]. The WWW represents significant advancements for the retrieval and dissemination of scientific and other literature and for the advancement of education [4,75]. A recent study by Lawrence and Giles estimated the size of the web at about 8 billion indexable pages [42]. In addition, the amount of scientific information and the number of electronic journals on the internet continue to increase[1].

Importance of web searching

A GVU[2] study showed that about 85% of web users launch search engines to locate information [35], and users rate searching as the most important activity conducted on the internet [37]. Immediate access to

---

[1] More than 1,000 journals as of 1996 were reported to be opened on the web [4,68].
[2] Graphic, Visualization, and Usability Center at Georgia Tech

all scientific literature has long been a dream of scientists, and web search engines have made a large and growing body of scientific literature and other information resources accessible within seconds. Scientific information retrieval, previously dominated by librarians, is now directly available to a widespread group of scientists [62].

Difference of web searching with traditional IR(Information Retrieval)

The networked and dial in connectivity available to web searchers creates a near ubiquitous online searching environment. Now, the web appears to be a whole new searching environment [65]. The web is distributed, dynamic, and rapidly growing. As a result, information resources present difficulties for traditional information retrieval technologies [34]. Traditional information retrieval systems were designed for different environments and have typically been used for indexing a static collection of directly accessible documents [61]. But the WWW is not similar to a database. For example, there is no uniform structure, no integrity constraints, no transactions, no standard query language or data model [22]. Search engines can return documents that do not contain the query terms. This behavior can occur because (i) the information retrieval technology used by the engine may not require an exact match (for example, Excite uses 'concept-based clustering', and Infoseek uses morphology; these engines can return documents with related words), (ii) documents may no longer exist (an engine that never

deletes invalid documents would be at an advantage), (iii) documents may still exist but may have changed and no longer contain the query terms [47].

Increased necessity of new approach to web searching

Querying the web has understandably gathered much attention from both researchers and industry [30]. With the advent of the WWW, a new category of searching now presents itself [36]. With the rapid expansion of information bases and user communities in the internet, efficient and effective discovery and use of the resources in the global information network have become an important issues in research into global information systems [74]. The popularity of the WWW has made it a prime vehicle for disseminating information. The relation and querying of this information has led to a significant body of recent research addressing these problems [22].

Problems of web searching

When searching the web, a user can be overwhelmed by thousands of results retrieved by a search engine, few of which are valuable [28]. And among the retrieved results, the percentages of invalid links were, from best to worst, 1.6% for Lycos, 2.0% for Excite, 2.5% for Altavista, 2.6% for Infoseek, 5.0% for Northen Light, and 5.3% for Hotbot [47].

Also, languages for querying www data are too complex for casual web users. Furthermore, proposed query approaches do not take advantage of the interactivity of typical web sessions - users are proficient at iteratively refining their web explorations. Search engines assume little about the semantics of a document, which works well for the conglomeration of disparate data sources that make up the web. But for searching within a single web site, a search engine may be too blunt a tool [30].

Nowadays, large web sites with thousands of pages are attracting millions of users. For example, the ESPN Sports site (espn.com) has over 90,000 different pages [66] and several million page views a day [67]. As large as some sites may be, they are fundamentally different from the web as a whole since a single site usually has a controlled point of administration. Thus, it becomes possible to consistently assign and expose the site's semantic data relationship and thereby enable more expressive searches [30].

1.2 Purpose of this thesis.

The purpose of this thesis is to derive a new and improved way of web searching. For that, research aimed at solving the above mentioned

problems and surveying the methodologies or approaches adapted to them is found first. Some research papers give hints that help analyze user requirements for web searches. This survey on background research papers and user requirement analyses will be a basis of my thesis. Another research area related with Formal Concept Analysis that is the theoretical background of my thesis is also surveyed. Following the research papers surveyed, a new searching methodology, *the advanced web searching methodology based on formal concept anlaysis*, is proposed.

1.3 Brief description of remainder chapters

The remainder of this paper is organized as follows:

Chapter 2 surveys the web searching methodologies such as explicit customization of searching (section 2.1), implicit customization of searching (section 2.2), specialized search engines for specific domains (section 2.3), interactive query and search (section 2.4), combining the results of multiple search engines (section 2.5), and data representation for web databases (section 2.7).

Chapter 3 gives user requirement analysis for web searching.

Design phase is described in the *chapter 4*. Formal Concept Analysis (FCA), a theoretical base of this thesis paper, is suggested and the overview of it is given in the section 4.1 followed by the definition of a new searching methodology in section 4.2. A simple example showing the operation of FCA is given in section 4.3.

Chapter 5, implementation details, follows the design phase: tools description (section 5.1); information gathering (section 5.2); lattice construction (section 5.3); user interface(section 5.4).

Finally some words for 'evaluation and future work' are given in the *chapter 6*.

References, source codes (Appendix A~C) and tools description (Appendix D~G) are attached.

2. SURVEY OF WEB SEARCHING METHODOLOGIES

2.1 Explicit customization of searching

"Explicit" is a term used to refer to certain customizing actions for searching made by users. This approach aims at improving the quality of result from web search engines. The preferences define a search strategy that specifies source selection, query modification, and result scoring [26].

A project named Inquirus 2 at NEC Research Institute requests context information (user preference) that is in the form of a category of information desired [26,28]. This project is based on the assumption that it is a user's information need that determines which documents are valuable [28].

Therefore, different results may come out of the same query depending on the different user preferences. It is said that Inquirus2 has proven to be highly effective at improving the precision of search results within given categories [43].

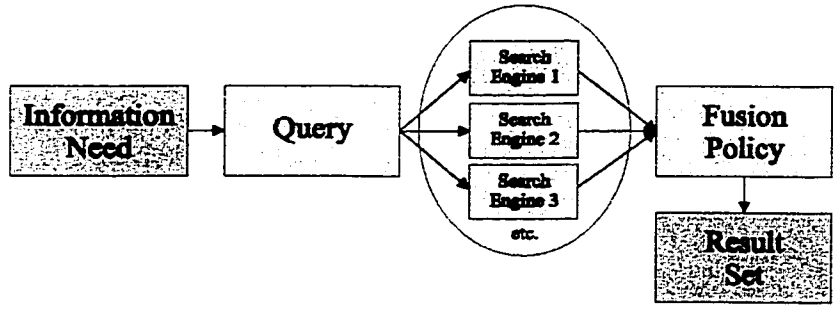


Fig. 1 The architecture of a standard meta search engine [28]

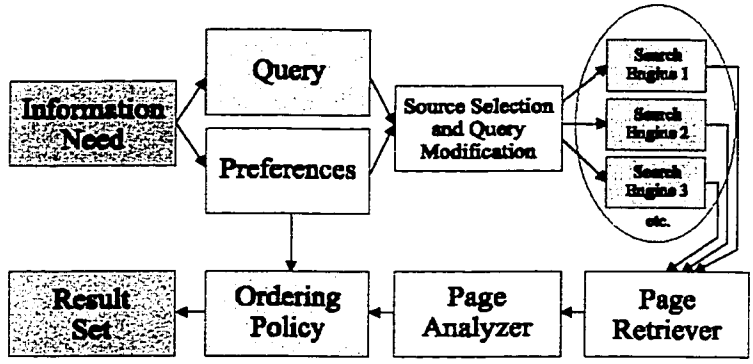


Fig. 2 The architecture of a Inquirus2 search engine [28]

Recent research related to Inquirus2 includes learning methods that automatically learn query modification [14,27].

Problem

The problem of this approach is that a user's explicit action of defining context information is required.

2.2 Implicit customization of searching

In this approach, no explicit actions from users such as requesting any document or setting any preference are made. All the users' preferences are inferred by the system. Information related to current action of a user is brought as the user opens some documents, this approach is also called a 'just-in-time information system' [8]. Designers of such systems typically make the assumption that the goal of the system should be to retrieve objects that are similar to the one currently being manipulated like word processors, WWW browsers, e-mail systems [8]. Rhodes [58] said "long-term use of such system provides a deeper understanding of the users' context."

2.2.1 Server model

The search engines Excite(http://www.excite.com), Lycos (http://www.lycos.com), Google (http://www.google.com), and Yahoo (http://www.yahoo.com) provide special functionality for certain kinds of queries.

Google keeps track of a user's previous queries and selected documents, and uses this information to infer user interests. For example, a user who searches for material related to computer science may have the homepage of a computer scientist ranked highly. Queries to Excite and

Lycos that match the name of an artist or a company produce additional results that link directly to a artist or company information. In Yahoo, a query for a sports team name leads a user to links of other teams and league information [43].

Problem: The cost is too expensive for the major web search engines

2.2.2 Client model

Watson Project

Watson is based on the content of documents being edited in Microsoft Word, or viewed in Internet Explorer. The documents that users are editing or browsing are analyzed with an algorithm that aims to identify words indicating a content of the documents. Then the words are weighted heuristically by the algorithm. If a user enters an explicit query, Watson modifies the query based on the content of the documents being edited or viewed, and forwards the modified query to web search engines; thus it automatically adds context information to the web search [7,8,9].

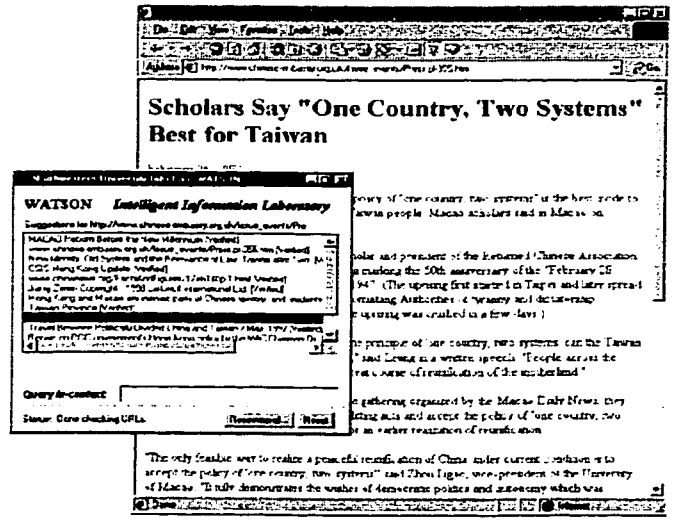


Fig. 3 Watson is recommending related documents to a user.

Remembrance Agent(RA)

Remembrance Agent can be thought of as a UNIX version of the Watson project (actually, Remembrance Agent precedes Watson in time). Remembrance Agent references an Emacs editor instead of Microsoft Word in the Watson project. It keeps eye on specific files of a user like email messages or research papers viewed in Emacs and keeps searching for related documents while the user edits the document in the Emacs editor in a UNIX environment [58,60]. With RA, users need not have a query in mind, or even know that information relevant to their situation exists since RA gives information without any prompting. So, users do not know that useful information exists in their given context.

Margin Notes

When a user refers to a web page, Margin Notes loads a new HTML document having a black margin to the right side of the document which has hyperlinks to personal files. It then compares each section of the document to pre-indexed email achieves, notes files, and other text files, based on co-occurring keywords. If one of these files is found to be relevant to the current section of the web page, a small 'suggestion box' is included in the margin next to the relevant section. The box contains a quick description of the suggested text, an bar representing the relevance of the suggestion, and a link to get more information [59].

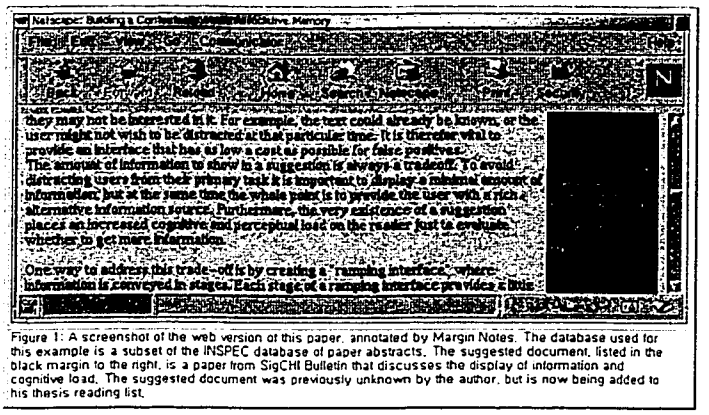


Figure 1: A screenshot of the web version of this paper, annotated by Margin Notes. The database used for this example is a subset of the INSPEC database of paper abstracts. The suggested document, listed in the black margin to the right, is a paper from SigCHI Bulletin that discusses the display of information and cognitive load. The suggested document was previously unknown by the author, but is now being added to his thesis reading list.

Fig. 4 Sample screen using Margin Notes

Autonomy's Kenjin Program

This program also automatically suggests content from the web or local files, based on the documents a user is reading or editing [http://www.kenjin.com].

Fab

This approach offers two kinds of services. One is a content based service which recommends items based on local information and the other is a collaborative service which recommends items based on the recommendations of other users [3].

Letizia

Letizia is an agent which helps a user browse the web. This agent is interested in the behavior of users who browse web sites by following links and searching the web as a background process.

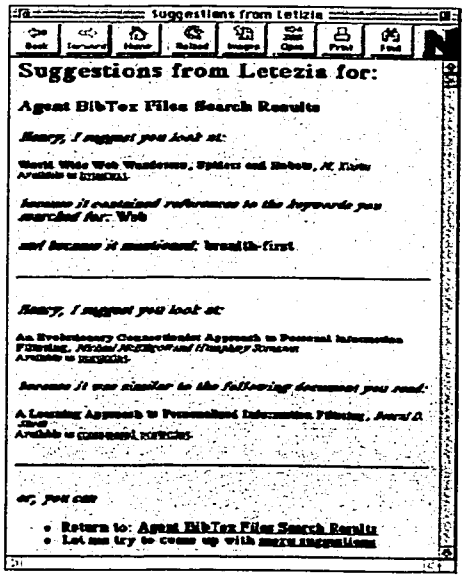


Fig. 5 Sample screen offered from Letzia

It tries to anticipate what items may be of interest to the user. A request from the user drives Letizia to display a page containing its current

recommendations as shown in Fig. 5 [49].

WebWatcher [2], Syskill and Wbert [55] are also other similar tries.

Problem

Watson and Kenjin extract context information only from the current document that a user is editing or viewing [43] and there is no way to gather appropriate user context information if the user views a web page that has no relation with the user's current interest. Remembrance Agent actually cannot learn which of those suggested documents are actually useful [60]. Automatic and even unrequested information may confuse users' web activities [59].

2.3 Specialized search engines for specific domains

"Specialized" is a term to indicate that for some specific topics specific search engines that are only interested in the topics have much better performance than general purpose search engines. Currently, thousands of specialized search engines already exist (http://www.completeplanet.com, http://www. invisibleweb.com).

The following sites are examples of specialized sites favored by

scientists in the world:

ReaserchIndex [http://www.researchindex.org]
Researchindex automates the creation of citation indices for scientific literature, provides easy access to the context of citations to papers and has specialized functionality for extracting information commonly found in research articles [44,45,46].

DEADLINER(currently under testing)
Parses conference and workshop information from the web, newsgroups and mailing lists [41].

FlipDog [http://www.flipdog.com]
Parses job information from employee sites.

HPSearch [http://hpsearch.uni-tier.de/hp/]
Indexes the homepages of computer scientists.

The main problem of specialized searches can be viewed two. The first one is the problem of a specialized search engine itself. That is to say, the specialized search engine must have some efficient algorithm to identify specific sites (communities) having their own interests. The second one is the problem of users. Namely, users don't know which

sites have specialized search engines of interest to them or what are the addresses (URLs) of the sites.

For these reasons, two closely related approaches are being researched as follows:

2.3.1 Algorithms for community identification

A web community is defined as "a collection of pages where each member has more links inside the community than outside of the community" [43]. Several algorithms for identification of communities of related pages on the web have been proposed.

a. Algorithm for topic distillation in a hyperlinked environment [5]
b. Automatic resource compilation by analyzing hyperlink structure and associated text [11]
c. Focused crawling: an approach to topic-specific web resource discovery [12]
d. Crawling through URL ordering [13]
e. Focused crawling using context graphs [17]
f. Efficient identification of web communities [21]
g. Inferring web communities from link topology [25]
h. Authoritive sources in a hyperlinked environment [39]
i. Using reinforcement learning to spider the web [57]

2.3.2 Methods for locating specialized search engines.

It may be the best solution for web users if the general purpose web search engine like Google can locate the specialized search engine for the users. However, an important issue for the major search engines to maintain their services is to keep as many users as possible. This situation has led to many researches to this area.

a. Server selection on the world wide web [16]

b. Experiences with selecting search engines using meta-search [18]

c. Profusion: Intelligent fusion from multiple distributed engines [24]

d. STARTS: Stanford proposal for internet meta-searching [33]

e. GIOSS: Text-Source discovery over the internet [32]

f. Generalizing GIOSS to vector-space database and broker [31]

g. Selecting task-relevant sources for just-in-time retrieval [48]

h. Estimating the usefulness of search engines [50]

i. Effective retrieval with distributed collections [72]

2.4 Interactive query and search based on semistructured databases

In this approach, users can begin with a simple keyword search, and then

dynamically browse the structure of the result, and finally submit further refining queries. Recently, numerous researchers have proposed semistructured data models, databases, and languages for modeling, storing, and querying WWW data. Such proposals argue that graph-based semistructured or unstructured database, without the requirement of an explicit schema, is better suited than traditional database systems for storing the varied, dynamic data of the web.

DataGuides, an interactive query tool, supports maintaining dynamic schemas that are generated from the database. Goldman and Widom [29] said "DataGuides is useful for browsing database structure, formulating queries, storing information such as statistics and sample values, and enabling query optimization".

Problem

While dynamic results given by a interactive query tool like DataGuides may be an important technology in the future, presenting the results as a summary of paths from the root may also confuse the users.

Many languages from the coresponding research papers are proposed for this kind of approach.

Lorel

Lorel is said to be a user-friendly language in the SQL/OQL style for querying databases and can be viewed as an extension of ODMG[3] and the language as an extension of OQL[4] [1].

UnQL

UnQL is a query language suitable for unstructured data models in which each component of the database carries its own description and is independent of other components. Buneman [10] proposed a simple language UnQL for querying data organized as a rooted, edge-labeled graph.

StruQL

StruQL is a language for the management of semistructured data, as well as a view definition for such data [20].

GraphLog

GraphLog is a visual query language designed for general purpose database applications [15].

Problem

---

[3] Object Database Management Group
[4] Object Query Language

So far, for the semistructured or unstructured data there has been little discussion of who will query such data and what typical queries will look like. Given the domain, usually a large and important group of clients is simply casual web users. It's nonsense to expect a typical web user to type such complicated queries. A research paper on queries of semistructured data by Goldman [30] also indicates above problem as, "It is possible to handle certain queries by having users fill in hard-coded forms, but this approach by nature limits query flexibility."

2.5 Combining the results of multiple search engines

This approach aims at improving the coverage of results from web search engines. The primary advantages of a metasearch engine over a single search engine are increased coverage and a consistent interface [63].

Another alternative for the meta search engines like MetaCrawler [47] is the 'softbot'. The softbot transforms queries into goals and uses a planning algorithm to generate a sequence of actions in order to satisfy the goal [19]. One successful implementation of softbot is "AHOY!" Service, which locates home pages for individuals.

2.6 Data Representation for web database

Graph data models

In this model, nodes represent web pages (or internal components of web pages), and arcs (edges) represent links between pages. The labels on the edges can be viewed as attribute names. Along with a labeled graph model, several query languages have been developed. One central feature that is common to these query languages is the ability to formulate regular path expression queries over the graph. Regular path expressions enable posing navigational queries over the graph structure [22].

Semistructured data models

Semi structured data refers to data with some of the following characteristics [22]:

a. The schema is not given in advance and may be implicit in the data.
b. The schema is relatively large (with respect to the size of data) and may be changed frequently.
c. The schema is descriptive rather than prescriptive, i.e., it describes the current state of the data, but violation of the schema is still tolerated.
d. The data is strongly typed, i.e., for different objects, the value of the same attribute may be of differing types.

In many cases, the structure of the data is irregular. Specifically, when modeling the structure of a web site, we don't have a fixed schema that is given in advance. When modeling data coming from multiple sources, the representation of some attributes (e.g., addresses) may differ from source to source [22]. Hence, several projects have considered models of semistructured data. The initial motivation for this work was the existence and relative success of permissive data models such as in the scientific community [69], the need for exchanging objects across heterogeneous source [52], and the task of managing document collections [51].

OEM (Object Exchanging model)

This is one of the proposed data models for semistructured data. In this data model, each object is described by a triple consisting of a label, a type, and the value of the object [56].

Some examples of research papers dedicated to other data models for web searching are Haystack [38] and Unstructured data model [10].

3. USER REQUIREMENTS ANALYSIS: INCREASING NEED OF DYNAMIC AND INTERACTIVE WEB SEARCHING

Most of the internet users utilize search engines as their important tool.

A GVU[5] study showed that about 85% internet users launch search engines to locate information [35][6]. And users rate searching as the most important activity conducted on the internet [37].

General web search engines are not good for tailored result

But, the major web search engines have significant limitations : they are often out-of-date; they only index a fraction of the publicly indexable web; they do not index documents that require authentication; and they do not index sites equally [6,42].

Users looking for documents within specific categories may have a difficult time locating valuable documents using general purpose search engines [26].

Web search engines generally treat search requests *in isolation*. The

---

[5] Graphic, Visualization, and Usability center at Georgia Tech.

[6] Another research shows that 71% of web users accesses search [33].

results for a given query are identical and independent of the user or the context in which the user made the request [43]. Web users are compared and contrasted with users of traditional information retrieval and online public access systems to discover if there is a need for more studies that focus predominantly or exclusively on web searching. The comparison in Jansen and Pooch [36] indicate that "web searching differs from searching in other environments."

Users want simple querying mechanism

The most common query length was *two terms*. In query syntax, the majority of queries contained *no boolean operators*. Kirsch, the chairman and founder of Infoseek Corporation, reported that the average query on Infoseek was 2.2 terms; although about 10% of the queries contained boolean operators, only 1% of the queries utilized advanced searching techniques, and the majority of queries were noun phrases [40].

User's need for dynamic and improved response from search engine is increasing.

The Excite report [73] says that "The average query length in the U.S. has increased from 1.5 terms in 1996 to 2.6 terms in 1999 and the use of boolean operators has increased from 22% in 1996 to 29% in 1999." It also states that "web searching is precision based with over 70% of

Excite searchers looking at only the top ten results or the first page of results. Over 29% of Excite searchers utilized suggestions from the online thesaurus."

The requirements described above motivate a new search engine that is different from current search systems surveyed in the previous chapter. The new search engine is needed to give users simple but meaningful query mechanism with a preferably dynamic aspect.

A simple, dynamic, and meaningful query mechanism can be obtained based on a lattice and the lattice can be constructed by Formal Concept Analysis.

4. Design : Introduction of Formal Concept Analysis

4.1 Formal Concept Analysis overview

Formal Concept Analysis (FCA) is a data analysis technique based on an ordered lattice theory. It can provide graph-based visualizations of tabular data and has successfully been applied to a number of fields including Text Data Mining, Psychology, Social Science and Software Engineering [70], especially in identifying modules [64], retrieving functions or software [53,54].

Formal Concept Analysis (FCA)[7], a relatively new field of applied mathematics, emerged as a branch of applied lattice theory. FCA provides a way to identify sensible groupings of objects that have common attributes [71] and gives a technique to create the concept lattice. The central idea of formal concept analysis is the understanding that a fundamental unit of thought is a concept and a context. A concept consists of two parts: *the extent* which contains all attributes common to all the objects, and *the intent* which contains all attributes common to all the objects [23].

---

[7] FCA was introduced by Wille.

A *formal context* is a triple(O,A,R) where O is a finite set of elements called objects of the context, A is a finite set of elements called attributes of the context, and R is a binary relation on $O \times A$ ($(o,a) \in R$ means that the *object o has the attribute a* where $o \in O$ and $a \in A$).

A *formal concept* of the context (O,A,R) is a pair (E,I) where $E \subseteq O$ and $I \subseteq A$. I is called *intent* and is the set of attributes common to the objects in E, and E is called *extent* and is the set of objects that have all attributes in I.

An Ordering ($\leq$) among concepts is defined as follows :

$$\text{Concept}_1: (E_1, I_1) \leq \text{Concept}_2: (E_2, I_2)$$

$$iff \quad E_1 \subseteq E_2$$

$$\text{or } iff \quad I_2 \subseteq I_1$$

It is read 'concept 1 is sub-concept of concept 2'

4.2 Definition of new searching methodology : Advanced web search based on Formal Concept Analysis[8]

[8] This is a new approach never been tried by others.

"Advanced" is a term used to refer 'interactive' and 'dynamic'. That is to say, our approach will give users a way to find their final searching category interactively. The user is given an available set of keywords from the system. The keyword set will be dynamically changed according to the user's position in a given lattice. The user will find his final category by narrowing or broadening a scope using the dynamically given keyword set.

So far, FCA has been applied only to systems that require relatively small amounts of data to be processed, an it has not been introduced into the field of web searching in which huge amounts of data must be processed. Moreover, the FCA is barely applicable to the system since it costs too much to construct the lattice. But, our approach has no interest in such a general searching system. The point is that the FCA may have a great role in web document searching field like those given above. Our approach will concentrate on relatively small sets of data, for example, specialized categories like 'patent', 'specialized search engine', 'directories from major search engines' or 'a specific domain such as ESPN or CNN' and so on.

Example I : Category Locator

In this scenario, category means the directories from major search engines which offers 'category search' such as Yahoo, Altavista,

Lycos, Infoseek, Hotbot and so on.

Every directory (category) is analyzed based on the formal concept. And then, the information like category name, link for the category, and keywords related with the category is stored in the form of concept lattice. Finally, the information in the lattice will be retrieved in response to users' requests.

With the aid of lattice, users are dynamically and interactively offered available categories as they zoom in or zoom out the scope of their search. If a user select a link from suggested result set, the user will be lead to a category that hold a lot of related sites with the user's request.

Ideally or hopefully, all the categories from the dominant web search engines will be suggested simultaneously. Therefore users don't need to choose many search engines separately.

Example II : For the specific topic like 'patent' in example I

If the previous scenario is applied to a specific topic such as 'patent'[9], the resulted system will be a specialized search engine for a specific topic. This case is implemented in Chapter 5[10].

---

[9] It can be thought as a subset of all category locator.
[10] The corresponding codes are APPENDIX A-II, APPENDIX B, and APPENDIX C.

4.3 Simulation of FCA over the example I in section 4.2

Consider 3 search sites that have following directory structures

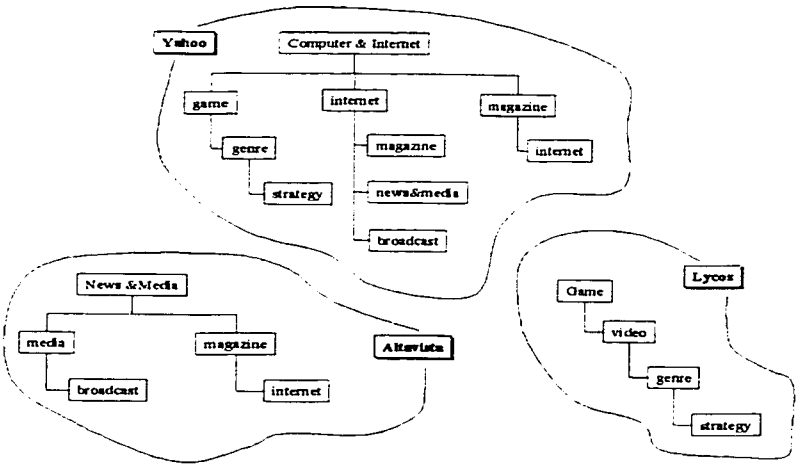


Fig. 6 Samples of three search engines that offers category searching

Using FCA, categories (O), keywords (A), and the relation(R) between them may be represented as in the table.

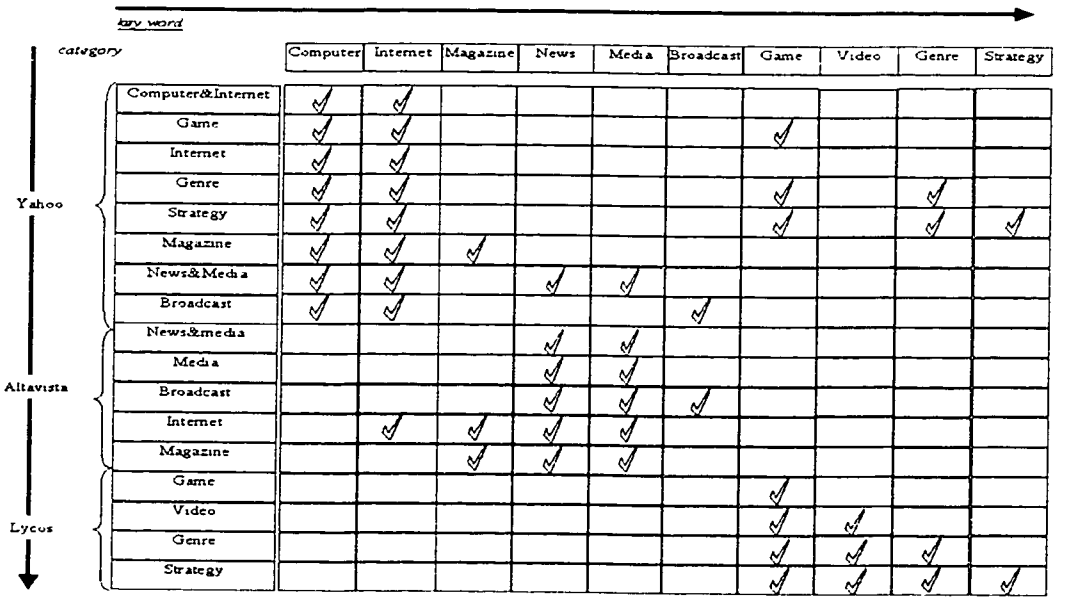


Fig. 7 Relation between Object and Attribute in the form of a table

The previous table may be converted into the following 23 concepts in

the same way as with the following process extracting a concept "n1."

- At first the intent of an object "concept&Internet" is calculated: {computer,internet}
- And then the extent of the intent above is calculated from the relation table(**Fig. 7**): {Computer&Internet, Game, Internet, Genre, Strategy, Magazine, News&Media,Broadcast }

Top=(all categories,{})

n1=({Computer&Internet(Y), Game(Y), Internet(Y), Genre(Y), Strategy(Y),Magazine(Y), News&Media(Y), Broadcast(Y)},{computer, internet})

n2=({Game(Y),Genre(Y),Strategy(Y)},{computer,internet,game})

n3=({Genre(Y),Strategy(Y)},{computer,internet,game,genre})

n4=({Strategy(Y)},{computer,internet,game,genre,strategy})

n5=({Magazine(Y)},{computer,internet,magazine})

n6=({News&Media(Y)},{computer,internet,news,media})

n7=({Broadcast(Y)},{computer,internet,broadcast})

n8=({News&Media(A),Media(A),Broadcast(A),Internet(A),Magazine(A)},{news,media})

n9=({Broadcast(A)},{news,media,broadcast})

n10=({Internet(A)},{internet,magazine,news,media})

n11=({Internet(A),Magazine(A)},{magazine,news,media})

n12=({Game(L),Video(L),Genre(L),Strategy(L)},{game})

n13=({Video(L),Genre(L),Strategy(L)},{game,video})

n14=({Genre(L),Strategy(L)},{game,video,genre})

n15=({Strategy(L)},{game,video,genre,strategy})

n16=({Computer&Internet(Y), Game(Y), Internet(Y), Genre(Y), Strategy(Y),Magazine(Y), News&Media(Y), Broadcast(Y),Internet(A)},{internet})

n17=({Genre(Y), Strategy(Y), Genre(L), Strategy(L)},{game, genre})

n18=({Strategy(Y), Strategy(L)},{game, genre,strategy})

n19=({Magazine(Y), Internet(A)},{internet,magazine})

n20=({Magazine(Y), Internet(A),Magazine(A)},{magazine})

n21=({Broadcast(Y), Broadcast(A)},{broadcast})

Bot=({},all key words)

Fig. 8 23 concpets are retrieved from the table (Fig. 7) using FCA

The context with 23 concepts is represented as a lattice in the form of

graph as follows:

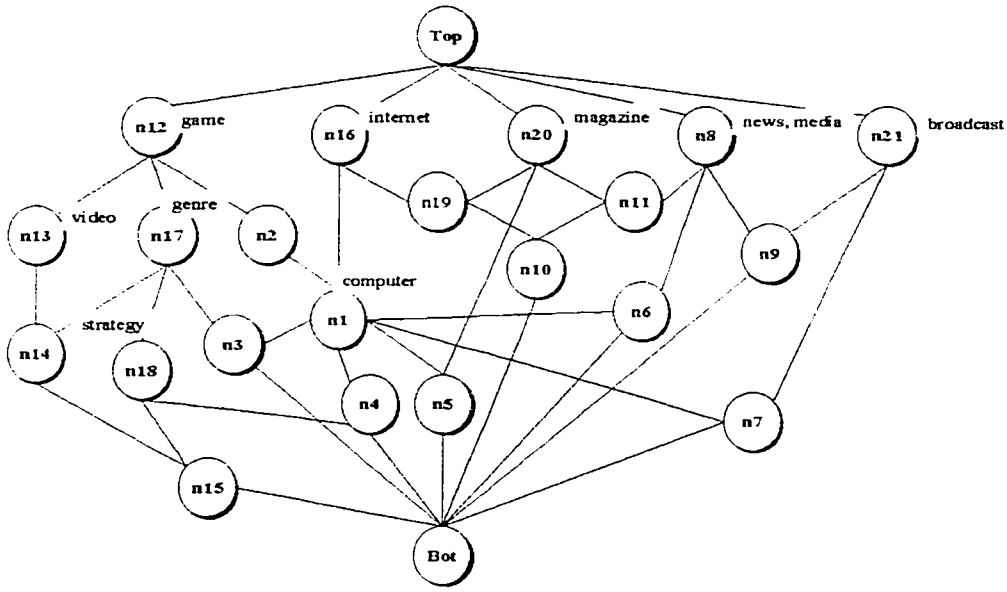


Fig. 9 A Lattice containing 23 concepts for the previous context

Users can locate their desired category by following a sequence of suggested keywords which is dynamically offered to them.

step1 :

At first, the system will suggest available key words from "Top". In this example, following 5 key words are available from the view of concept "Top"

{game, internet, magazine, news and media, broadcast}

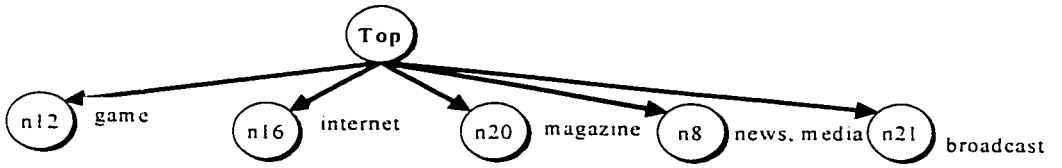


Fig. 10 From the "Top" concept, five subconcepts can be reached

step2

A user selects one key word and is provided the next available key words from my system repeatedly. Assuming a user selects "internet", then he will be in "n16" of the concept lattice and be provided two available key words: `{magazine, computer}`

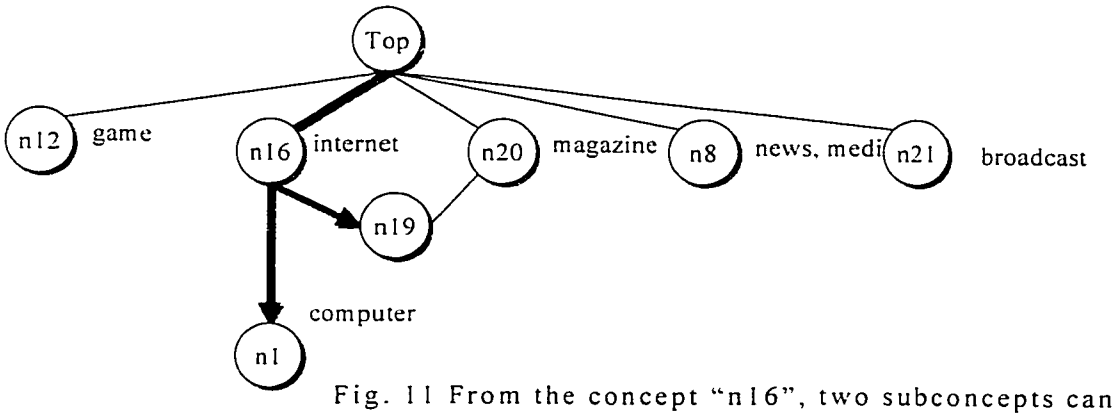


Fig. 11 From the concept "n16", two subconcepts can be reached

The concept "n16" will be shown on the screen depicted as follows:

* current key word : internet
Available directories...
Yahoo >>>>
Computer&Internet Game Internet Genre
Strategy Magazine New&Media Broadcast
Altavista >>>>
Internet

define your category ...next keywords are
magazine, computer

If the user then selects "computer", he will be in the concpet "n1" of the concept lattice and the system will provide following available key words: {game, game&genre, game&genre&strategy, news&media, broadcast, magazine}

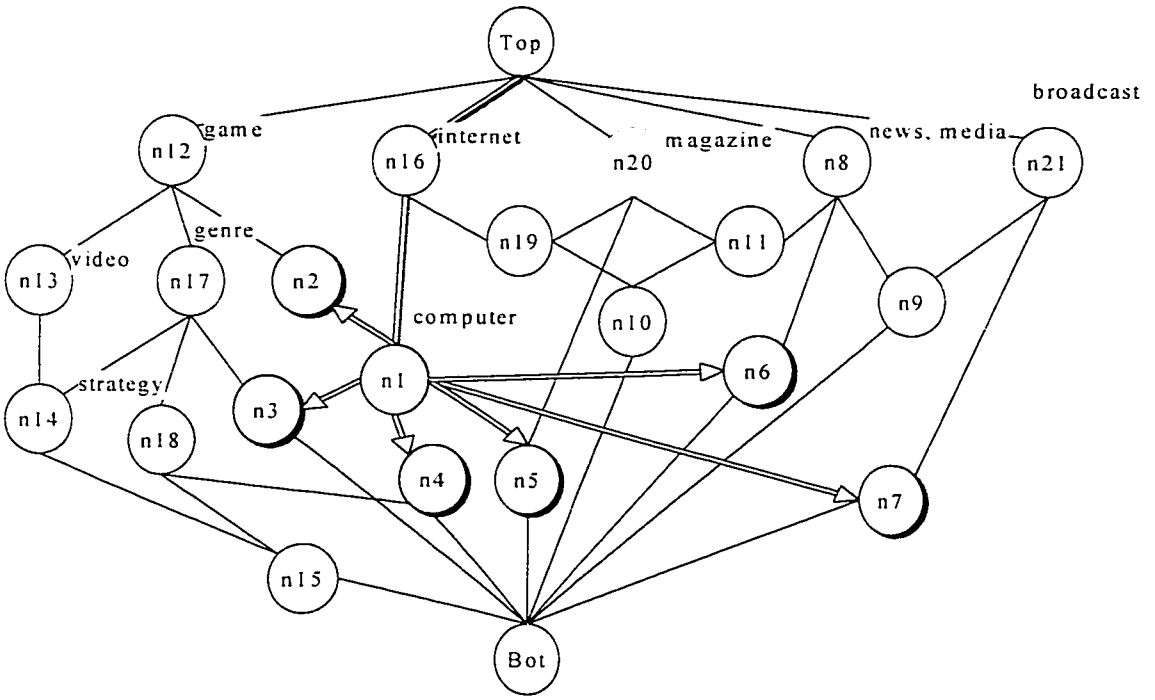


Fig. 12 From the concept "n1", six more subconcepts can be reached

In the same way as with the concept "n16", concept "n1" will be represented as Fig.13. On the screen, if the user selects category "game[11]", he can get a Yahoo page (Fig. 14) that includes a lot of web documents which have high possibility of exactness because the page is managed by humans.

* current key word : internet, computer

Available directories...

Yahoo >>>>

Computer&Internet **Game** Internet Genre
Strategy Magazine New&Media Broadcast

define your category ...next keywords are

game, game&genre, game & genre & strategy,
news&media, broadcast, magazine

Fig. 13 Prototype screen of concept "n1" in Fig. 12

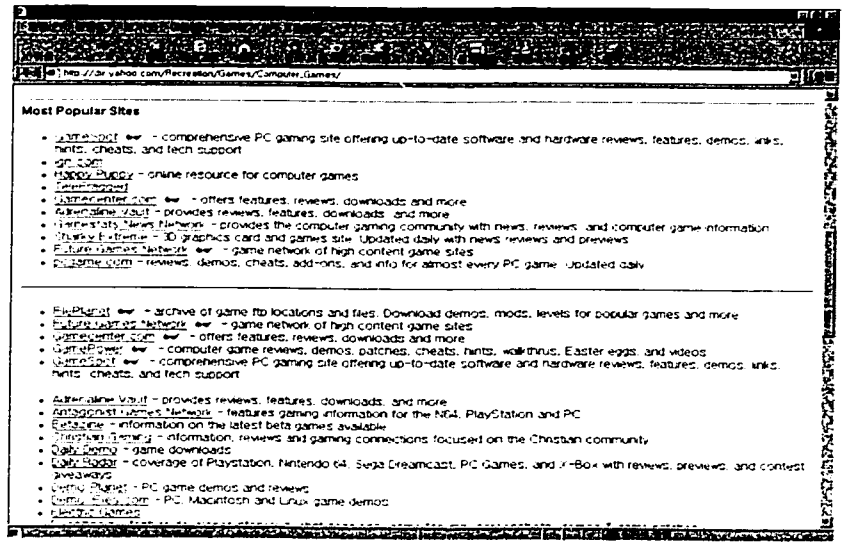


Fig. 14 Web links in the final target category

By this dynamic and interactive way, users can reach the most appropriate category having document links that they want to find. All the categories from the dominant web search engines will be suggested simultaneously. Therefore, users don't need to choose many search

[11] This is concept "n2" shown in the **Fig. 12**

engines separately. Source code descriptions and implementations for the prototype screens in this section are given in CHAPTER 5.

5. IMPLEMENTATION

My thesis implementation is composed of 3 parts.

- The first part is for gathering the information in the web. A detailed description is given in section 5.2
- The second part is for constructing data structures using data from the first part. Refer section 5.3
- The last part is for offering a interface to users. Section 5.4 is dedicated to that part of the implementation.

5.1 Description of Tools and Skills adopted

Programming Language

Among many candidate programming languages, *Perl* should be selected because of its powerful pattern matching ability which is the most important factor for my thesis implementation that deals with many web documents. Furthermore, Perl is the best one for web programming such as cgi scripts. The powerful supports for network programming (for example, LWP Library) were also an important point in selecting this language. (see APPENDIX F)

DBMS

I had three available databases on hand: *Oracle 8, Access2000, and MySql*. My 'Oracle 8' was a university student edition with many restrictions in its functionality. The capability of Access2000 didn't meet my thesis requirements; for example, it could have only 256 columns in a table. *MySql* was the best among them. It's free first of all, and performance was reported excellent (see APPENDIX E).

Perl and MySql connection

To deal with MySql in my perl code, I needed some bridge between 'perl' and 'MySql'. There were no other options here; I had to use *DBI* and *DBD:MySql* for the bridge. Although I couldn't select the other combinations for the bridge, the given combination 'perl <-> DBI <-> DBD:mysql <-> MySql' worked the best job for me (see APPENDIX F).

Web Server

My thesis will offer internet access for users. For that purpose, I needed a web server that would handle requests from the internet users. Some free web servers *Apache, Omnihttpd, and WeSite ver 1.1* were tested for a while. After the test period, *Apache* was selected because only *Apache* gave me a full control over the web server. I could handle all functionality through the file 'httpd.conf' (see APPENDIX D).

LWP module (APPENDIX G) in Perl mostly roles as a basis for a search robot. It offers simple ways of network programming for programmers. In my thesis, the sites that I want to find are done by simple search robot which is under control of my program helped by the LWP module. Details will be given in the next sections.

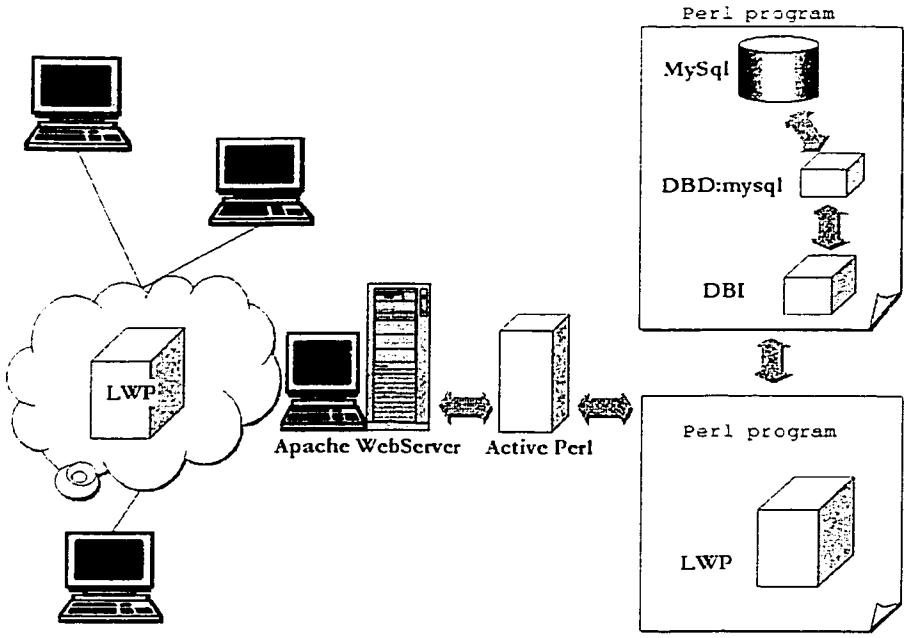


Fig. 15 Setup of Tools

5.2 Implementation of target sites collection using Search robot[12]

This is the first step of the whole thesis implementation. For this step a

[12] See 'APPENDIX A-I' for code details.

program named `indexing.pl'[13] recursively and automatically extracts *sets of a link and a key* from every page under category sites of dominant search engines like Yahoo, Lycos, Altavista and so on.

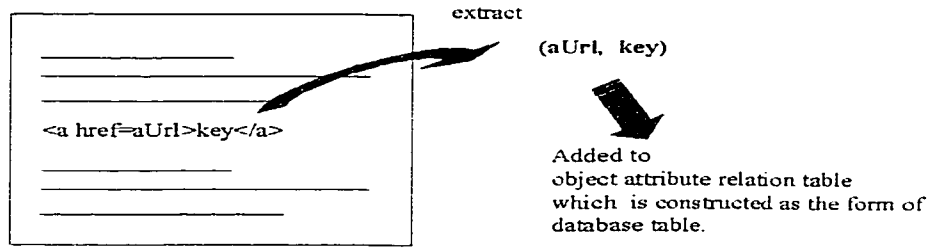


Fig. 16 Which information is extracted ?

The link part in the form of HTML format is represented as a "aUrl" and the word (phrase) between the tags <a> and is represented as a "key."

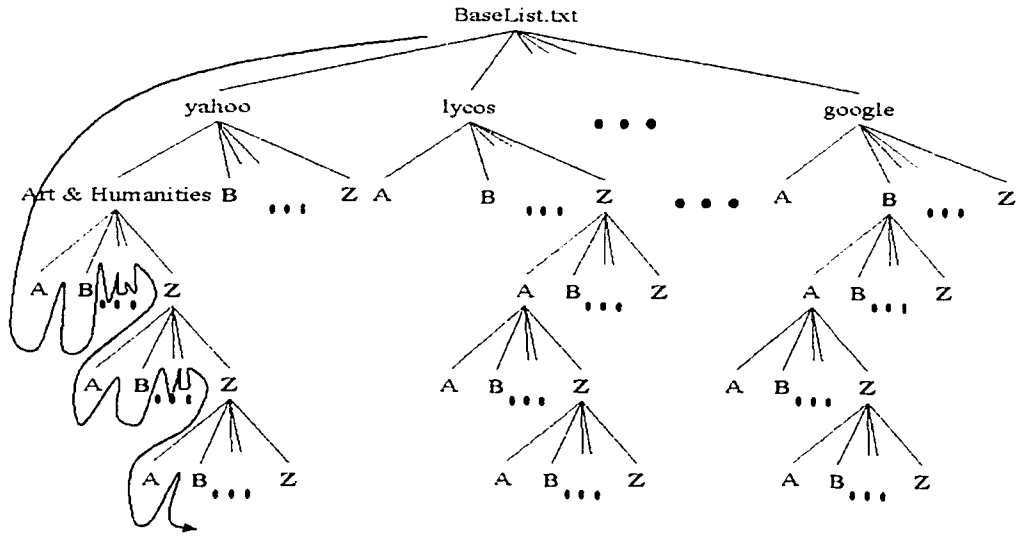


Fig. 17 Depth First Searching of a program 'indexing.pl'

[13] This program is applied to the example I in section 4.2. In a similar way(so description will be omitted), code in APPENDIX A-II is applied to the example II

This program crawls from upper most directory (for example, the first page of Yahoo site) to the lowest directory and grabs all the sets of the a link and a key recursively. The recursion is based on "Depth First Search Algorithm" as you can see from

Fig. 17.

The following is the description of the program 'indexing.pl' which collects target sites using Search robot

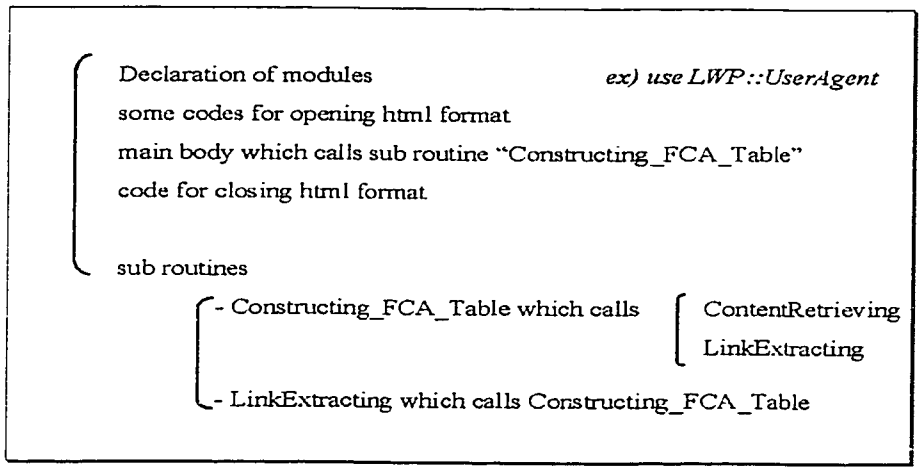


Fig. 18 The structure of a program 'indexing.pl'

Declarations of modules:

Functions of the declared modules can be imported by using the reserved word in Perl '`use`'. For example, by declaring '`use LWP::UserAgent`'[14] that is a module for a search robot (agent), I can use all the functions

---

in section 4.2

[14] See APPENDIX G for LWP module which facilitates many aspects of network

declared in the module. This software reuse aspect of Perl helps programmers to save their coding time and transplant bug-tolerant & neat code into their work.

Some codes for opening html format & code for closing html format:

This is a code for formatting the result from the above search robot into html text so that I can see the result through a web browser.

main body which calls sub routine "Constructing_FCA_Table" :

Between the html format codes, the main body which calls a sub routine is there. My search robot is driven by this sub-routine call. In the sub routine, other subroutine calls are made to give the program an ability of recursion. A sub routine named "AddConceptToFCA" is not related with NetWork programming but is related with the database. The result extracted will be stored in a table through this sub routine.

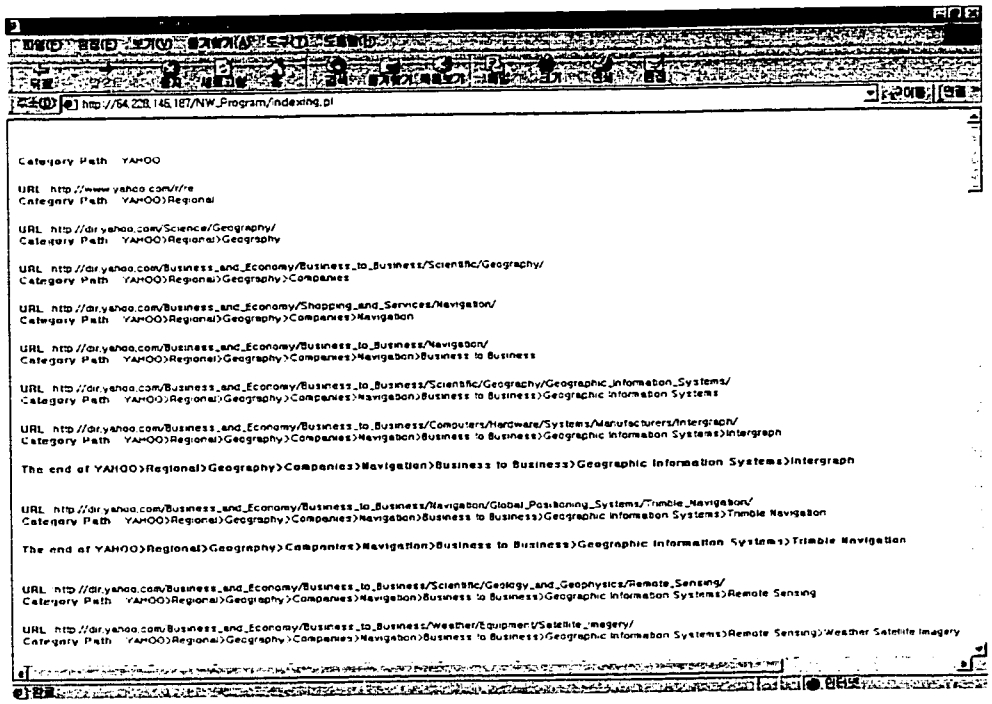


Fig. 19 A screen for monitoring the implementation of 'indexing.pl'[15]

As similar way, information extraction of example II in section 4.2 can be done and source code for the example is given in APPENDIX A-II. Instead of repeating the description of the details for example II, the next 2 parts of the implementation will be based on it.

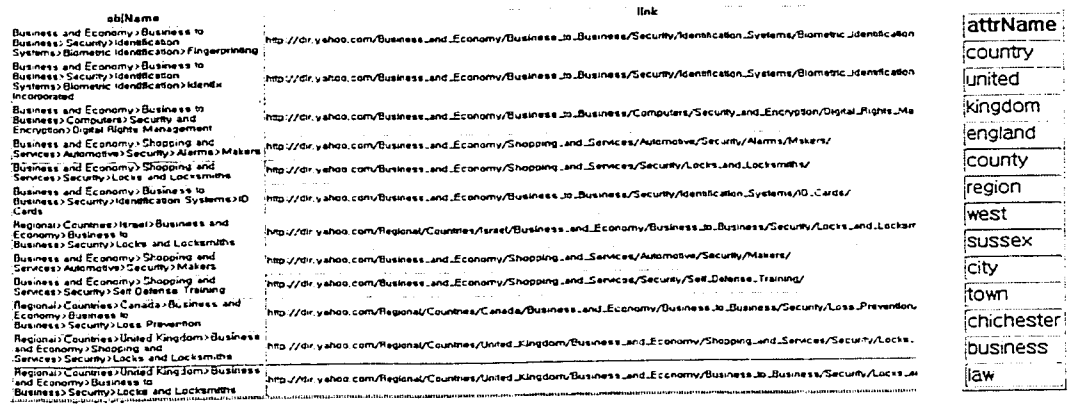
5.3 Construction of Lattice

From the previous section (5.2), an object-attribute relation table is obtained. A lattice is constructed using the relation table. Basically, I followed the algorithm of constructing a lattice given by Siff and Reps

[64]. That is to say, atomic concepts are constructed first and then the other concepts are extracted from the atomic concepts.

Detailed descriptions are given in APPENDIX B with the corresponding source code.

All the concepts are stored in the form of a table. For example, attributes and objects are represented as following tables:



objName	link
Business and Economy>Business to Business>Security>Identification Systems>Biometric Identification>Fingerprinting	http://dir.yahoo.com/Business_and_Economy/Business_to_Business/Security/Identification_Systems/Biometric_Identification
Business and Economy>Business to Business>Security>Identification Systems>Biometric Identification>Identix Incorporated	http://dir.yahoo.com/Business_and_Economy/Business_to_Business/Security/Identification_Systems/Biometric_Identification
Business and Economy>Business to Business>Computers>Security and Encryption>Digital Rights Management	http://dir.yahoo.com/Business_and_Economy/Business_to_Business/Computers/Security_and_Encryption/Digital_Rights_Ma
Business and Economy>Shopping and Services>Automotive>Security>Alarms>Makers	http://dir.yahoo.com/Business_and_Economy/Shopping_and_Services/Automotive/Security/Alarms/Makers/
Business and Economy>Shopping and Services>Security>Locks and Locksmiths	http://dir.yahoo.com/Business_and_Economy/Shopping_and_Services/Security/Locks_and_Locksmiths/
Business and Economy>Business to Business>Security>Identification Systems>ID Cards	http://dir.yahoo.com/Business_and_Economy/Business_to_Business/Security/Identification_Systems/ID_Cards/
Regional>Countries>Israel>Business and Economy>Business to Business>Security>Locks and Locksmiths	http://dir.yahoo.com/Regional/Countries/Israel/Business_and_Economy/Business_to_Business/Security/Locks_and_Locksm
Business and Economy>Shopping and Services>Automotive>Security>Makers	http://dir.yahoo.com/Business_and_Economy/Shopping_and_Services/Automotive/Security/Makers/
Business and Economy>Shopping and Services>Security>Self Defense Training	http://dir.yahoo.com/Business_and_Economy/Shopping_and_Services/Security/Self_Defense_Training/
Regional>Countries>Canada>Business and Economy>Business to Business>Security>Loss Prevention	http://dir.yahoo.com/Regional/Countries/Canada/Business_and_Economy/Business_to_Business/Security/Loss_Prevention/
Regional>Countries>United Kingdom>Business and Economy>Shopping and Services>Security>Locks and Locksmiths	http://dir.yahoo.com/Regional/Countries/United_Kingdom/Business_and_Economy/Shopping_and_Services/Security/Locks_
Regional>Countries>United Kingdom>Business and Economy>Business to Business>Security>Locks and Locksmiths	http://dir.yahoo.com/Regional/Countries/United_Kingdom/Business_and_Economy/Business_to_Business/Security/Locks_a

attrName
country
united
kingdom
england
county
region
west
sussex
city
town
chichester
business
law

Fig. 20 Each object and attribute are represented in the form of a table

A lattice is also represented in the form of a database table that has its column components as 'conceptNumber', 'upLink', 'downLink', 'attributes', and 'level'[16]. Following is a screen shot showing a part of

[15] APPENDIX A-I
[16] level is the length from the top(root) of the lattice.

the lattice.

conceptNum	
1	733,734,735,736,737
2	733,734,737,738
3	734,739
4	740,741,742
5	740,741,742,743,744,745,746,747
6	741,748
7	741,749
8	741,743,750,751,752,753
9	741,750,751,752,753
10	741
11	10,741,743,754
12	741,755,756
13	741,755,757
14	741,755,758,759,760,761
15	741,762,763,764
16	15,741,762,763,764,765
17	741,766,767,768
18	741,766,769,770,771,772
19	741,767,768,773,774,775,776,777,778,779,780,781,705,786,1017
20	741,773,782,783,784,785,786,787,1017
21	741,743,788,789,790,791,792,793
22	741,754,788,789,791,792,793
23	22,741,754,788,789,791,792,793
24	741,788,789,791,792,793
25	741,788,789,791,792,793,794
26	741,767,768,789,792,795,796,797,798,799,800,801
27	741,774,789,792,802,803,804,805,871
28	741,789,792,802,804,805,871

Fig. 21 A part of the table which represents a lattice

UpLink and downLink are represented as a string of concept numbers separated by a comma (','). An upLink is a list of superconcept numbers and a downLink is a list subconcept numbers.

5.4 A Dynamic Graphical User Interface

All five relatively small programs are combined to offer users an interface (**Fig. 22**). Every content is filled with information based on the users' query or request (action) and the information is retrieved from the lattice as described in section 5.4

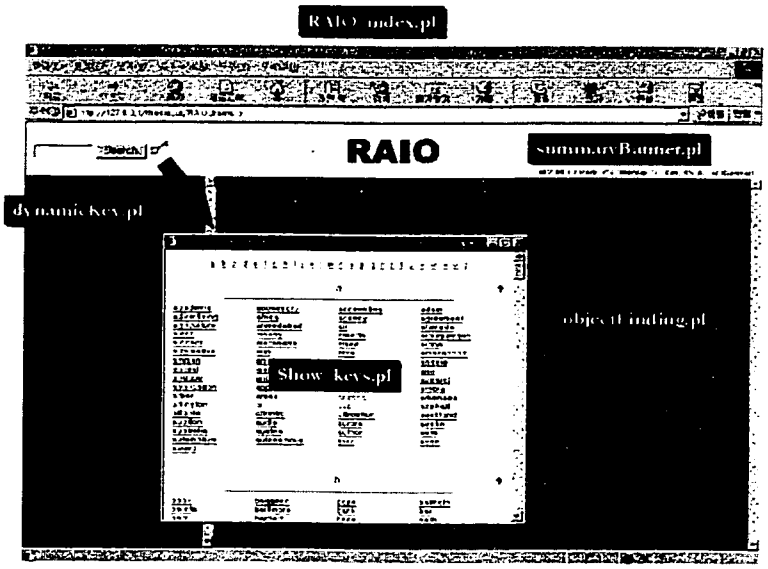


Fig. 22 The configuration of programs consisting a user interface

To see the action of the user interface, assume the following case:

A User wants to access pages of

'Government Law related with patents'

At first, a user will see the first screen if he connects to my thesis implementation page shown in **Fig. 23**. Keywords seen at the "Top" of the lattice are suggested on the left side of the first screen. The user can select a keyword that is appropriate for their searching. In this example keyword 'government' may be selected. Of course, other paths are possible. For example, the user may start with keyword 'law' or 'patent'.

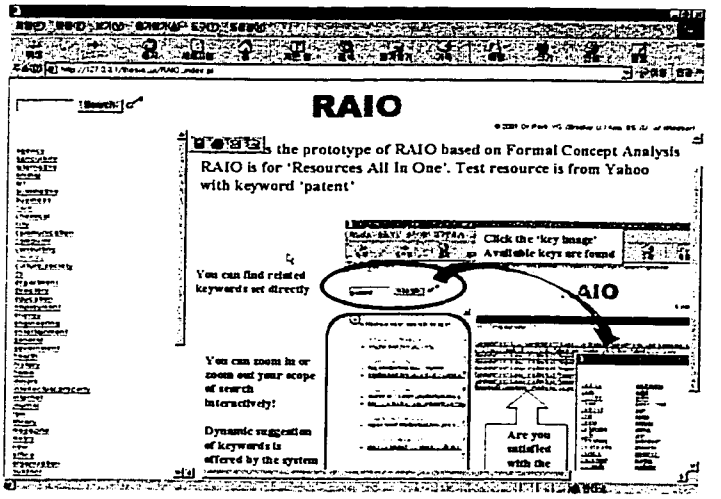


Fig. 23 The first screen of user interface

As the return for a selection of a keyword 'government' from the user, more available keywords seen from the current keyword ('government') are suggested on the left side of the user interface and links that are related with the current keyword are shown on the right side (Fig. 24).

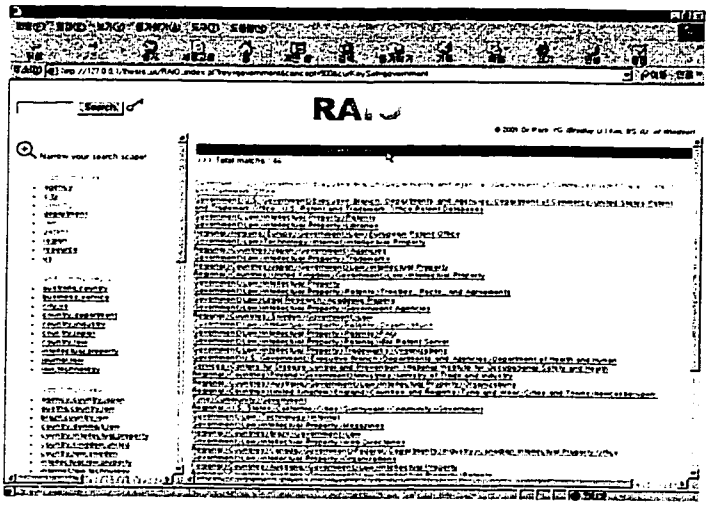


Fig. 24 The resulted screen when a keyword 'government' is selected

In a similar way to the previous step, the user can narrow his category by

selecting one among the suggested keywords. **Fig. 25** shows a dynamic screen for the keyword 'law'.

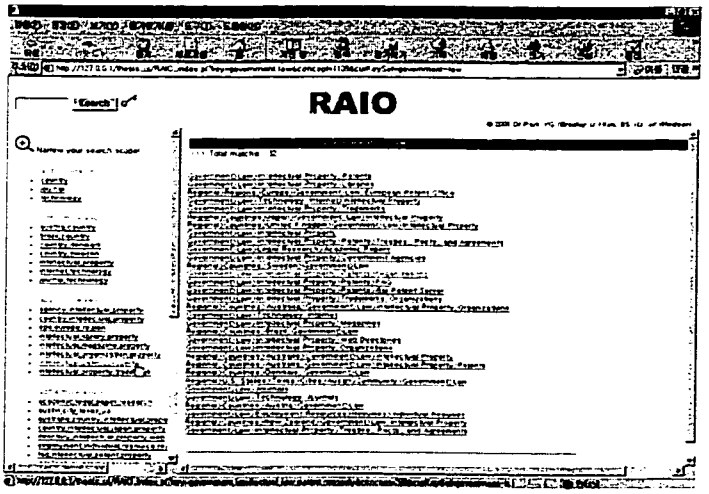


Fig. 25 The resulted screen when a keyword 'law' is selected after 'government'

The final selection of a keyword[17] leads the user to the following screen **(Fig. 26)**.

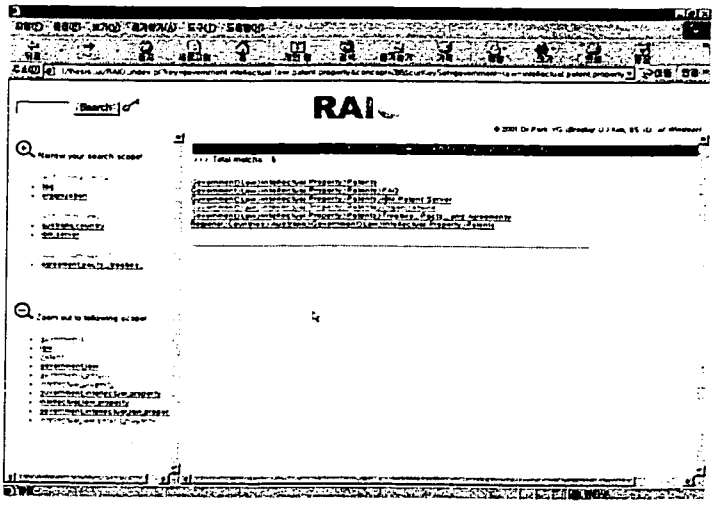


Fig. 26 The final screen for keywords: government>law>intellectual,patent,property

[17] In this example, it's a keywords set : 'intllectual, patent, property'

The keywords trace that has been selected by the user is also shown in the blue strip under title RAIO.

As explained in the previous example, dynamic aspect of this approach satisfies the user's requirement supposed in CHAPTER 3. That is to say, a user narrows his scope of search with only one keyword (or a keywords set) that is dynamically suggested by the system.

Another facility using 'search box' which accepts keywords from a user, gives an alternative shortcut to a designated result. With the given query case, the user may obtain a list of available keyword sets having the keywords 'law, government, patent' as a part.

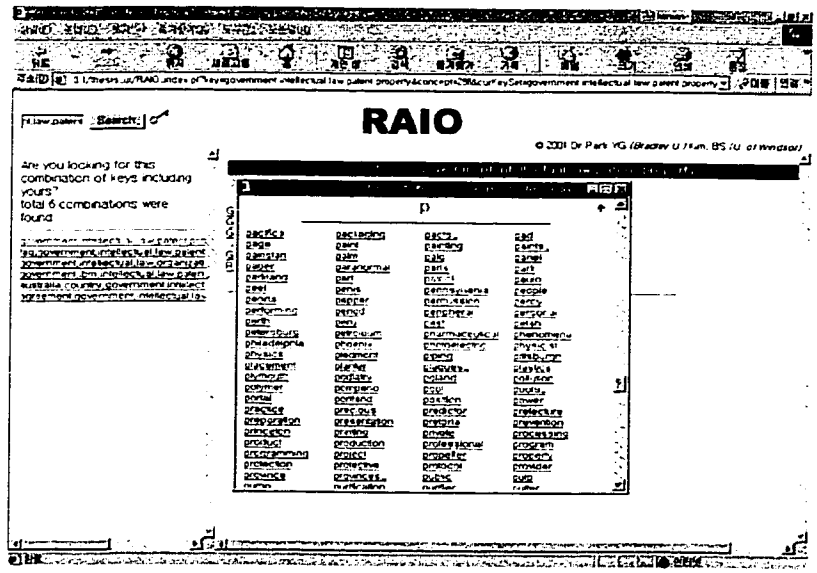


Fig. 27 An another way of searching offered by the system

6. Evaluation and Future Work

As shown in the previous sections, the interest of my thesis is not in a general search engine but in a specialized (interactive, dynamic, specific domain) one. That's the point at which I could adopt Formal Concept Analysis although it requires a huge time allotment for making a lattice.

We have demonstrated that FCA is a promising tool in developing an effective and revolutionary search engine. With the aid of a lattice, dynamic and interactive web search engines that focus on the specific domain such as the following may be possible:

a. categories in the major search engines that offer directory service
b. sites related with patent information
c. sites of specialized search engines that offers much more excellent results than general search engines in some specific topics
d. a site which manages a lot of web pages like ESPN, CNN and so on

However, there are many problems remaining. The most important ones are: 'How to locate the desired sites in the web' and 'How to create a lattice as fast as possible' My experiment shows that around 5 days were needed to create a lattice with 2058 concepts. More reasonable time

complexity is thus required[18].

My thesis suggests the way of new searching methodology and the scope of searching is much smaller than that of a general search engine[19].

Besides improving the time complexity of lattice construction, another elaborate algorithm for gathering the information in the web is also needed to apply this new methodolgy to a commercial domain.

Although the above 2 research areas could be very difficult, it's worth the challenge.

---

[18] The worst case scenario of time complexity for constructing a lattice is O(n^n). But with the fast algorithm by Harvard University, a time complexity of O(n^4) is possible.

[19] It can be said to be a niche search engine.

REFERENCES

[1] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, Janet Wiener, *The Lorel Query Language for Semistructured Data*, Journal of Digital Libraries, 1(1):68-88, April 1997

[2] Robert Armstrong, Dayne Freitag, Thorsten Joachims, Tom Mitchell, *WebWatcher: A Learning Apprentice for the World Wide Web*, AAAI Spring Symposium on Information Gathering

[3] Marko Balabanovic, *An Adaptive Web Page Recommendation Service*, Proceedings of 1st International Conference on Autonomous Agents

[4] J.M. Barrie and D.E. Presti, *The World Wide Web as an instructional Tool*,Science vol 274, page 371(1996)

[5] K. Bharat and MR Henzinger, *Improved Algorithms for Topic Distillation in a Hyperlinked Environment*, 21st ACM SIGIR Conference on Research and Development in Information Retrieval

[6] P.Bruza, R. McArthur, and S. Dennis, *Interactive Internet Search:Keyword, directory and query formulation mechanisms compared*, In proceedings of the SIGIR Conference, 2000

[7] Jay Budzik, Kristian Hammond, Cameron Marlow, and Andrei Scheinkman, *Anticipating Information Needs: Everyday Applications as Interfaces to Internet Information Resources*, In Proceedings of the 1998 World Conference on the WWW, Internet, and Intranet. AACE Press, (1998)

[8] Jay Budzik, Kristian J. Hammond, Larry Birnbaum, and Marko Krema Intelligent., *Beyond Similarity*, Proceedings of the 2000 Workshop on Artificial Intelligence and Web Search

[9] Jay Budzik, Kristian J. Hammond, *User Interactions with Everyday Applications as Context for Just-in-time Information Access*, Proceedings of the 2000 International Conference on Intelligent User Interfaces

[10] Peter Buneman, Susan Davidson, Gerd Hillebrand, Dan Suciu, *A Query Language and Optimization Techniques for Unstructured Data*, In SIGMOD, 1996

[11] Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, *Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text*, In

Proceedings of the Seventh International World Wide Web Conference, Brisbane, Australia, April 1998

[12] ELSEVIER Soumen Chakrabarti, Martin van den Berg, Byron Dom, *Focused crawling: a new approach to topic-specific Web resource discovery*, In Proceedings of the 8th World-Wide Web Conference, 1999

[13] Junghoo Cho, Hector Garcia-Molina, Lawrence Page, *Efficient Crawling Through URL Ordering*, in the Proceedings of the 7 th International WWW Conference, 1998

[14] Frans Coetzee, Eric Glover, Steve Lawrence, and C. Lee Giles., *Feature selection in web applications using ROC inflections*, In Symposium on Applications and the Internet, SAINT, San Diego, CA, January 8--12 2001

[15] M.P. Consens, F.Ch.Eiger, M.Z. Hasan, A.O. Mendelzon, E.G. Noik, A.G.Ryman, and D.Vista., *Architecture and applications of the hy+ visualization system.*, IBM Systems Journal, 33:3:458-476,1994

[16] Nick Craswell and Peter Bailey David Hawking, *Server selection on the World Wide Web*, In Proceedings of the Fifth ACM Conference on Digital Libraries, pages 37--46, 2000

[17] M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, M. Gori, *Focused Crawling Using Context Graphs*, In Proc. Very Large Data Bases 2000 (VLDB 2000), September 2000. To appear.

[18] Daniel Dreilinger, *Experiences with Selecting Search Engines using Meta-Search*, ACM Transactions on Information Systems, 15(3), 1997, 195-222

[19] Oren Etzioni, Daniel Weld, *A Softbot-Based Interface to the Internet*, Communications of ACM 37(7): 72-76, 1994

[20] Mary Fernandez Daniela Florescu Alon Levy Dan Suciu, *A Query Language for a Web-Site Management System*, SIGMOD Record, 26(3): pp. 4-11, September 1997

[21] Gary William Flake, Steve Lawrence, C. Lee Giles, *Efficient Identification of Web Communities*, In Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, August 20-23 2000, pp.150-160

[22] Daniela Florescu, Alon Levy, Alberto Mendelzon, *Database Techniques for the World-Wide Web: A Survey*, SIGMOD Record (ACM Special Interest Group on Management of Data)

[23] Bernhard Ganter and Rudolf Wille, *Formal Concept Analysis: mathematical Foundations.*, Springer-Verlag(1999)

[24] Susan Gauch, Guijun Wang, Mario Gomez, *ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines*, in the Journal of Universal Computer Science, Volume 2. Number 9, Sept. 1996

[25] David Gibson, *Inferring Web Communities from Link Topology,* In Conference on Hypertext and Hypermedia ACM, 1998. 166

[26] Eric J. Glover, Steve Lawrence, William P. Birmingham, and C. Lee Giles, *Architecture of a metasearch engine that supports user information needs*, In Eighth International Conference on Information and Knowledge Management (CIKM'99), pages 210-- 216, Kansas City, MO, November 1999. ACM Press

[27] Eric Glover, Gary Flake, Steve Lawrence, William P. Birmingham, Andries Kruger, C. Lee Giles, David M. Pennock, *Improving category specific web search by learning query modifications*, . In Symposium on Applications and the Internet, SAINT, San Diego, CA, January 8--12 2001

[28] Eric J. Glover, Steve Lawrence, Michael D. Gordon, William P. Birmingham, and C. Lee Giles., *Web search -- your way*, Communications of the ACM, 2000. Accepted for publication

[29] Roy Goldman, Jennifer Widom, *DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases*, Twenty-Third International Conference on Very Large Data Bases

[30] Roy Goldman, *Jennifer Widom, Interactive Query and Search in Semistructured Databases*, In WebDB'98, Proc. Int. Workshop on the Web and Databases, 1998

[31] Luis Gravano, Hector Garcia-Molina, *Generalizing gloss to vector-space databases and broker hierarchies*, In Proceedings of the 21st VLDB Conference (Zurich, Switchland, 1995)

[32] Luis Gravano Columbia University and Hector Garca-Molina, *GlOSS: Text-Source Discovery over the Internet*, to appear in ACM Transactions on Database Systems, 1999

[33] Luis Gravano, Chen-Chuan K. Chang, Hector Garcia-Molina, Andreas Paepcke, *STARTS:*

Stanford Proposal for Internet Meta-Searching, SIGMOD 1997, pp. 207--218

[34] M. Gray, *Measuring the growth of the Web:June 1993 to June 1995*, http://www.mit.edu/people/mkgray/growth/(1996)

[35] Graphic, Visualization, and Usability Center at Georgia Tech. 1998., 9 th WWW User Survey

[36] Jansen, B. J. and Pooch, *A Review of Web Searching Studies and a Framework for Future Research*, U. 2000. Web user studies

[37] Jupiter Research(1999) , *Go Network Announces New INFOSEEK Search:30 Percent Faster, 50 Percent Larger.*, http://info.go.com/press/search.html

[38] Karger, David and Lynn Stein, *haystack: per-user information environments*, In Proceedings of the 1999 Conference on Information and Knowledge Management, CIKM, 1999

[39] JON M. KLEINBERG, *Authoritative Sources in a Hyperlinked Environment*, Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, 1998

[40] Krishna, B. & Broder, A, *A technique for measuring the relative size and overlap of public Web search engines.*, Proc. Of the 7th International World Wide Web Conference

[41] Andries Kruger, C. Lee Giles, Frans M. Coetzee, Eric Glover, Gary W. Flake, Steve Lawrence, Christian Omlin, DEADLINER: Building a New Niche Search Engine 1-37, Ninth International Conference on Information and Knowledge Management, CIKM 2000

[42] Steve Lawrence and C.Lee Giles., *Accessibility of information on the web*, Nature, 4000(July 8):107-109,1999.

[43] Steve Lawrence, *Context in Web Search*, IEEE Data Engineering Bulletin, Vol23, Num 3, pp 25-32,2000

[44] Steve lawrence, Kurt Bollacker, and C. Lee Giles, *Digital libraries and autonomous citation indexing*, IEEE Computer, 32(6):67-71,1999

[45] Steve lawrence, Kurt Bollacker, and C. Lee Giles, *Indexing and retrieval of scientific literature*, In 8th International Conference on Information and Knowledge Management, CIKM 99, pp139-146, Kansas City, Missouri, Nov.1999

[46] Steve Lawrence, C. Lee Giles, *Searching the web: General and scientific information access*, IEEE Communications, 37(1):116--122, 1999

[47] Steve Lawrence, C. Lee Giles, *Searching the world wide web*, Science, 280(5360):98. 1998

[48] David B. Leake, Ryan Scherle, Jay Budzik, Kristian Hammond, *Selecting Task-Relevant Sources for Just-in-Time Retrieval*, In Proceedings of the AAAI-99 Workshop on Intelligent Information Systems, Menlo Park, CA, 1999. AAAI Press

[49] Henry Lieberman, *Letizia:An agent that assists Web browsing*, In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, pages 924--929. Morgan Kaufmann, 1995

[50] Weiyi Meng, King-Lup Liu, Clement Yu, Wensheng Wu, Naphtali Rishe, *Estimating the Usefulness of Search Engines*, Proc. Of the 15th International Conference on Data Engineering (ICDE'99), Sydney, Australia, March 1999

[51] Kenneth Moore and Michelle Peterson, *A groupware benchmark based on lotus notes.*, In proc. Of Int. Conf. On data Engineering(ICDE), 1996

[52] Yannis Papakonstantinou, Ashish Gupta, Hector Garcia-Molina, and Jeffery Ullman, *A query translation scheme for rapid implementation of wrappers.*, In Proc. Of the Int. Conf. On Deductive and Object-Oriented Databases (DOOD), 1995

[53] Young Park, *Polymorphic Function Retrieval via Formal Concept Analysis,* Proceedings of the AoM/IAoM International Conference on Computer Science(ICCS), pp165-170,1999

[54] Young Park, *Software retrieval by samples using concept analysis*, Journals of Systems and Software, 54(3), pp. 179-183,2000

[55] M. Pazzani, J. Muramatsu, and D. Billsus, *Identifying interesting web sites*, In Proceedings of the National Conference on Artifitial Intelligence

[56] Raghu Ramakrishnan and Johannes Gehrke, *Database Management Systems*, 2nd Edition(2000)

[57] Jason Rennie, *Andrew McCallum, Using Reinforcement Learning to Spider the Web Efficiently*, Proc. 16th International Conf. On Machine Learning

[58] Bradley James Rhodes, *Just-In-Time Information Retrieval*, Ph.D thesis, MIT June. 2000

[59] Bradley J. Rhodes, *Margin Notes: Building a Contextually Aware Associative Memory*, The Proceedings of the International Conference on Intelligent User Interfaces (IUI '00), New Orleans, LA, January 9-12, 2000

[60] Bradley J. Rhodes,Thad Starner, *Remembrance Agent: A continuously running automated information retrieval system*, The Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology (PAAM '96), pp. 487-495

[61] G.Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York(1983)

[62] B.R. Schatz, *Information Retrieval in Digital Libraries: Bringing Search to the Net*, Science, Jan. 1997

[63] E.Selberg and O. Etzioni., *The MetaCrawler architecture for resource aggregation on the web.*, IEEE Expert,(Jan-Feb.):11-14,1997

[64] Michael Siff and Thomas Reps, *Identifying modules via concept analysis.* In International Conference on Software Maintenance, ICSM97 (1997), IEEE Computer Society

[65] Sparck-Jones, K. & Willet, P(Eds.)(1997), *Readings in Information Retrieval*, San Fransisco:Morgan Kaufman

[66] Starwave Corp., *About ESPN*, http://www.starwave.com/starwave/about.espn.html (1996)

[67] Starwave Corp., *ESPN SportsZone surpasses 2 billion page views.*, http://www.starwave.com/starwave/release.sz.billion.html (1997)

[68] G.Taubes., *Science Journals Go Wired*, Science vol.171, page 764.

[69] J.Thierry-Mieg and R.Durbin., *A C.elegans database:syntactic definitions for the ACEDB database manager.* 1992

[70] Thomas Tilley, *Formal Concept Analysis And Formal Methods* (2000)

[71] R. Wille, *Restructuring Lattice Theory: An Approach Based on Hierarchies of Concepts*, Ordered Sets, D. Reidtel, Dordrecht, 1982, pp. 445--470

[72] Jinxi Xu and Jamie Callan, *Effective Retrieval with Distributed Collections*, In Proceedings of the 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (Melbourne, Australia, 1998), pp. 112--120

[73] Xu, J.L., *Interent Search Engines:real World IR Issues and Chanllenges.*, Conference on Information and Knowledge Management. Kansas City, Missouri

[74] Osmar R. Zaane 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.

[75] Justin Zobel, *Collection Selection via Lexicon Inspection*

APPENDIX A : SOURCE CODE FOR SECTION 5.2 (TARGET SITES COLLECTION USING SEARCH ROBOT)

I. Code for collecting directories information (url, keyword, and name of Yahoo directories)

```
#!c:/perl/bin/perl.exe                                          # location of perl program

use LWP::UserAgent;
use HTTP::Request;
use HTTP::Response;
use HTML::Parse;
use URI::URL;
use DBI;

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++                                    Main                                                  ++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Start_HTML();
FileWrite("LinkList","");                                       # initialization of LinkList.txt
open(BaseList, "BaseList.txt");
       while(!eof(BaseList)){
          $Line=<BaseList>;
          $Line =~ /(.*),(.*)/;
          Constructing_FCA_Table($1, $2);
```

```
	}
close(BaseList);
End_HTML();
exit;

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++                              Subroutines                                                      ++
#++              italic font : for monitoring current execution status                            ++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
sub Constructing_FCA_Table{
	my $parent_dir = $_[0];
	my $parent_url = $_[1];
	print "<b>Category Path : </b>$parent_dir<br><p>";
	my $content=ContentRetrieving($parent_url);
	LinkExtracting($parent_dir, $parent_url, $content);
}

sub ContentRetrieving{
	my $current_url=$_[0];
	my $ua=new LWP::UserAgent;
	my $request=new HTTP::Request('GET',$current_url);
	my $response=$ua->request($request);
	if ($response->is_success){
			$rtn=$response->content;
```

```
        }
        elsif($response->is_info){
                $rtn= $response->code;
                print "response code is $rtn\n<br>";
        }
        elsif($response->is_redirect){
                $rtn=$response->code;
                print "response code is $rtn\n<br>";
        }
        else{
                print $response->error_as_HTML;
                $rtn= $response->code;
        }
        return $rtn;
}

sub LinkExtracting{
        my $parent_dir= $_[0];                  # $_[0] Accumulated dir
        my $parent_url=$_[1];                   # $_[1] Link
        my $content=$_[2];                      # $_[2] content
        my %link;
        my %link_leaf;
        my $final="No";
        my $tags="<[^>]*>";
```

```
if ($content =~ /<base href=([^>]*)>/g){
   $parent_url=$1;
}
if ($parent_dir eq 'YAHOO'){
   %link=$content=~/<a href=(?:"|)([^>]*)(?:"|)>(?:$tags)*([^<@]*)(?:@|)</gi;
}
elsif ($parent_dir =~ /YAHOO/){
            if ($content =~ /Categories(.*)<hr/s){
                     my $tmp=$1;
                     %link=$tmp=~/<a href=(?:"|)([^>\*]*)(?:"|)>(?:$tags)*([^<@]*)(?:@|)</gi;
                     if($tmp=~/<a href=(?:"|)([^\*]*[^>]*)(?:"|)>([^<]*)<)/i){
                              %link_leaf=$tmp=~/<a href=(?:"|)([^\*]*[^>]*)(?:"|)>([^<@]*)(?:@|)<)/gi;
                     }
            }
            else{
                     %link_leaf=$content=~/<a href=(?:"|)([^\*]*[^>]*)(?:"|)>([^<]*)<)/gi;
                     $final="Yes";
            }
}
else{       } #do nothing
if ($final eq 'No'){
            foreach my $aUrl(sort keys %link){
                     my $attribute= $link{$aUrl};
                     my $object = $parent_dir.">".$attribute;
```

```
                my $relative_url = new URI::URL($aUrl);
                my $absolute_url = $relative_url->abs($parent_url);
                print "<font color=blue><b>URL :</b></font>$absolute_url\n";

        $lists=FileRead("LinkList");
        if  ($lists !~
                FileWrite("LinkList","$lists$absolute_url\#\!");                    #  url1#!url2#!...urln#!
                Constructing_FCA_Table($object,$absolute_url);
        }
        else{     } #do nothing
    }
}
foreach my $aUrl(sort keys %link_leaf){
        my $attribute=$link_leaf{$aUrl};
        my $object = $parent_dir.">".$attribute;
        my $relative_url = new URI::URL($aUrl);
        my $absolute_url = $relative_url->abs($parent_url);
        print "<font color=blue><b>URL :</b></font>$absolute_url\n";
        $lists=FileRead("LinkList");
        if  ($lists !~
                FileWrite("LinkList","$lists$absolute_url\#\!");                    #  url1#!url2#!...urln#!
        }
        else{ } #do nothing
}
```

```
        print "<font color=green><b>The end of $parent_dir<p></b></font><br>\n";
}

sub FileRead{
        $FileHandle="FH4Read_$_[0]";
        open ($FileHandle,"$_[0].txt");
           my $result=<$FileHandle>;
           $result=~ s/\n//g;
        close($FileHandle);
        return $result;
}

sub FileWrite{
        $FileHandle="FH4Write_$_[0]";
        open ($FileHandle,">$_[0].txt");
           print $FileHandle $_[1];
        close($FileHandle);
}
sub Start_HTML{
        print "Content-type: text/html\n\n";
        print "<html><head><title>Thesis Phase1</title></head>";
        print  "<body bgcolor=\white'><font size=2>";
        print "\n<font color=black><br>\n";
}
```

```
sub End_HTML{

        print "Congratulations - Termination of Phase 1 <br>";

        print "</body></html>";

}
```

II. Code for collecting sites related with term 'patent' (in Yahoo)

```
#!c:/perl/bin/perl.exe
use LWP::UserAgent;
use HTTP::Request;
use HTTP::Response;
use HTML::Parse;
use URI::URL;
use DBI;

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++                                    Main                                              ++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Start_HTML();
my @link_dir;
my @link_site;
my $link4nextSearchCondition;
my $nextSearchFlag=0;                                          # Initially block next search
my %k_List;
my $l_List;
my $dbh = DBI->connect("DBI:mysql:database=thesis","usr","pw",{PrintError => 1, RaiseError => 0});
$yahoo_patent_search="http://search.yahoo.com/search?p=patent&d=y&o=i&h=s&g=0&n=100";
my $content=ContentRetrieving($yahoo_patent_search);
```

```
setup_phase1();                                                                    # prepare phase1 environment;
LinkNObjectExtracting(\$content,\@link_dir, \@link_site,\$link4nextSearchCondition,\$nextSearchFlag);
while($nextSearchFlag){
        $content=ContentRetrieving($link4nextSearchCondition);
        LinkNObjectExtracting(\$content,\@link_dir,\@link_site,\$link4nextSearchCondition,\$nextSearchFlag);
}
$cnt=1;
for $tuple(0..$#link_dir){
        my ($obj,$attr)=prepareAttr_dir($link_dir[$tuple]{'object'},$link_dir[$tuple]{'attribute'});
        FCATbl($obj,$attr,$link_dir[$tuple]{'link'},\%k_List,\%l_list);
        print "<font color=green>end of $cnt</font><p>";
$cnt++;
}
makeKeyWordDB(\%k_List);
$dbh->disconnect();
End_HTML();
exit;


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++                                          Subroutines                                                    ++
#++                                  italic font : for monitoring current execution status                  ++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
sub Start_HTML{
        print "Content-type: text/html\n\n";
```

```
        print   "<html><head><title>Thesis Phase1</title><META HTTP-EQUIV=Content-Language content=KO></head>";
        print   "<body bgcolor=\white'><font size=2>";
        print "\n<font color=black><br>\n";
}

sub End_HTML{
        print "Congraturations - Termination of Phase_1 <br>";
        print "<a href=./thesisDBmonitor.pl target=_blank><font size=6> Do you wanna see the result details? -o.o-</font></a>\n";
        print "</body></html>";
}

sub setup_phase1{
        FileWrite("obj_attr_List","\t");
        $dbh->do("drop table fca");
        $dbh->do("create table fca (object varchar(255), link varchar(255))");
        $dbh->do("drop table k_list");
        $dbh->do("create table k_list (itm varchar(255))") or print "<b>Error occured on add</b> : <font face=Courier>".$dbh->errstr()."<br></font>\n>";
        print "use thesis\;\n";
}

sub makeKeyWordDB{
        $key_hashref=$_[0];

        foreach $aKey(keys %{$key_hashref}){
```

```
        $key_ins=qq/insert into k_list (itm) values ('$aKey')/;
        $dbh->do($key_ins) or print "<b>Error occured on add</b> : <font face=Courier>".$dbh->errstr()."<br></font>\n>";
    }

sub ContentRetrieving{
    my $current_url=$_[0];

    my $ua=new LWP::UserAgent;
    my $request=new HTTP::Request('GET',$current_url);
    my $response=$ua->request($request);

    if ($response->is_success){
            $rtn=$response->content;
    }
    elsif($response->is_info){
          $rtn= $response->code;
    }
    elsif($response->is_redirect){
        $rtn=$response->code;
    }
    else{
        print $response->error_as_HTML;
        $rtn= $response->code;
    }
```

```
	return $rtn;
}

sub LinkNObjectExtracting{
	my $content_ref=$_[0];
	my $link_dir_ref=$_[1];
	my $link_site_ref=$_[2];
	my $link4nextSearchCondition_ref=$_[3];
	my $nextSearchflag_ref=$_[4];

	doDirInfoExtracting($content_ref,$link_dir_ref);
	doSiteInfoExtracting($content_ref,$link_site_ref);
	AskNextSearchPossible($content_ref,$link4nextSearchCondition_ref,$nextSearchflag_ref);
}

sub doDirInfoExtracting{
	@ObjLinkDirWrap=${$_[0]}=~/<dt>(.*?)<\/dt>/gs;
	foreach $aAttrLinkObj(@ObjLinkDirWrap){
	   $aAttrLinkObj=~/.*?>(.*?)\s*&gt;\s*<a.*?\*(.*?)"?>(.*?)<\/a>/;                    # <font...>attribute><a...*link>object</a>
	   push @{$_[1]},{attribute=>$1,link=>$2,object=>$3};                                # attribute   : A > B >...N
	}                                                                                    # link        : http://....
                                                                                         # object      : blah blah...
                                                                                         # I use 'array of hash'
                                                                                         # link        : http://....
```

```
}


sub doSiteInfoExtracting{
        @ObjLinkSiteWrap=${$_[0]}=~/<li>(.*?)<\/li>/gs;
        foreach $aLinkObj(@ObjLinkSiteWrap){
           $aLinkObj=~/.*?\"(.*?)"?>(.*?)<\/a>/;
                                                                    # ...*link>object</a>
                                                                                # link      : http://....
                                                                                # object    : blah blah...
           push @{$_[1]},{link=>$1,object=>$2};                     #   I use 'array of hash'
        }
}


sub AskNextSearchPossible{
        ${$_[0]}=~/.*<\/dl>.*?<center.*?<table.*?<table.*?<a href="(.*?)">(.*?)<\/a>/gs;
        my $rtn_url=$1;


        if ($2=~/next/i){
                    ${$_[1]}=$rtn_url;
                    ${$_[2]}=1;
        }
        else{
                    ${$_[2]}=0;
        }
}
```

```
sub prepareAttr_dir{
        my $dir_attribute=$_[1];
        my $dir_object="$dir_attribute>$_[0]";

        $dir_object=~s/\s*&gt;\s*/>/g;
        $dir_object=~s/'(s|)/ /gi;
        $dir_object=~s/,/_/gi;
        $dir_object=~s/<.*?>//gi;
        $dir_object=~s/\(.*?\)//g;

        my $primitive_attr="$dir_attribute > $dir_object";
        print "<br><b>at first</b> : <font color=red >$primitive_attr</font><br>";
        my $parsed_attr=parsing_attr($primitive_attr);
        print "<b>at last</b>:  <font color=blue >$parsed_attr</font><p>";

        return ($dir_object,$parsed_attr);
}

sub prepareAttr_site{
        my $dir_object=$_[0];
        my $primitive_attr="$dir_object";

        $dir_object=~s/'(s|)/ /gi;
```

```
        $dir_object=~s/\s+/_/gi;
        $dir_object=~s/<.*?>//gi;

        print "<br><b>at first</b> : <font color=red>$primitive_attr</font><br>";
        my $parsed_attr=parsing_attr($primitive_attr);
        print "<b>at last</b> :  <font color=blue>$parsed_attr</font><br>";

        return ($dir_object, $parsed_attr);
}

sub parsing_attr{
        my $string_in=" $_[0] ";

        $string_in =~       s/(<\/?.+?>)|(&gt;)|-//g;                    # delete tags, &gt; and -
        $string_in =~       s/(\w)\.(\s?)/$1$2/gi;

                                                                         # deletion of special character
        $string_in =~       s/(
                                    >
                            |       \.
                            |       \(
                            |       \)
                            |       ,
                            |       \+
```

```
                    |       \!
                    |       \?
                    |       :
                    |       &
                    |       \[
                    |       \]
                    |       @
                    |       \/
                    |       \;
                    |       \%
                    |       \^
                    |       \#
                    |       \$
            )/ /xgi;
```

print "after del of special char : $string_in \n";

deletion of reserved words

```
$string_in =~       s/(
                            (\bby\b)
                    |       (\bfor\b)
                    |       (\bin\b)
                    |       (\bfrom\b)
                    |       (\bshow\b)
                    |       (\bcolumn(s|)\b)
```

```
|          (\breal(s|)\b)
|          (\bon\b)
|          (\boption(s|)\b)
|          (\bset(s|)\b)
|          (\bgroup(s|)\b)
|          (\bkey(s|)\b)
|          (\blong\b)
|          (\bwhere(s|)\b)
|          (\bto\b)
|          (\bcreate(s|)\b)
|          (\balter(s|)\b)
|          (\bdrop(s|)\b)
|          (\bdelete(s|)\b)
|          (\binsert(s|)\b)
|          (\bupdate(s|)\b)
|          (\bselect(s|)\b)
|          (\binto\b)
|          (\border(s|)\b)
|          (\bjoin(s|)\b)
|          (\bleft(s|)\b)
|          (\bcross\b)
|          (\bfull(s|)\b)
|          (\bright(s|)\b)
|          (\blimit(s|)\b)
```

```
                    |       (\bor(s|)\b)
                    |       (\blike(s|)\b)
                    |       (\bas\b)
                    |       (\bfield(s|)\b)
                    |       (\block(s|)\b)
                    |       (\bprocedure(s|)\b)
                    |       (\ball\b)
                    |       (\bfield(s|)\b)
                    |       (\bshow(s|)\b)
                    |       (\bunique(s|)\b)
            )/ /xgi;
```

*print "after del of reserved words : $string_in
\n";*

deletion of general words

```
$string_in =~ s/(
                            ('(s|))
                    |       (\bis\b)
                    |       (\bthe\b)
                    |       (\d+)
                    |       (\bof\b)
                    |       (\bat\b)
                    |       (\band\b)
                    |       (\bregional\b)
                    |       (\bcom+unity\b)
```

```
                    |       (\bcapitolw*\b)
                    |       (\bheight\w*\b)
                    |       (\bconcern\w*\b)
            )/ /xgi;
```

*print "after del of general words : $string_in
\n";*

```
$string_in =~ s/(l|n|d|r|t|c)ies\b/$1y /gi;                                   # ex: studies->study, humanities->humanity
$string_in =~ s/(n|r|t|iew|e|l|m|o|y|k|p|d)s\b/$1 /gi;                        # ex: boys->boy
$string_in =~ s/european\s+patent\s+office/EPO /gi;
$string_in =~ s/(United\s+State\s+Patent\s+Trademark\s+Office)|(US\s+Patent\s+Trademark\s+Office)/USPTO/gi;
$string_in =~ s/US\s+state\b/US /gi;
$string_in =~ s/business\s+shopping/ /gi;
$string_in =~ s/business\s+economy/business/gi;
$string_in =~ s/shopping\s+service/service/gi;
$string_in =~ s/scientific/science/gi;
$string_in =~ s/\bdatabase\b/DB /gi;
$string_in =~ s/\s+/,/g;
$string_in=~tr/[A-Z]/[a-z]/;
$string_in =~ s/^,//;                                                         # elimination of starting ,
$string_in =~ s/,$//;                                                         # elimination of ending ,
```

*print "before foreach : $string_in
\n";*

```
my @attrArr = split(/,/,$string_in);
```

```
        if ((scalar @attrArr) ==1){
           $string_in =~ s/,//gi;
        }
        else{
           foreach $keyInAttr(@attrArr){
                    $string_in =~ s/\b$keyInAttr\b//gi;
                    $string_in =~ s/,,+/,/gi;                    # more than 2 consecutive ,'s are set to only one ,
                    $string_in =~ s/^,//;                        # elimination of starting ,.
                    $string_in =~ s/(|,)$/,/;                    # elimination of ending ,
                    $string_in = "$string_in$keyInAttr";         # deleted keywords are attached at the end
           }
           print "after foreach : $string_in<br>";
        }
        $string_in=~tr/A-Za-z,/_/cs;                             # change non-{ASCII} alphas to single underbar
        $string_in=~s/,\w,/,/g;
        $string_in=~s/(,\w$)|(^(\w|),)//g;                       # delete case aAlpha or a _

        return $string_in;
}
sub FCATbl{
        my $obj=$_[0];
        my $attr=$_[1];
        my $link=$_[2];
        my $kList_hashref=$_[3];
```

```
        my $lList_hashref=$_[4];


#++  this part is for phaseII – concept extracting.          ++++++++++++++++++++++++
        $List=FileRead("obj_attr_List");
        FileWrite("obj_attr_List","$List$obj:$attr\t");
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


        my $val4insert=$attr." ";
        $val4insert =~ s/([^, ]*)(,| )/'Yes'$2/g;
        my $val4modify=$attr." ";
        $val4modify =~ s/([^, ]*)(,| )/$1='Yes'$2/g;
        my @attrArr = split(/,/ , $attr);
        foreach $aKey(@attrArr){
           if(exists $kList_hashref->{$aKey}){
                     my $lookupTime=$kList_hashref->{$aKey};
                     $kList_hashref->{$aKey}=$loopupTime++;
           }
           else{
                     $kList_hashref->{$aKey}=1;
                     $add_attr="alter table fca add column $aKey varchar(4)";
                     $dbh->do($add_attr) or print "<b>Error occured on add</b> : <font
                     face=Courier>".$dbh->errstr()."<br></font>\n>";


           }
```

```
        }
        if(exists $lList_hashref->{$link}){
           my $lookupTime=$lList_hashref->{$link};
           $lList_hashref->{$list}=$loopupTime++;

           $modify_fca="update fca set $val4modify where object='$obj'";
           $dbh->do($modify_fca) or print "<b>Error occured on modify</b> :<font face=Courier> ".$dbh->errstr()."<br></font>\n>";
        }
        else{
           $lList_hashref->{$link}=1;
           $insert_object="insert into fca (object, link, $attr) values ('$obj', '$link', $val4insert)";
           $dbh->do($insert_object) or print "<b>Error occured on insert</b> :<font face=Courier> ".$dbh->errstr()."<br></font>\n>";
        }
}
sub FileRead{
        $FileHandle="FH4Read_$_[0]";
        open ($FileHandle,"$_[0].txt");
           my $result=<$FileHandle>;
           $result=~ s/\n//g;
        close($FileHandle);
        return $result;
}
sub FileWrite{
        $FileHandle="FH4Write_$_[0]";
```

```
        open ($FileHandle,">$_[0].txt");
           print $FileHandle $_[1];
        close($FileHandle);
}
```

APPENDIX B : SOURCE CODE FOR SECTION 5.3 (CONSTRUCTION OF LATTICE)

```
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++                          file name latticeConstructing.pl                          ++++
#++      This program is for extracting all concepts from the FCA context table        ++++
#++            created by 'categoryExtraction.pl' or 'patentDirInfoExtraction.PL'      ++++
#++            I will construct atomic concepts first        and then extend to all concepts.   ++++
#++            All concepts are basically constructed with the aid of   DATABASE manipulation  ++++
#++                  first version : 2001. 1. 22(up : 2001.2.3)                        ++++
#++                  Programmer : Kim, Bong-Seop(U. of Windsor, On, Canada)            ++++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#!c:/perl/bin/perl.exe


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++            1.          HTML Starts (optional part)                                 ++++
#++            I'll show the result in HTML format                                     ++++
#++            (Actually this is a CGI program to which others in remote can access    ++++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
print "Content-type: text/html\n\n";
print   "<html><head><title>Concept Ectracting</title><META HTTP-EQUIV=Content-Language content=KO></head>\n";
print   "<body bgcolor='white'>\n\t<font color=black size=2><br>\n";


#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++            2.          DataBase Connection and setup environment                   ++++
```

```
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
use DBI;
my $dbh = DBI->connect("DBI:mysql:database=thesis","usr","pw",{PrintError => 1, RaiseError => 0});
setup_phase23();

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++    3.     preparation for acquire atomic concepts : We need object list from FCA table    ++++
#++    3.1    we obtain the primitive object(means object not yet in concept) from the FCA table    ++++
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
my @object_Q=get_objectList();                                    #++    3.1

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++  4.  AtomicConcept Retrieval(-> next phase : All concept retrieving from atomic concept)    ++++
#++    4.1    $seqNum : index of concepts                                  ++++
#++           => if we make n atomic concepts, $seqNum will be increased to n    ++++
#++    4.3    we will get 3 results from the subroutine sigma_tau    ++++
#++                  1st   : a concept object set separated by + :   obj1+...+objn    ++++
#++                  2nd   : a concept attribute set separated by , :   attrj1,...,attrn    ++++
#++                  3rd   : additive link info for above object   :   obj1_link+...+objn_link    ++++
#++    4.4    we prevent more than one duplicate concept object    ++++
#++    4.2    by the way of keeping variable 4.2    ++++
#++                  which has accumulated 4.3.1st separated by \t    ++++
#++    4.5    if the concept object set(4.3.1st) is unique,    ++++
#++                  we store the concept info(4.3) and seqNum(4.1) to DB    ++++
```

```
#++                              through the subroutine Add2ConceptDB(4.6)                                    ++++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
my $seqNum = 1;                                                                                    #  4.1
my $tmpObjectList= "\t";                                                                           #  4.2
my $list=FileRead("obj_attr_List");                                                                #  This file was create during phase1
foreach $cur_object(@object_Q){
       my ($flatObjectList,$flatAttributeList,$flatLinkList,$success_code)=sigma_tau($cur_object,\$list);     #  4.3
       if ($success_code eq 'T'){
          $ObjectsWholePattern="$tmpObjectList";
          if ($ObjectsWholePattern !~ /\t$flatObjectList\t/i){                                     #  4.4, 4.5
                    Add2ConceptDB($flatObjectList,$flatAttributeList,$flatLinkList,$seqNum);       #  4.6
                    $tmpObjectList = "$tmpObjectList$flatObjectList\t";
                    print "<font color=blue>concept $seqNum was created succesfully</font><br>\n";
                    $seqNum++;
          }
       }
}
print "<p><font color=red><b>Atomic concept phase complete</b></font><p><p>\n";

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++      5.  Extracting all concepts from the previous atomic concepts!                               ++++
#++          (-> next phase : Lattice constructing - lattice is build at the same time with all concept extracting)                  ++++
#++          5.1          $seqNum : index of concepts(has current concept num to be build)            ++++
#++      * details will be descripted in the sub routine   :                                        ++++
```

```
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
my $lastConcept=Extracting_All_Concepts_And_Building_Lattice($seqNum);          # 5.1
print "Now, under final Processing\n";
postProcess($lastConcept);


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++     6.  HTML ends                                                                   ++++
#++                    This program will result in                                      ++++
#++        (a)         all concepts in form of table(in DB name thesis) like obj1, attr1 ++++
#++        (b)         lattice with concept number, uplink, downlink, attribute set for the concept ++++
#++        And following link will show above all information in HTML format            ++++
#++        So, anybody can see the result from given site(reside in my computer)        ++++
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
print "<a href=/thesisDBmonitor.pl target=_blank><font size=6 > Do you wanna see the result details? -o.o-</font></a>\n";
print "</body></html>";


#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++     7.  DataBase Disconnection(DB connection and disconnection happens only once!(efficient)  ++++
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
$dbh->disconnect();
exit;


#====================          Definition of subroutines          =======================
```

```
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++     3.1 Name of Subroutine :        get_objectList                              ++++
#++                                     input : noting                              ++++
#++                         return : array of all primitive objects                 ++++
#++                         source of data : table FCA in thesis database constructed in phaseI(indexing.pl)   ++++
#++                         remark : each object will be used as input of subroutine sigma_tau   ++++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
sub get_objectList{
	my $sth=$dbh->prepare(qq/select distinct object from fca order by object/);
	$sth->execute();
	my @objList=();
	while(@tuple= $sth->fetchrow_array){
		push @objList, $tuple[0];
	}
	$sth->finish();
	return @objList;
}

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++     4.3 Name of Subroutine :        sigma_tau                                   ++++
#++                                     input : a primitive object from 3.1         ++++
#++                         return :                                  i.e.          ++++
#++                         (a) a concept object set separated by '+'    :  obj1+...+objn   ++++
#++                         (b) a concept attribute set separated by ','  :  attrj1,...attrn   ++++
```

```
#++                    (c) additive link info for above object separated by '+'        :    obj1_link+...+objn_link                    ++++
#++          source of data :                                                                                                          ++++
#++                    (a) table FCA in thesis database constructed in phase1(indexing.pl)                                             ++++
#++                    (b) obj_attr_List.txt conctaining primitive object & attributes set for the object                              ++++
#++                    remark : atomic concepts are constructed by this sub routine                                                    ++++
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
sub sigma_tau{
        my @result = ();                                                    #  initialization
        my @conceptObj = ();                                                #  of array
        my $oa_list=${$_[1]};
        my @attrArr=();
        my $rtn_objs="";
        my $rtn_links="";
        my $success='T';
        my $attribute_condition="";
        $oa_list =~ /\t$_[0]:(.*?)\t/i;                                     #  $list is the form of "$_[0]:attr1,attr2,...,attrn\tobj2: ..."
        my $flat_attributes = $1;                                           #  $flat_attribute is the form of "attr1,attr2,...,attrn"
        $attrNum = @attrArr = split( /,/,$1);
        if($attrNum > 1){
                $attr_condition = $flat_attributes;
                $attr_condition =~ s/,/='Yes' and /g;                       #  we wil convert "attr1,...,attrn"
                                                                            #  =>"attr1=Yes and ...and attrn" for SQL query
          $attr_condition = "$attr_condition='Yes'";                        #  condition(Yes) for the last attribute.  So, we get the final form like
                                                                            #  "attr1=Yes and ... and attrn=Yes"
```

```
}
elsif ($attrNum == 1){
   $attr_condition = "$flat_attributes='Yes'";
}
else{$success='F';}
if ($success eq 'T'){
   $sql=qq/select distinct object, link from fca where $attr_condition order by object/;
   my $sth=$dbh->prepare($sql);
   $sth->execute();
   my @objs=();
   my @links=();
   while(@result= $sth->fetchrow_array){
            push @objs, $result[0];
            push @links, $result[1];
   }
   $rtn_objs = join("+",@objs);                    # we obtain objects list for a concept separated by '+'  i.e.,  obj1+...+objn
   $rtn_links = join("+",@links);                  # we obtain links list separated by '+'                  i.e.,
   obj1_link+...+objn_link
   $sth->finish();
}
else{
   $rtn_obj=""; $flat_attributes=""; $rtn_links="";
}
return ($rtn_objs, $flat_attributes, $rtn_links,$success);
```

```
}

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++     4.6 Name of Subroutine :        Add2ConceptDB                                   ++++
#++                 input : return value from subroutine 4.3 & concept number to be created!   ++++
#++                 return :        noting                                              ++++
#++                 remark :        table for each input(object set & attribute set) will be created   ++++
#++                                 = a object table and a attribute table for atomic concept          ++++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
sub Add2ConceptDB{
        my @ObjectArr=();
        my @AttributeArr=();
        my @linkArr=();
        $objNum = @ObjectArr = split( /\+/,$_[0]);                      # $_[0] is object list separated by +
        print "<font color=blue>concept $_[3] object</font> : $_[0]<br>\n";
        $attrset = $_[1];
        $level = @AttributeArr = split( /,/,$_[1]);                     # $_[1] is attribute list separated by ,
        $link4eachObj = @linkArr = split( /\+/,$_[2]);                  # $_[2] is link list separated by +
        my $conceptNum = $_[3];
        my $tableName = "conceptIndex";                                 # this table contains concept number and the number of attribute which
                                                                        # will be used as level in my coding    *higher level => many attributes.

        $concept_ins="insert into $tableName (conceptNum, level) values ('$conceptNum','$level')";
        $dbh->do($concept_ins) or print "<b>Error occured on add</b> : <font face=Courier>".$dbh->errstr()."<br></font>\n>";
```

```
	$tableName = "obj_$conceptNum";
	$createAobj="create table $tableName (objName varchar(255), link varchar(255))";
	$dbh->do($createAobj) or print "<b>Error occured on create tbl in 4.6</b> : <font face=courier>".$dbh->errstr()."<br></font>\n>";

	for $i(0..$#ObjectArr){
	   $dbh->do(qq/insert into $tableName (objName, link) values ("$ObjectArr[$i]", "$linkArr[$i]")/) or print "<b>Error occured on addObject in 4.6</b> : <font
	face=Courier>".$dbh->errstr()."<br></font>\n>";
	}

	$tableName = "attr_$conceptNum";
	$createAttr="create table $tableName (attrName varchar(255))";
	$dbh->do($createAttr) or print "<b>Error occured on createAttr in 4.6</b> : <font face=Courier>".$dbh->errstr()."<br></font>\n>";

	foreach $Attr(@AttributeArr){
	   $dbh->do("insert into $tableName (attrName) values ('$Attr')") or print "<b>Error occured on addAttr in 4.6</b> : <font face=Courier>".$dbh-
	>errstr()."<br></font>\n>";
	}
	register2lattice($conceptNum, $attrset);			# this concept info will be registered to lattice table!
}

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++	5.1 Name of Subroutine :	Extracting_All_Concepts_And_Building_Lattice		++++
#++				input :		concept number to be created!		++++
#++			return :	noting					++++
```

```
#++                remark :this sub routine will make all other new concepts from the atomic concepts                ++++
#++                        during the job, link between the two concepts will be established                         ++++
#++                Created : 2001. 1. 27                                                                             ++++
#++                Modified :2001. 2. 3                                                                              ++++
#++                        - PART I. complete elimination of duplicate set of attributes                             ++++
#++                        - PART II. complete links between two concepts                                            ++++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
sub Extracting_All_Concepts_And_Building_Lattice{
        $newConceptNum=$_[0];                                                  # initially set to the number of next concept to be added in the future. During progress
                                                                               # of this sub routine, the value will be increased.
        $totalConceptNum=$_[0]-1;                                              # dynamically changing value in outer while
        $theRestConceptNum=$totalConceptNum-1;                                 # dynamically changing value in inner while

        my $sth_index=$dbh->prepare("select conceptNum, level from conceptIndex order by conceptNum");   #++++++++  result sample  ++++++++++#
                                                                               # ------------------------------------
                                                                               # conceptNum  |  level
                                                                               # ------------------------------------
                                                                               #     1       |     3
                                                                               #     2       |     4
                                                                               #     3       |     5
                                                                               #     4       |     7
                                                                               #     ...     |     ...

        $sth_index->execute();
        my @conceptSeq=(0);                                                    # initializatiion of the array containing conceptNum
```

```
my @level=(0);                                                    # initializatiion of the array containing level
while(@tuple = $sth_index->fetchrow_array){
   push @conceptSeq, $tuple[0];
   push @level, $tuple[1];
}
$sth_index->finish();
#++++++++++++++++++++++  start of work list  ++++++++++++++++++++
$base=1;
$compare=2;
while($totalConceptNum){                                          # since new concepts are supposed to be added, loop must be considered on the fact.
                                                                  # totalConceptNum will be chaging dynamically
   while($theRestConceptNum){                                     # inside this inner loop, a concept from outer for loop will be compared
                                                                  # with all concepts from this inner loop.
   #++++++++ decide which object is low level or high level by comparing the number of attributes  +++++
      my ($lower_level,$higher_level_attribute,$lower_level_attribute,$higherConcept,$lowerConcept)
            =decideLowOrHigh($base,$compare,$level[$base],$level[$compare]);
            print "low : $lowerConcept <====> high : $higherConcept\n";
            #++++++++ check whether there exists subconcept or not between the two concepts  +++++++++
            my $sql_conceptComparison=sql_clause_make($higher_level_attribute,$lower_level_attribute);
            my $sth_attr=$dbh->prepare($sql_conceptComparison) or print "Error occured on prepare in 5.1 : ".$dbh->errstr()."\n";
            $sth_attr->execute();
```

```
my $numOfResultAttr = $sth_attr->rows;                                  # Number of attributes from above sql query
if($numOfResultAttr==0){
        #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        #++          <<Case I : Top>>          : Do nothing                 ++
        #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
}
elsif($numOfResultAttr == $lower_level){
        #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        #++          <<CaseII : Sub Concept>>                             ++
        #++          Higher level concept is a subset of lower level concept  ++
        #++          Need to build links between the two concepts which are now being compared ++
        #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        print "<font color=gray>===> case of subConcept : link in the lattice</font><br>\n";
        linkBuildingIntheLattice($higherConcept,$lowerConcept);
}
else{
        #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        #++          <<CaseIII : New concept>>                            ++
        #++          New concept is created and added to worklist also need to register this concept ++
        #++          to lattice table                                     ++
        #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

        #++++++++++++++++   prevent the duplicate creating of the same concepts   ++++++++++++++++++++
        @tmpAttr=();
```

```
while(@resultAttr=$sth_attr->fetchrow_array){     #358 error?
        push @tmpAttr, $resultAttr[0];
}
$sth_attr->finish();
$attrPattern =  join(',',@tmpAttr);

#++          we keep the set of attribute list.  So, if it happens again,  we don't make another copy of  the concept
my $attrlist=FileRead("attrOfConceptList");                                          #++          to prevent duplicate concept

if($attrlist !~ /\t$attrPattern\t/i){
#+++++++++++++++          new concept creating          ++++++++++++++++++++
        $tableName = "attr_$newConceptNum";
        $dbh->do("create table $tableName (attrName varchar(255))")  or print "Error occured on new concept : create table : ".$dbh-
>errstr()."\n";

        foreach
                $dbh->do("insert into $tableName (attrName) values ('$conAttr')");
        }
        $tableName = "obj_$newConceptNum";
        $dbh->do("create table $tableName (objName varchar(255), link varchar(255))") or print "Error occured on new concept : insert(obj) : ".$dbh->errstr()."\n";
        $flat_attrList = join("='Yes' and ",@tmpAttr);
        $dbh->do("insert into $tableName (objName, link) select object, link from fca where $flat_attrList='Yes'") or print "<b>Error occured on
new concept : insert(attr) : </b><font face=Courier>".$dbh->errstr()."<br></font>\n";
        FileCat("attrOfConceptList","$attrPattern\t");
        print "concept $newConceptNum was created succesfully\n";
```

```
                push @conceptSeq, $newConceptNum;
                push @level, $numOfResultAttr;
                $dbh->do("insert into conceptIndex (conceptNum, level) values ('$newConceptNum','$numOfResultAttr')") or print "Error occured on new concept:
conceptIndex : ".$dbh->errstr()."\n";
                register2lattice($newConceptNum, $attrPattern);                    # we register this concept to lattice!

                #++++++++++++++++++          2001.2.3 newly added code for lattice_patch          ++++++++++++++++
                $maxConceptNum=lattice_patch($base,$newConceptNum,\@conceptSeq,\@level);
                if ($maxConceptNum > $newConceptNum ){
                        $newlyAddedConceptNum=$maxConceptNum-$newConceptNum+1;
                        $totalConceptNum=$totalConceptNum+$newlyAddedConceptNum;
                        $theRestConceptNum=$theRestConceptNum+$newlyAddedConceptNum;
                        $newConceptNum=$maxConceptNum+1;
                }
                else{
                        $totalConceptNum++;
                        $theRestConceptNum++;
                        $newConceptNum++;
                }
                #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        }
        else{
                print "=====> case of duplicate concept : no action<font ><br>\n";
```

```
                        }
    #++++++++++++++++++++          considering dynamically changing work list          ++++++++++++++++++++++++
                }
    $compare++;
    $theRestConceptNum--;
    }
  #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  my $sql_final ="select attributes from lattice where conceptNum = $base";
  my $sth_attr_final=$dbh->prepare($sql_final);                                                    #++   modify PART I
  $sth_attr_final->execute();                                                                      #++   2001. 2. 3 newly added!
  @rslt=$sth_attr_final->fetchrow_array;                                                           #++   Newly added module
#   $attrlist="attrlist$rslt[0]\t";                                                                #++   not to make same copy of attributes
  FileCat("attrOfConceptList","$rslt[0]\t");                                                       #++   appearing in the lattice DB
  #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  $base++;
  $totalConceptNum--;
  $theRestConceptNum=$totalConceptNum-1;
  $compare=$base+1;
  }
  #++++++++++++++++++++          the end of work list          ++++++++++++++++++++
  return $newConceptNum;
}

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
#++     4.6.1 & 5.1.1     Name of Subroutine :     register2lattice                                    ++++
#++                       input : concept number to be created & attributes set                        ++++
#++                       return :noting                                                               ++++
#++                Created        : 2001.1.28                                                         ++++
#++                Modyfied : 2001.2.2 ->level attribute add                                           ++++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
sub register2lattice{
        $seqCon=$_[0];                                                          # sequence num of concept
        $_[1]=~tr/A-Z/a-z/;                                                     # change lowercase to upper case
        $lvl=@unsorted_keys=split( /\,/,$_[1]);
        @sorted_keys = sort @unsorted_keys;                                     # sorting
        $flat_sorted_keys=join(',',@sorted_keys);
        FileCat("./updownLink/up_$seqCon","0");
        FileCat("./updownLink/down_$seqCon","-1");
        my $data_ins=qq/insert into lattice (conceptNum, upLink, downLink, attributes, level) values ("$seqCon", '0', '-1', "$flat_sorted_keys","$lvl")/;
        $dbh->do($data_ins) or print "Error occured on data ins in 4.6.1 :".$dbh->errstr()."\n";
}

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#++     5.1.1             Name of Subroutine :     linkBuildingInthel attice                           ++++
#++                       input :        2 concept numbers to be linked                                ++++
#++                       return :       noting                                                        ++++
#++                Created        : 2001.1.28                                                         ++++
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
sub linkBuildingIntheLattice{
        my $hConcept = $_[0];
        my $lConcept = $_[1];
                                                                    # higher concept is a concept which has less attributes
                                                                    # lower concept is a concept which has more attributes
                                                                    #                 downlink
                                                                    # higherConcept ————> lowerConcept
                                                                    # higherConcept <———— lowerConcept
                                                                    #                 uplink
        print "linkBuilding\n";
        FileCat("./updownLink/down_$hConcept",",$lConcept");
        FileCat("./updownLink/up_$lConcept",",$hConcept");
}
sub decideLowOrHigh{
        #++++++++++++++          decide which object is low level or high level by comparing the number of attributes      ++++++++++++++
        my ($lower_level,$higher_level_attribute,$lower_level_attribute,$higherConcept,$lowerConcept);
        my $base=$_[0];
        my $compare=$_[1];
        my $base_level=$_[2];
        my $compare_level=$_[3];

        if($base_level>=$compare_level){
           $lower_level=$compare_level;                             # $lower_level will be used later in constructing lattice for up-down
        link
           $higher_level_attribute = "attr_$base";                  # $higher_level_attribute is the 'attribute table' of concept $i which has
           $lower_level_attribute = "attr_$compare";                # more keys than that of concept $j will be used in the lattice link
```

```
        building
            $higherConcept  = $compare;
            $lowerConcept = $base;
        }
        else{
            $lower_level=$base_level;                                   #
            $higher_level_attribute = "attr_$compare";                  #          opposite case
            $lower_level_attribute = "attr_$base";                      #
            $higherConcept  = $base;                                    # will be use in the lattice link building
            $lowerConcept = $compare;                                   #
        }
        return ($lower_level,$higher_level_attribute,$lower_level_attribute,$higherConcept,$lowerConcept);
}
sub lattice_patch{                                                      # This module patches lattice recursively
        my $notYetComparedMaxConcept=$_[0]-1;
        my $maxConcept2Bcompared=$_[1];
        my $startCon2Bcompared=$_[1];
        my $conceptSeq_arrayref=$_[2];
        my $level_arrayref=$_[3];
        my $loop;
        my $newlyaddedconcept=0;
        print "In lattice patch \n";
        for $loop(1..$notYetComparedMaxConcept){
            my $totalConceptNum2Bcompared=$maxConcept2Bcompared - $startCon2Bcompared+ 1;
```

```
my $curConcept2Bcompared=$startCon2Bcompared;


while($totalConceptNum2Bcompared){                                                    #  code for $loop <-> $curConcept2Bcompared
        $totalConceptNum2Bcompared--;
        my ($lower_level,$higher_level_attribute,$lower_level_attribute,$higherConcept,$lowerConcept)= decideLowOrHigh($loop,$curConcept2Bcompared,$level_arrayref->[$loop],$level_arrayref->[$curConcept2Bcompared]);

        print "lower_level : $lower_level, $higher_level_attribute  <---> $lower_level_attribute\n";
        #++++++++++++++          check whether there exists subconcept or not between the two concepts    ++++++++++++++
        my $sql_conceptComparison=sql_clause_make($higher_level_attribute,$lower_level_attribute);
        my $sth_attr_new=$dbh->prepare($sql_conceptComparison);
        $sth_attr_new->execute();
        my $numOfResultAttr = $sth_attr_new->rows;                                    #  Number of attributes from above sql query
        if($numOfResultAttr==0){
                print  "====> case of no attribute(top) : do nothing</font><br>\n";
        }
        elsif($numOfResultAttr == $lower_level){
                linkBuildingIntheLattice($higherConcept,$lowerConcept);               #  <<CaseII : Sub Concept>>
                print "<font color=green>====> case of subConcept : link in the lattice</font><br>\n";
        }
        else{                                                                         #  <<CaseIII : New concept>>

                #+++++++++++++++        prevent the duplicate creating of the same concepts        ++++++++++++++++++++
                my @tmpAttr=();
                while(@resultAttr=$sth_attr_new->fetchrow_array){
```

```
        push @tmpAttr, $resultAttr[0];
}
$attrPattern = join("\,",@tmpAttr);
my $attrlist=FileRead("attrOfConceptList");                                    # to prevent duplicate concept
if($attrlist !~ /\t$attrPattern\t/i){

        #+++++++++++++++      new concept creating      ++++++++++++++++++++++++++++
        $maxConcept2Bcompared++;
        $totalConceptNum2Bcompared++;
        print "newConcept $maxConcept2Bcompared is created\n";
        $tableName = "attr_$maxConcept2Bcompared";
        $dbh->do("create table $tableName (attrName varchar(255))") or print "<b>Error occured on create attrtbl : ".$dbh->errstr()."\n";
        foreach $conAttr(@tmpAttr){
                $dbh->do("insert into $tableName (attrName) values ('$conAttr')");
        }
        $tableName = "obj_$maxConcept2Bcompared";
        $dbh->do("create table $tableName (objName varchar(255), link varchar(255))") or print "Error occured on create obj tbl : ".$dbh->errstr()."\n";
        $flat_attrList = join("='Yes' and ",@tmpAttr);
        $dbh->do("insert into $tableName (objName, link) select object, link from fca where $flat_attrList='Yes'");

        FileCat("attrOfConceptList","attrPattern\t");
        push @$conceptSeq_arrayref, $maxConcept2Bcompared;
        push @$level_arrayref, $numOfResultAttr;
        $dbh->do("insert into conceptIndex (conceptNum, level) values ('$maxConcept2Bcompared','$numOfResultAttr')");
```

```
                                register2lattice($maxConcept2Bcompared, $attrPattern);                    # we register this concept to lattice!

                                $returnedMaxConNum=lattice_patch($loop,$maxConcept2Bcompared,$conceptSeq_arrayref,$level_arrayref);

                                if ($returnedMaxConNum>$maxConcept2Bcompared){
                                        $newlyaddedconcept=$returnedMaxConNum-$maxConcept2Bcompared;
                                        $totalConceptNum2Bcompared=$totalConceptNum2Bcompared+$newlyaddedconcept;
                                        $maxConcept2Bcompared=$returnedMaxConNum;
                                }
                        }
                        else{
                                print "Duplicate concept creating is discarded\n";
                        }
                }                                                                                         # basic new concept case( including duplicate)
        $curConcept2Bcompared++;
        $sth_attr_new->finish();
        }                                                                                                 # end of while
    print "end of while\n";
    }                                                                                                     # end of for
    return $maxConcept2Bcompared;
}
sub setup_phase23{
    @names = $dbh->tables();
    foreach $aTable(@names){
```

```
        if ($aTable eq 'fca'){
                print "table fca is set well<br>\n";
        }
        elsif($aTable eq 'k_list'){
                print "table k_list is set well<br>\n";
        }
        else{
                $dbh->do("drop table $aTable") or print "<b>Error occured on setup</b> : <font face=Courier>".$dbh->errstr()."<br></font>\n";
        }
    }
    $dbh->do("create table conceptIndex (conceptNum int, level int)") or print "<b>Error occured on create tbl in setup</b> : <font face=Courier>".$dbh->errstr()."<br></font>\n";
    my $create_tbl="create table lattice(conceptNum int, upLink text, downLink text, attributes text,level int)";
    $dbh->do($create_tbl) or print "Error occured on create lattice : ".$dbh->errstr()."\n";
    FileWrite("attrOfConceptList","\t");
}

sub sql_clause_make{
    my $h_level_attr=$_[0];
    my $l_level_attr=$_[1];
        my $where_clause = " where $h_level_attr.attrName =  $l_level_attr.attrName";
        my $orderBy = " order by $h_level_attr.attrName";
        my $from_clause =" from $h_level_attr, $l_level_attr";
        my $rtn_clause= "select $h_level_attr.attrName $from_clause $where_clause $orderBy";
    return $rtn_clause;
```

```
}
sub FileRead{
        $FileHandle="FH4Read_$_[0]";
        open ($FileHandle,"$_[0].txt");
           my $result=<$FileHandle>;
           $result=~ s/\n//g;
        close($FileHandle);
        return $result;
}
sub FileWrite{
        $FileHandle="FH4Write_$_[0]";
        open ($FileHandle,">$_[0].txt");
           print $FileHandle $_[1];
        close($FileHandle);
}
sub FileCat{
        $FileHandle="FH4Write_$_[0]";
        open ($FileHandle,">>$_[0].txt");
           print $FileHandle $_[1];
        close($FileHandle);
}
sub postProcess{
       my $maxCon=$_[0]-1;
       for $i(1..$maxCon){
```

```
my $final_uplinks=FileRead("./updownLink/up_$i");
if($final_uplinks eq '0'){}
else{$final_uplinks=~s/0,//;}

my $final_downlinks=FileRead("./updownLink/down_$i");
if($final_downlinks eq '-1'){}
else{$final_downlinks=~s/-1,//;}

$dbh->do(qq/update lattice set uplink='$final_uplinks', downlink='$final_downlinks' where conceptNum='$i'/);
print "concept $i was updated..\n";
}
```

APPENDIX C : SOURCE CODE FOR SECTION 5.4 (DYNAMIC GRAPHICAL USER INTERFACE

I. Code for outermost Frame (RAIO_index.pl)

```
#!c:/perl/bin/perl.exe

%queryFetch = readQueryString();

$conceptNum=$queryFetch{'concept'};

$attr=$queryFetch{'key'};

$kList=$queryFetch{'curKeySet'};


#++++++++++++++++++++++

print <<HTML;

Content-type: text/html


<html><head><title>Welcome to DAIO system</title></head>

<frameset rows="80,*" border=0 >

  <frame src=summaryBanner.pl  name=summaryFrame  scrolling=no>                                                    # code follows in Appendix C - II

  <frameset cols=260,*>

        <frame src=dynamicKey.pl?key=$attr&concept=$conceptNum&curKeySet=$kList  name=keywordFrame scrolling=yes >    # code follows in Appendix C - III

        <frame src=objectFinding.pl?concept=$conceptNum&curKeySet=$kList  name=ResultFrame scrolling=yes>            # code follows in Appendix C - IV

  </frameset>

</frameset>

</html>
```

```
HTML

#++++++++++++++++++++++

exit;


sub readQueryString{

          my (%searchfield,$buf,$pair,@pairs);

          $buf = $ENV{'QUERY_STRING'};

          @pairs=split(/&/,$buf);


          foreach $pair(@pairs){

                    ($name,$val) = split(/=/,$pair);

                    $searchField{$name}=$val;

          }

          return (%searchField);

}
```

II. Code for summaryFrame (summaryBanner.pl)

```
#!c:/perl/bin/perl.exe


%queryFetch = readQueryString();

my $kList=$queryFetch{'curKeySet'};
```

```
print "CONTENT-TYPE: text/html\n\n";
print "<HTML><HEAD><TITLE>Suumary Banner</TITLE>\n";
print "<script language=javascript>\n";
        print "<!--\n";
        print "function showkeys(){\n";
  print qq@\tvar keyWin=window.open('./showKeys.pl','available_key_list','scrollbars,toolbar=0,width=500,height=400,resizable=0,top=300,left=600,alwaysRaised=1,dependent')\n@;      # code follows in Appendix C - V
        print "}\n";
        print "//-->";
print "</script>";
print "</HEAD><BODY>\n";
        print "<table cellpadding=0 cellspacing=0><tr >";
        print "<td width=260><form name=frm_kword  action=./dynamicKey.pl  method=POST  target=keywordFrame>\n";
        print "\t<input type=text  name=txt_keybox value  size=10  maxlength=255>\n";
        print "\t<input  type=submit  name=sbmt_search  value=Search>\n";
        print qq/<a href="javascript:showkeys()">\n/;
        print "\t<img src=../resource/thesis/key4.gif  border=0 height=25></a>       \n";
        print "</td>\n";
        print "</form>\n";
        print "<td valign=top><center><img src=../resource/thesis/raioSymbol.gif  border=0></center></td>\n";
        print "<td width=260></td>\n";
        print "</tr><tr><td colspan=3 align=right><font size=2>&copy; 2001 Dr.Park YG <i> (Bradley U.)</i>  Kim, BS <i>(U. of Windsor)</td></tr></table>";
print "</BODY></HTML>\n";

exit;
```

III. Code for keywordFrame (dynamicKey.pl)

```
#!c:/perl/bin/perl.exe
print <<HTML_I;
CONTENT-TYPE: text/html



<HTML><HEAD><TITLE>List of all availabel keys</TITLE>
</HEAD><BODY>
HTML_I


use DBI;
$dbh_key=DBI->connect("DBI:mysql:database=thesis","root","");


if ($ENV{'REQUEST_METHOD'} eq 'POST'){
        %postInputs = readFormInput();
        my $searchKeys=$postInputs{'txt_keybox'};
        if($searchKeys eq ''){                                                          #  for users' mistakes
                print "<font size=2 color=red>No blank!<br> You can refer available key set by clicking key image on the right side....^^<p>\n";
        }
        else{
                $searchKeys=~tr/A-Z/a-z/;                                               #  change uppercase to lowercase
                $searchKeys=~s/\s+/,/g;                                                 #  change space to ,
                $searchKeys=~s/,+/,/g;                                                  #  more than 2 commas changed to one comma
```

```
		$searchKeys=~s/,/%' and attributes like '%/g;                                   # key1,key2 -> key1%' and attributes like '%key2
		my $sql_search=qq/select distinct conceptNum,attributes from lattice where (attributes like '%$searchKeys%') order by level\n/;
		my $sth_lattice=$dbh_key->prepare($sql_search);
		$sth_lattice->execute();
		my $rsltNum=$sth_lattice->rows;
		if($rsltNum == 0){                                                          # CASE : no match
			print "Such key from you is  unavailable in this system";
		}
		else{
			print "Are you looking for this combination of keys including yours?<br>\n";
			print "total $rsltNum combinations were found.<hr>\n";
			while(my @tuple = $sth_lattice->fetchrow_array){
				print qq@<font size=2><a href=RAIO_index.pl?key=$tuple[1]&concept=$tuple[0]&curKeySet=$tuple[1] target=_top>$tuple[1]</a></font><br>\n@;
			}
		}
	}
}
else{                                                                               # method GET
	%queryFetch = readQueryString();
	my $conceptNum=$queryFetch{'concept'};
	my $curKeyList=$queryFetch{'curKeySet'};
	if ($conceptNum==""){
		$sql_key_top="select distinct attributes, conceptNum from lattice where uplink=0 order by attributes";
		$sth_key_top=$dbh_key->prepare($sql_key_top);
```

```
                $sth_key_top->execute();
                while(@rtn_keyFromTop=$sth_key_top->fetchrow_array){
                        print qq@<font size=2><a href=RAIO_index.pl?key=$rtn_keyFromTop[0]&concept=$rtn_keyFromTop[1]&curKeySet=$rtn_keyFromTop[0] target=_top>$rtn_keyFromTop[0]</a></font><br>\n@;
                }
        }
        else{
                $sql_UpDown=qq/select upLink, downLink from lattice where conceptNum = $conceptNum\n/;
                $sth_UpDown=$dbh_key->prepare($sql_UpDown);
                $sth_UpDown->execute();
                @rtn_UpDown=$sth_UpDown->fetchrow_array;
                parseConceptList($curKeyList,$rtn_UpDown[1],"zoomIn");
                print "\n\n";
                parseConceptList($curKeyList,$rtn_UpDown[0],"zoomOut");
        }
}
print "</BODY></HTML>\n";
$dbh_key->disconnect();
exit;

sub readFormInput{
  my (%searchfield,$buf,$pair,@pairs);
  read(STDIN,$buf,$ENV{'CONTENT_LENGTH'});
  @pairs=split(/&/,$buf);
  foreach $pair(@pairs){
```

```
        ($name,$val) = split(/=/,$pair);
        $val=~ tr/+/ /;
        $val=~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex($1))/eg;
        $name=~ tr/+/ /;
        $name=~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C",hex($1))/eg;
        $searchField{$name}=$val;
    }
    return (%searchField);
}
sub parseConceptList{
        my $curKeys=$_[0];
        my $listOfConcepts=$_[1];                    #1,173,496
        my $zoomInOut=$_[2];
        if ($zoomInOut eq 'zoomIn'){
                print "<img src=../resource/thesis/zoomIn.jpg border=0> Narrow  your search scope!<br>\n";
                print "<ul>";
                if ($listOfConcepts==-1){
                        print "<li>No available keys to zoom in<br>\n";
                }
        }
        else{
                print "<p><img src=../resource/thesis/zoomOut.jpg border=0> Zoom out to following  scope!<br>\n";
                print "<ul>";
                if ($listOfConcepts==0){
```

```
                print "<li>No available keys to zoom out<br>\n";
        }
}
$listOfConcepts=~s/,/' or conceptNum='/g;        #1' or conceptNum='173' or conceptNum='496
my $sql_orderedConcept="select attributes,conceptNum,level from lattice where (conceptNum='$listOfConcepts') order by level,attributes";
my $sth_orderedConcept=$dbh_key->prepare($sql_orderedConcept);
$sth_orderedConcept->execute();
my $curNum=0;
my $curAttr="";
my @result_orderConcept="";
while(@result_orderedConcept=$sth_orderedConcept->fetchrow_array){
        my ($listOfCurrentKeys, $listOfNextKeys, $numKeys)=makeCurrentKeyList($curKeys,$result_orderedConcept[0], $zoomInOut);
        if ($curNum != $numKeys && $zoomInOut eq 'zoomIn'){
                $curNum=$numKeys;
                print "<p><font color=gray>add $curNum more key(s)</font><br>";
        }
        if($curAttr ne $listOfNextKeys){
                $curAttr=$listOfNextKeys;
                print qq@<li><font size=2><a href=RAIO_index.pl?key=$result_orderedConcept[0]&concept=$result_orderedConcept[1]&curKeySet=$listOfNextKeys target=_top>$listOfCurrentKeys</a></font><br>\n@;
        }
}
print "</ul><p><hr width=80%><p>\n";
}
```

```
sub makeCurrentKeyList{
        my $curKeys=$_[0];
        my $newKeys=$_[1];
        my $operation=$_[2];
        my $finalNextKey="";
        if ($operation eq 'zoomIn'){
                $curKeys=~s/~/,/g;
                @allKeys=split(/\,/,$curKeys);
                foreach $aKey(@allKeys){
                        $newKeys=~s/(^$aKey,|,$aKey$)//;
                        $newKeys=~s/,$aKey,/,/;
                }
                if ($curKeys ne ""){
                        $finalNextKey="$_[0]~$newKeys";
                }
                else{
                        $finalNextKey=$newKeys;
                }
        }
        else{
                $finalNextKey=$_[1];
                $newKeys=$_[1];
        }
        my $numOfKeys=@keyArr=split (/,/,$newKeys);
```

```
        return ($newKeys, $finalNextKey, $numOfKeys);
}
```

IV. Code for resultFrame(objectFinding.pl)

```
#!c:/perl/bin/perl.exe
use DBI;
my $dbh_objectFinding=DBI->connect("DBI:mysql:database=thesis","root","");
print "CONTENT-TYPE: text/html\n\n";
print "<HTML><HEAD><TITLE>RAIO -- Results found</TITLE>\n";
print "</HEAD><BODY><font size=2>\n";

%queryFetch = readQueryString();
my $conceptNum=$queryFetch{'concept'};
my $kList=$queryFetch{'curKeySet'};
if ($conceptNum eq ''){
        print "<img src=../resource/thesis/overviewRAIO.jpg border=0>\n";
}
else{
        $kList=~s/~/ -> /g;
        $sql_search=qq/select distinct conceptNum, upLink, downLink, attributes from lattice where conceptNum = $conceptNum\n/;
        my$ sth_lattice= $dbh_objectFinding->prepare($sql_search);
        $sth_lattice->execute();
```

```
        my @tuple = $sth_lattice->fetchrow_array;
        show_match_case($tuple[0],$tuple[1],$tuple[2],$tuple[3]);                    # subroutine call
}
print "</font></body></html>";
$dbh_objectFinding->disconnect();
exit;



sub show_match_case{
        my $conNum=$_[0];
        my $zoomOut=$_[1];
        my $zoomIn=$_[2];
        my $searchWords=$_[3];
        my $sql_match_obj="select distinct objName, link from obj_$conNum";
        my $sth_match= $dbh_objectFinding->prepare($sql_match_obj);
        $sth_match->execute();
        my $rsltNum=$sth_match->rows;
        print "<table width=800 border=0><tr><td bgcolor=#0000CD><font color= white>Your current keyword-set tracing  : </font><font color=yellow><b>$kl.ist</b></font></td></tr>\n";
        print "<tr><td>>>> Total matchs : $rsltNum</td></tr></table><p>\n";
        while(@final_objs=$sth_match->fetchrow_array){
                print "<a href=$final_objs[1] target=ResultFrame><font size=2>$final_objs[0]</font></a><br>";
        }
        print "<p><hr width=80% align=left><p>";
        print "<hr width=50% align=left>";
```

```
}
```

V. Code for keyWin(showKeys.pl)

```
#!c:/perl/bin/perl.exe

print <<HTML_javascript;

CONTENT-TYPE: text/html

<HTML><HEAD><TITLE>List of all availabel keys</TITLE>

<script language=javascript>

<!--

  function attachToSearchField(argText){

          var curSearchVal=opener.frm_kword.txt_keybox.value;

          if (curSearchVal == "enter,your,search,keys" || curSearchVal==""){

                  curSearchVal=argText;

          }

          else{

                  curSearchVal=curSearchVal+","+argText;

          }

          opener.frm_kword.txt_keybox.value=curSearchVal;

  }

//--></script>

</HEAD><BODY>

HTML_javascript
```

```
use DBI;
$dbh_thesis=DBI->connect("DBI:mysql:database=thesis","bongeun","bong0413");
@alpha_index=(a..z);
print "<a name=top></a>\n";
print "<center>";
foreach $idx(@alpha_index){
          print "<font size=2> <a href=#$idx>$idx</a> </font>";
}
print "</center>";
print "<p>\n";
foreach $idx(@alpha_index){
          print "<A NAME=$idx></A>\n";
          print "<table width=100% border=0 cellpadding=0 cellspacing=0><tr>";
          print "<td width=25%> </td>\n";
          print "<td colspan=2><center><font size=5 color=gray><b>$idx</b></font></center></td>\n";
          print "<td width=25% align=right><A Href=#top><img src=../resource/thesis/up_2.gif border=0 height=12 alt='top'></a></td>\n";
          print "</tr>\n";
          print "<tr>";
          print "<td width=25%><hr width=40% align=right></td><td width=25%><hr width=100%></td><td width=25%><hr width=100%></td><td width=100%><hr width=40% align=left></td></tr>\n<tr>";
          $sql_by_idx="select distinct itm from k_list where itm like '$idx%' order by itm";
          $sth_key=$dbh_thesis->prepare($sql_by_idx);
          $sth_key->execute();
          $colNum=0;
```

```
        while(@rtn_keys=$sth_key->fetchrow_array){
                if ($colNum%4 ==0){
                        print "</tr>\n<tr>";
                        print qq@<td width=25%><font size=2><a href=javascript:attachToSearchField("$rtn_keys[0]")>$rtn_keys[0]</a></font></td>\n@;
                }
                else{
                        print qq@<td width=25%><font size=2><a href=javascript:attachToSearchField("$rtn_keys[0]")>$rtn_keys[0]</a></font></td>\n@;
                }
                $colNum++;
        }
        $theRest=$colNum%4;                                                    # 4 is number of columns
        while($theRest>0){
                print "<td width=25%></td>";
                $theRest--;
        }
        print "</tr><tr><td colspan=4> <br></td></tr>\n";
        print "<tr><td colspan=4>  <br></td></tr>\n";
        print "</table>";
}
print "</BODY></HTML>\n";
exit;
```

[redacted]:

If you need the file for the source code, ask for me.

E-mail: bongeun@chollian.net
HomePage: http://user.chollian.net/~bongeun

APPENDIX D : SETTING THE WEBSERVER 'APACHE'

1. Configuring Apache web server for my thesis environment.

1.1 My environment requirement

For the implementation of this thesis, I will use Apache as web server mostly hosting a cgi program in perl code.

All my cgi program have 'pl' as their extension. Ex) myPerlProgram.pl

My system is connected to internet with dynamic ip address.

Every time my system connect to internet, my system is assigned different address.

I manage many directories for web hosting and I don't want use given, default document root directory as my actual document root.

'C:/Program Files/Apache Group/Apache/htdocs' is given document root. But, I would like to use 'C:/httpd/htdocs' as my root directory.

Clients of my system usually speak in English or Korean

1.2 Modifying 'httpd.conf' file for my system requirement.

When I start the Apache web server daemon, the server refers 'httpd.conf' file first and then load daemon following the settings in the conf file. Via httpd.conf file, I can control Apache server. Hence, httpd.conf could be thought as a handle for web server.

1.2.1 Let Apache know my perl program

At first, httpd.conf blocks(#) 'AddHandler' which allows cgi script(perl program) to be recognized by server daemon and to be executed.

```
#AddHandler cgi-script .cgi              <before>
AddHandler cgi-script .cgi .pl           <after>
```

More over, I have lots of directories containing perl programs which are waiting to be executed. I need let Apache know the directories. It can be possible by following way.

```
scriptAlias /thesis/ "C:/httpd/thesis/"
<Directory "C:/httpd/thesis">
  AllowOverride None
  Options ExecCGI
</Directory>
```

* just directory aliasing can be done as following way.

```
Alias /study/ "C:/httpd/study"
```

1.2.2 Let Apache know its location in the internet

Currently, I am using ADSL by Bell which assign a new ip address for each time I connect to the Bell server. Such address is called dynamic ip address. So every time I connect to the Bell server, I need to let my Apache web server know its location in the internet. That could be possible changing ServerName as following:

```
ServerName 64.230.128.66
```

1.2.3 Let Apache know my own document root not default root.

```
DocumentRoot "C:/Program Files/Apache Group/Apache/htdocs"   <before>
DocumentRoot "C:/httpd/htdocs"                               <after>
```

1.2.4 Let Apache know my preferred languages.

```
AddLanguage ko .kr      where ko is for language & kr for country
LanguagePriority kr en da nl et fr de el it pt ltz ca es sv
```

APPENDIX E : OVERVIEW OF THE DBMS 'MYSQL'

1. What is MySql?

- MySql is a RDBMS and said to be very stable because it has proven itself over 10 years.

- MySql is a multithreaded server. Multithreaded means that every time someone established a connection with the server, the server program creates a thread or process to handle that client's requests. This makes for an extremely fast server.

- MySql is fully ANSI SQL 92 compliant
- MySql is free.

program location

MySql is free software that we can download the latest version from following site and then install.

```
http://www.mysql.com/
```

Interfaces of MySql

MySql provides us with a number of programming interfaces including interfaces for Perl(DBI/DBD). These interfaces or APIs allow us to build applications that use MySql.

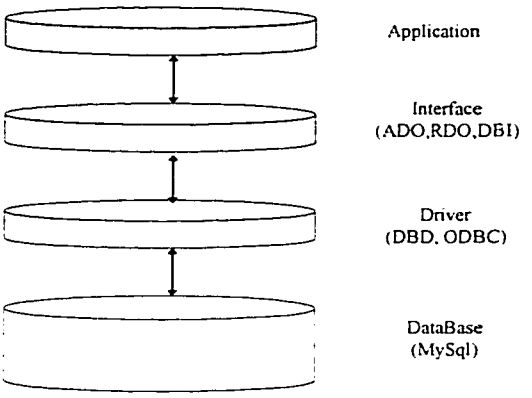


Fig. 28 MySql and its Interfaces

2. MySql DBI

DBI can be simply said to be perl database interface. And can be installed with the aid of the tool called PPM supported by Activestate Perl(see Appendix F).

Invoking DBI module in perl programs.

Because DBI uses perl's object orientation features, just adding one line in a perl program let the programmer use the functions of the module DBI.

```
use DBI;
```

3. MySql DBD

DDB is a DataBase Driver as a module that does most of real work between a specific database(i.e., MySql) and DBI of a application. DBD is database dependent in comparison with DBI which is database independent. As the result, while there is only one DBI, there are many DBDs for corresponding databases.

Currently available database drivers for perl DBI are :

DBD::*CSV*(general comma separated value ASCII file), DBD::*DB2*, DBD::*Empress*, DBD::*Informix*, DBD::*Ingress*, DBD::*InterBase*, **DBD::*mysql***, DBD::*ODBC*, DBD::*Oracle*, DBD::*Pg*(PostgresSQL), DBD::*SearchSearver* (Fulcrum Search Searver), DBD::*Sybase* (for Sybase and MS SQL Server), DBD::*Xbase*(for Xbase files i.e., dBase)

Among above drivers, my concern is DBD::mysql.

DBD:mysql also can be installed as the same way with DBI.(see Appendix F).

Following code is a commonly used example in coding :

```
$dbh=DBI->connect("DBI:mysql:database=thesis", "id", "pw");
```

DBD — database — user — password

APPENDIX F : ACTIVESTATE PERL OVERVIEW

1. What is Active Perl?

The ActivePerl is Package for Perl platforms merging the popular Perl ports. ActivePerl contains **Perl**, the **Perl Package Manager** which is called PPM(for installing packages of CPAN modules), and complete online help. ActivePerl is available for Linux, Solaris and Windows Operating Systems.

This tool is also free and can be obtained at :

http://www.perl.com

2. Perl

Perl is a very high-level programming language originally developed in the 1980s. Perl is now being developed by a group of individuals known as the Perl5-Porters.

One of Perl's many stengths is its ability to process arbitrary chunks of textural data, known as strings, in many powerful ways, including regular expression string manipulation.

This capabilty makes perl an excellent choice or database programming, since the majority of information stored within databases is textual in nature.

Perl program tends to be far smaller than equivalent C programs and are generally portable to other operating systems that run perl with little or no modification.

Perl also features the ability to dynamically load external modules, which are pieces of

software that can be slotted into perl to extend and enhance its functionality . There are hundreds of these modules available now ranging from mathematical modules to three-dimensional graphics-rendering modules, to modules that allows us to interact with networks and network software.

Perl has been accepted as being the language on the WWW for writing CGI programs.

3. An example of installing a module using PPM

Before running PPM, the system should be connected to the Internet. PPM can then be started by typing *'ppm'* in a command prompt window

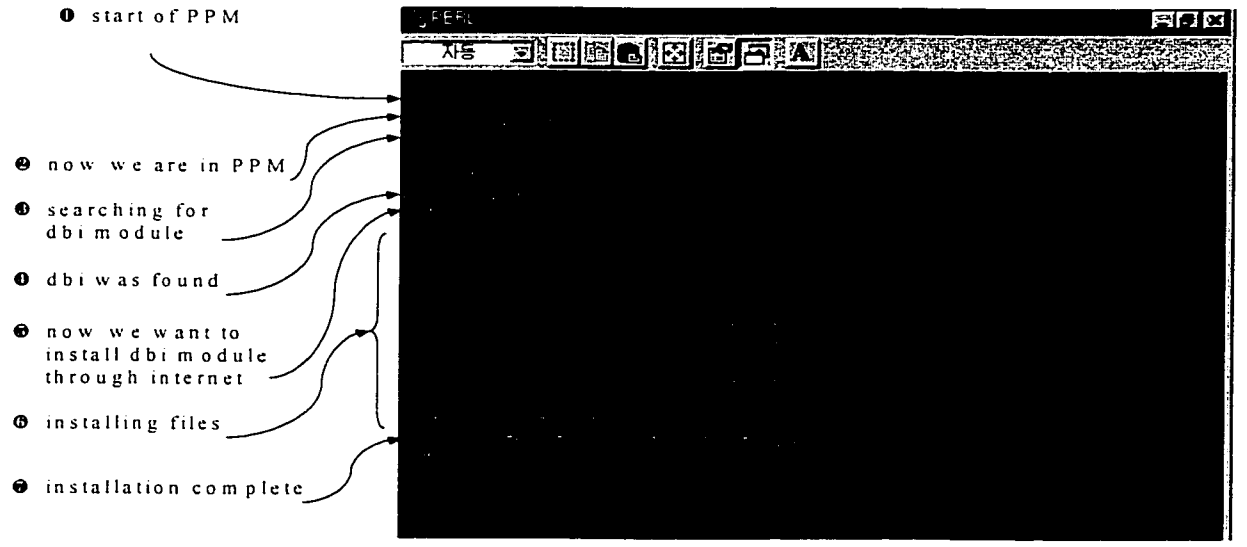


Fig. 29 An example of installing the DBI module using PPM

APPENDIX G : THE NETWORK PROGRAMMING SUPPORT IN PERL (LWP)

Internet Architecture is represented like above diagram. In perl, network application means usually the programs on the TCP Layer. Perl has lots of modules or libraries to support constructing the network applications.

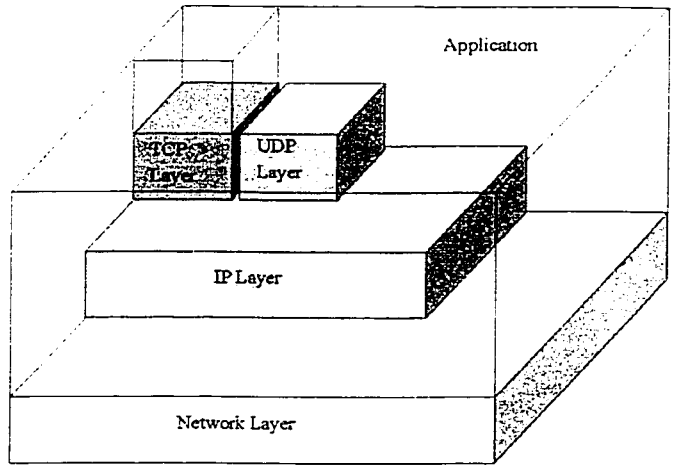


Fig. 30 The Internet Architecture

1. Modules supported in Perl for Network Programming

In writing network programming in Perl, there are two approaches. The one is using module Socket supported in the tcp layer. With the aid of the module, we can establish a connection manually and then build codes on the established tcp connection. This job is usually tedious and results in messy codes. The other approach is using application level module for example POP3, SMTP, NNTP and so on.

The codes build in the application level could be divided into several part. Perl has appropriate module for the divided part. That is to say, for the E-mail applications, perl has

two modules. The one is 'POP3' that supports Post Office protocol for reading emails and the other is 'SMTP' that supports Simple Mail Transfer Protocol for sending emails. For news applications, module NNTP is supported. And for the FTP applications, module FTP is supported. But most of NW program is related with WEB applications using HTTP and Perl has more detailed and delicate set of modules for such kind of applications.

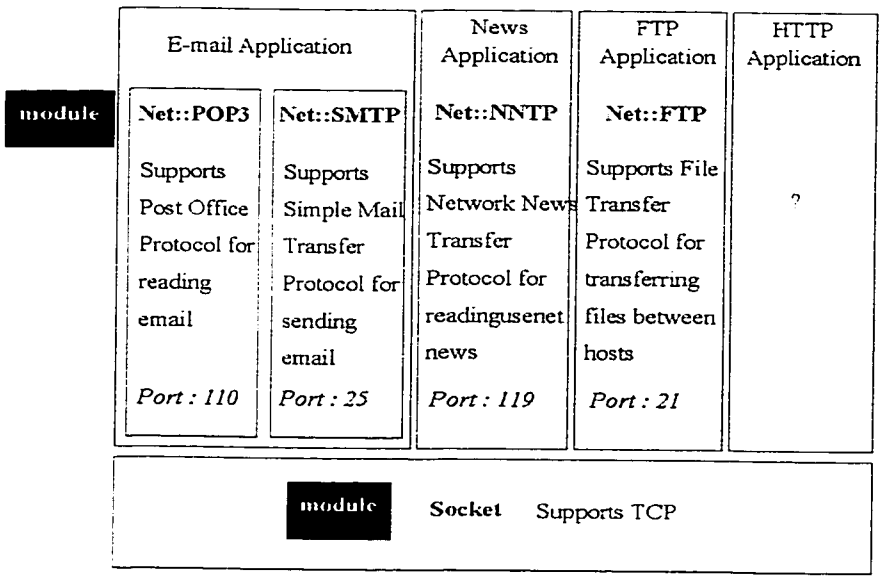


Fig. 31 Modules supported in Perl for Network Programming

2. LWP Library for HTTP Application

Library LWP is the set of modules, for web programming (HTTP Application) in Perl. With the use of the library, we can construct consistent and neat codes for the application. The library contains the following modules(Bold faced modules are more useful than others) and can be downloaded as the single file named 'libwww-perl' in CPAN site (http : // www.perl.com/CPAN)

File	Parses directory listings.
Font	Handles Adobe Font Metrics.
HTML	Parses HTML files to HTML syntax trees and converts them to printable or other formats
HTTP	Provides client requests, server responses, and computes a client/server negotiation
LWP	The core of all web client programs. It allows the client to communicate over the network with server
URI	A relative URL is converted to a absolute URL.
WWW	Implements standards used for robots (automatic client programs).

2.1 The HTML Module

There are "eleven" classes in the HTML module. Among them, bold faced modules are used more often than others. With the HTML module we can do following things :

- parse the HTML into an HTML syntax tree
- extracting links
- converting the HTML into text or PostScript

2.2 The LWP Module

LWP module in the library LWP has 10 classes and perfoms performs client requests over the network. The LWP modules provide the core of functionality for web programming in Perl. It contains the foundations for networking applications, protocol implementations, media type definitions, and debugging ability.

The modules LWP::Simple and LWP::UserAgent define client applications that implement network connections, send requests, and receive response data from servers. LWP::RobotUA is another client application that is used to build automated web searchers following a specified set of guidelines.

LWP::UserAgent is the primary module used in applications built with LWP. With it, we can build our own robust web client. It is also the base class for the Simple and RobotUA modules. These two modules provide a specialized set of functions for creating clients.

Additional LWP modules provide the building blocks required for web communications, but we often don't need to use them directly in our applications. LWP::Protocol implements the actual socket connections with the appropriate protocol. The most common protocol is HTTP, but mail protocols (like SMTP), FTP for file transfers, and others can be used across networks.

LWP::MediaTypes implements the MIME definitions for media type identification and mapping to file extensions. The LWP::Debug module provides functions to help us debug our LWP applications.

2.3 The HTTP Module

HTTP module has 8 classes and mainly specifies HTTP requests and responses. The HTTP modules implement an interface to the HTTP messaging protocol used in web transactions. Its most useful modules are HTTP::Request and HTTP::Response, which create objects for client requests and server responses. Other modules provide means for manipulating headers, interpreting server response codes, managing cookies, converting date formats, and creating basic server applications.

Client applications created with LWP::UserAgent use HTTP::Request objects to create and send requests to servers. The information returned from a server is saved as an HTTP::Response object. Both of these objects are subclasses of HTTP::Message, which provides general methods of creating and modifying HTTP messages. The header information included in HTTP messages can be represented by objects of the

HTTP::Headers class.

HTTP::Status includes functions to classify response codes into the categories of informational, successful, redirection, error, client error, or server error. It also exports symbolic aliases of HTTP response codes; one could refer to the status code of 200 as RC_OK and refer to 404 as RC_NOT_FOUND.

The HTTP::Date module converts date strings from and to machine time. The HTTP::Daemon module can be used to create webserver applications, utilizing the functionality of the rest of the LWP modules to communicate with clients.

2.4 The URI Module

The URI module contains functions and modules to specify and convert URLs. The URI module contains functions and modules to specify and convert URIs. (URLs are a type of URI.) There are three URI modules: URL, Escape, and Heuristic. Of primary importance to many LWP applications is the URI::URL class, which creates the objects used by LWP::UserAgent to determine protocols, server locations, and resource names.

The URI::Escape module replaces unsafe characters in URL strings with their appropriate escape sequences. URI::Heuristic provides convenience methods for creating proper URLs out of short strings and incomplete addresses.

VITA AUTORIS

NAME : KIM, BONG-SEOP

PLACE OF BIRTH : BUSAN, SOUTH KOREA

YEAR OF BIRTH : 1969

EDUCATION : HAEWOONDAE HIGH SCHOOL, PUSAN, SOUTH KOREA

1985-1987

YONSEI UNIVERSITY, SEOUL, KOREA

1988-1991 B.SC.

UNIVERSITY OF WINDSOR, WINDSOR, ONTARIO, CANADA

1999-2001 M.SC.