Challenging the Invisible Web:
Improving Web Meta-Search by Combining Constraint-based Query Translation and Adaptive User Interface Construction

Vom Fachbereich Informatik der Technischen Universität Darmstadt genehmigte

Dissertation

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

vom

Master of Engineering (Computer Software)

Lieming Huang

Aus Guangdong, V. R. China

Referent: Prof. Dr. Erich J. Neuhold

Koreferent: Prof. Dr. Alejandro P. Buchmann

Tag der Einreichung: 04.07.2003

Tag der mündlichen Prüfung: 26.09.2003

Darmstadt 2003
D 17

Darmstädter Dissertation

Challenging the Invisible Web:
Improving Web Meta-Search by Combining Constraint-based Query Translation and Adaptive User Interface Construction

A DISSERTATION SUBMITTED TO
THE DEPARTMENT OF COMPUTER SCIENCE OF
TECHNISCHE UNIVERSITÄT DARMSTADT
FOR THE DEGREE OF DOCTOR OF ENGINEERING

by

Master of Engineering (Computer Software)

Lieming Huang

From Guangdong, P. R. China

Reviewers:
Prof. Dr. Erich J. Neuhold
Prof. Dr. Alejandro P. Buchmann

Date of thesis submission: 04.07.2003
Date of Viva Voce: 26.09.2003

2003
Technische Universität Darmstadt
Darmstadt, Germany

Abstract

The revolution of the World Wide Web (WWW or Web) has set off the globalization of information publishing and access. Organizations, enterprises, and individuals produce and update data on the Web everyday. With the explosive growth of information on the WWW, it becomes more and more difficult for users to accurately find and completely retrieve what they want. Although there are hundreds of thousands of general-purpose and special-purpose search engines and search tools, most users still find it hard to retrieve information precisely. Moreover, considering the great amount of valuable information hidden in the Invisible Web that is generally inaccessible to traditional "crawlers", providing users with an effective and efficient tool for Web searching is necessary and urgent.

First, this dissertation proposes an adaptive data model for meta-search engines (ADMIRE) that can be used to formally and meticulously describe the user interfaces and query capabilities of heterogeneous search engines on the Internet. Compared with related work, this model focuses more on the constraints between the terms, term modifiers, attribute order, and the impact of logical operators.

Second, this dissertation presents a constraint-based query translation algorithm. When translating a query from a meta-search engine to a remote source, the mediator considers the function and position restrictions of terms, term modifiers and logical operators among the controls in the user interfaces to the underlying sources sufficiently, thus allowing the meta-search engine to utilize the query capabilities of the specific sources as far as possible. In addition, a two-phase query subsuming mechanism is put forward to compensate for the functional discrepancies between sources, in order to make a more accurate query translation.

Furthermore, this dissertation presents a mechanism for constructing adaptive, dynamically generated user interfaces for meta-search engines based on the above-

mentioned model. The concept of control constraint rules has been proposed and applied to the user interface construction. Depending on the state of interaction between users and system, such meta-search engines adapt their interfaces to the concrete user interfaces of differing kinds of search engines (Boolean model with differing syntax, vector-space/probabilistic model, natural language support, etc.), so as to overcome the constraints of heterogeneous search engines and utilize the functionality of the individual search engines as much as possible.

Finally, this dissertation also tackles some issues on wrapper generation and result merging for Web information sources. The experiments show that an information integration system with an adaptive, dynamically generated user interface, coordinating the constraints among the heterogeneous sources, will greatly improve the effectiveness of integrated information searching, and will utilize the query capabilities of sources as far as possible. The adaptive meta-search engine architecture proposed in this dissertation has been applied to the information integration of scientific publications-oriented search engines. It can also be applied to other generic domains or specific domains of information integration, such as integrating all kinds of WWW search engines (or search tools) and online repositories with quite different user interfaces and query models. With the help of source wrapping tools, they can also be used to integrate queryable information sources delivering semi-structured or non-structured data, such as product catalogues, weather reports, software directories, and so on.

Zusammenfassung

Die Revolution des World Wide Web (WWW oder Web) hat weltweit die Informationspublikationen und den Zugang zu Informationen auf den Weg gebracht. Organisationen, Unternehmen und Einzelpersonen produzieren und aktualisieren täglich Informationen im Web. Angesichts der explosiven Zunahme der Informationen im Web wird es für die Benutzer immer schwieriger, exakt die Information zu finden, die sie suchen. Obgleich Hunderttausende von universellen und spezifischen Suchmaschinen und Suchwerkzeugen existieren, fällt es den meisten Benutzern noch immer schwer, gezielt Informationen zu gewinnen. Angesichts der großen Menge wertvoller Informationen, die im verborgenen Web verstecken, und daher im Allgemeinen für die traditionellen "Crawler" unzugänglich sind, ist es unerlässlich, dem Benutzer ein wirkungsvolles und leistungsfähiges Werkzeug für die Suche im Web an die Hand zu geben.

Zuerst wird in dieser Dissertation ein adaptives Datenmodell für Meta-Suchmaschinen (ADMIRE) vorgestellt, das verwendet wird, um die Benutzerschnittstellen und Anfragefähigkeiten von heterogenen Suchmaschinen im Internet formal und ausführlich zu beschreiben. Im Vergleich mit verwandten Arbeiten liegt der Schwerpunkt dieses Modells auf den Constraints von bzw. zwischen Termen, Termmodifiziereren und Attributanordnungen, sowie auf dem Einfluss logischer Operatoren.

Zweitens wird ein constraint-basierter Algorithmus zur Anfrageübersetzung in dieser Dissertation vorgestellt. Bei der Übertragung einer Anfrage von einer Meta-Suchmaschine auf eine entfernte Quelle berücksichtigt der Mediator die funktionellen Beschränkungen, die zwischen den Termen, Termmodiziereren und logischen Operatoren der Steuermechanismen der Benutzerschnittstellen und den zugrundeliegenden Quellen bestehen, d.h. die Meta-Suchmaschine kann die Anfragefähigkeiten der spezifischen Quellen weitestgehend ausnutzen. Zusätzlich

wird ein zweiphasiger Zuordnungsmechanismus eingesetzt, der die funktionellen Unterschiede zwischen den Quellen ausgleichen und die Anfrageübersetzung präzisieren soll.

Darüber hinaus wird von dieser Dissertation ein Konstruktionsmechanismus für adaptive, dynamisch generierte Benutzerschnittstellen der Meta-Suchmaschinen vorgestellt, die auf dem oben erwähnten Modell basieren. Zum Aufbau der Benutzerschnittstelle wurde das Konzept von Constraints-Regeln der Steuerung angewandt. Abhängig vom Zustand der Interaktion zwischen Benutzern und System passen diese Meta-Suchmaschinen ihre Schnittstellen den konkreten Benutzerschnittstellen der unterschiedlichen Suchmaschinen an (Boolsches Modell mit unterschiedlicher Syntax, Vektor-Raum/probabilistisches Modell, Unterstützung natürlicher Sprache usw.), um die Constraints der heterogenen Suchmaschinen zu überwinden, und weitestgehend die Funktionalität der jeweiligen Suchmaschinen auszunutzen.

Zuletzt diskutiert die Dissertation einige Implementierungaspekte zur Wrapper-Erzeugung und Zusammenstellung der Ergebnisse für Web-Informationsquellen. Die Tests zeigen, dass ein Informationsintegrations-System mit adaptiver, dynamisch generierter Benutzerschnittstelle, die die Constraints zwischen heterogenen Quellen koordiniert, die Wirksamkeit der integrierten Informationssuche erhöht und die Anfragefähigkeit der Quellen weitestgehend nutzt. Die in dieser Dissertation vorgestellte adaptive Architektur der Meta-Suchmaschine wurde zur Informationsintegration von Suchmaschinen angewendet, die auf die Suche wissenschaftlicher Publikationen ausgerichtet sind. Sie eignet sich auch für andere generische oder spezifische Domänen der Informationsintegration, z.B. zur Integration der verschiedensten WWW-Suchmaschinen (oder Suchwerkzeuge) und Online-Datenbeständen mit unterschiedlichen Benutzerschnittstellen und Anfragemodellen. Mit Hilfe von Quellen-Wrapping-Werkzeugen kann die Architektur zur Integration anfragbarer Informationsquellen verwendet werden, die semi-strukturierte Daten oder nicht-strukturierte Daten liefern (z.B. Wetterberichte, Softwareverzeichnisse ,etc.).

Acknowledgements

First of all, I would like to thank my thesis supervisor, Prof. Dr. Erich J. Neuhold, for his help, encouragement, advices, and comments, without which this PhD research work would never have been accomplished. I appreciate my second thesis reviewer, Prof. Dr. Alejandro P. Buchmann, for his helpful comments on my dissertation. I also thank the other members of my doctoral examination committee: Prof. Dr. Sorin A. Huss (the chair), Prof. Dr. Wolfgang Henhapl, and Prof. Dr. Wolfgang Bibel.

I am grateful to Dr. Matthias Hemmje for his constant encouragement, support, and technical suggestions. During my five years study and work at IPSI, I have got lots of help from my colleagues in the Delite division, and Barbara Lutes, Emil Wetzel, Ute Sotnik, and other IPSI colleagues, here I would like to thank them all together. Help from the Department of Computer Science, Technische Universität Darmstadt is appreciated.

I want to thank those anonymous reviewers for their precious comments on my previous publications that later contributed to this thesis. Special thanks go to Prof. Longxiang Zhou for his encouragement.

I would like to thank my wife and my parents for their support.

Finally, all those people who have ever helped me and showed solicitude for me are appreciated.

Lieming Huang
September 2003, Darmstadt

Contents

List of Figures

List of Tables

List of Definitions

Chapter 1
Introduction

In this thesis, we address three problems of Web meta-searching: (1) data modeling for describing the query input interfaces and query capabilities of heterogeneous, autonomous search engines and digital libraries on the Web; (2) translating queries from a meta-search engine into the specific formats understood by remote search engines; and (3) constructing adaptive, dynamically generated user interfaces for meta-search engines.

At the beginning of this chapter, the background of our research work is briefly introduced. Then in section 1.2, a motivating example is given to illustrate what kind of issues this dissertation tackles. In section 1.3, we discuss several problems encountered when integrating multiple heterogeneous search engines into a meta-search engine. In section 1.4, the main contributions presented in this dissertation are summarized. Finally, in section 1.5, the organization of this dissertation is described.

1.1 Background

With the tremendous developments in Internet technologies and the exponential growth of digital information, searching and browsing the World Wide Web (WWW or Web for short) [BLCG92, BLC+94, BL96] has become one of the most significant methods by which people can acquire information. It goes without saying that it is vital to provide users with a query tool to rapidly and efficiently search for what they need.

The Web is a huge and highly dynamic repository of information that is contained in various forms (such as HTML pages, programming codes in various languages, document files, data in traditional databases, etc.) and is authored by a wide variety of people with differing culture, education, and interests. Document sources are available everywhere, both within the intranets of organizations and on the Internet. There are many challenges to information searching on the Web. The Web is fertile ground for information retrieval (IR).

In this section, we introduce the background of this PhD research work. First, in section 1.1.1, we describe the rapid growth of various search engines on the WWW. Then, section 1.1.2 discusses searching on the Web for scientific publications, which is the starting point for our research work. Section 1.1.3 discusses some disadvantages of current Web search engines. In section 1.1.4, we discuss the Invisible Web. In section 1.1.5, the advantages of meta-search engines are discussed. Finally, section 1.1.6 briefly introduces SPOMSE, a scientific publication-oriented meta-search engine prototype.

1.1.1 The surging of search engines

Nowadays, a vast number of search engines and search tools exist on the Internet. Some search engines, such as Google [1], AltaVista [2], Infoseek [3], HotBot [4], Northern Light [5], WebCrawler [6], etc., index almost all accessible pages on the WWW by using spiders (or crawlers); some search engines only search local databases and repositories, such as ACM–DL [7], IDEAL [8], etc.; and others provide information on specific domains for users to query, such as NCSTRL [9], Weather search [10], DBLP Bibliography [11], MP3 search [12], and so on. In [LG98b], the authors pointed out, "Web search engines have made the large and growing body of scientific literature and other information resources accessible within seconds". These search engines allow users to enter keywords that are run against a centralized database and retrieve Web documents that match the keywords. Some search engines use their own indexing and retrieving software. Other search engines employ public software. For example, WAIS (Wide Area Information Servers) [KM91] is a widely used information retrieval system. Content providers can use it to create a searchable index of their information.

Most search engine users, however, do not know the working mechanism of search engines. They have the misconception that search engines search the Internet in real-time, i.e., searching all documents on all remote web sites just after a query is submitted to a search engine. Actually, users are searching a fixed database that has been compiled some time previous to their search.

Besides the differences in domains and functionalities existing among search engines, there are many differences with regard to other aspects. Most search facilities provide form-like query interfaces based on HTML, and others use Java Applet interfaces. Some offer a simple interface with only a text input box for arbitrary keyword(s), while some provide complicated forms with all kinds of specific selectors and input fields. Some match the user's query using only the Boolean retrieval model, and some use the vector-space retrieval model to match users' queries; others use probabilistic or thesaurus-based models. In section 3.1 (Chapter 3), we will discuss in detail the heterogeneities among data sources.

The Google search engine is a typical general-purpose Web search engine that makes heavy use of link popularity as a primary method to rank web sites. Its most relevant features are:

(1) It has indexed great numbers of web pages (about 2.5 billion web pages in 2002);
(2) Its page ranking algorithm based on link structure is effective [BP98];
(3) It can support searching for Web pages written in different languages;
(4) It can support searching for non-HTML file formats including PDF documents and others.

1.1.2 Scientific publication-oriented search engines

Many universities, research institutions, libraries, academic organizations, and publishing companies provide scientists and engineers with facilities on the WWW for searching online scientific literature (papers, technical reports, bibliographic information, etc). Today there exist many Internet information services for scientific research. Most search engines in these Internet scientific information services focus on specific domains or publications of one publisher, such as Elsevier Science [13], Kluwer academic publishers Online [14], ACM-Digital Library [7], IDEAL [8], ETRDL [15], etc.

However, a comprehensive tool for users to acquire the information from all domains and all publications does not exist. In order to realize this vision, some efforts have already been made. NCSTRL[9] is an international collection of computer science research reports and papers from a number of participating institutions and archives. In chapter 2, we will discuss the Dienst protocol [LSDK95] on which NCSTRL is based. The DBLP Bibliography [11] at Trier University provides bibliographic information (Author, Title, Year, Conference, and Journal) for the publications in major proceedings and journals of computer science.

Cora [16] is a special-purpose search engine covering computer science research papers. It applies machine learning to document classification, information extraction, clustering and spidering [MNRS99]. It crawls the Web and indexes information on computer science. In this sense, Cora is more like a "general-purpose" search engine, compared to most current scientific publication-oriented search engines that only provide query interface for local databases.

1.1.3 The deficiencies of search engines

People benefit a lot from all kinds of Web search engines. However, just like some authors pointed out and many search engine users perceived, search engines have a number of deficiencies:

(1) Low coverage of the WWW. Many studies have shown that the coverage of a single search engine is very limited [BB98, LG98a, SE95a, Bra97]. Therefore, it is not completely satisfactory to search for information on the Web by using only one search engine even if this search engine is most powerful.

(2) Inconsistent and inefficient user interfaces. We can say that even if you are an expert user of one search engine, you may not be an expert of all kinds of search engines.
(3) Poor query construction support. Users do not know the capabilities of a search engine.
(4) Poor relevancy ranking and precision.
(5) Difficulties with spamming techniques [LG98a].
(6) Users do not know where they can find the search engines that might provide the best answers for their queries. On the WWW, there are a lot of search engines, but most people only use several famous ones. However, some special-purpose search engines can better answer users' specific information needs than the general-purpose ones can do.
(7) Out of date databases. Because Web pages change frequently and Web information grows rapidly, it is difficult for a search engine to update its indexed information in time, i.e., to detect new or obsolete entries, and then to insert or delete them from databases. Today, how to construct efficient Web crawlers has become a hot research topic.
(8) Network delays [ABT97].

Moreover, there is one other big problem with search engines. This problem is:

1.1.4 The Invisible Web

On the Web there are innumerable information sources (databases and repositories) containing vast amounts of authoritative and very useful information that ***cannot*** be indexed by general-purpose search engines and are hence invisible to most common users who do not know where these sources are. These kinds of information sources form the so-called "Invisible Web" (or "Hidden Web" or "Deep Web"). Unlike pages on the visible Web, information in databases is generally inaccessible to the spiders and crawlers that compile search engine indexes by following the hypertext links in the Web pages. If there is no link to a page, a spider cannot see it. Users can get information from these kinds of specialized databases by typing queries on the query input forms. Most such results are dynamically created, instead of static pages identified by a URL. Though some dynamic pages have a unique URL address that allows them to be retrieved again later, they are not persistent. Another type of invisible Web pages are within the Web sites that require authorization or prior registration. It is believed that hundreds of thousands of special-purpose search engines currently exist on the Web [Berg00].

In [Cal00], the author argues: "The single database approach to Web search is a poor foundation upon which to build next generation systems. New, multi-database architectures provide a more solid foundation, because they explicitly model the multi-site, multi-resource nature of the Web". [Gra00] says: "current search engines ignore the often valuable contents of search-only text databases available on the web, since crawlers cannot download and index the contents of such databases." In [Berg00], Bergman estimated that the size of the "hidden web" was about 500 billion pages, while the largest general-purpose search engines index less than two billion pages of the 'crawlable' web.

Just as Chris Sherman pointed out: "Search engine spiders are the map makers of the Web. They roam freely through most Web servers, recording the addresses of Web pages they discover. When they come across a database, though, it's as if they've run smack into the entrance of a massive library with securely bolted doors. Spiders can record the library's address, but can tell you nothing about the books, magazines or other documents it contains." [17]

The Invisible Web is big, and it is getting bigger. There is a definite trend away from putting content onto the Web in static pages, and toward putting everything into databases. Databases are more robust, and allow Web sites to offer customized content that's often assembled on the fly from many parts of the database. For example, the Delite-Online [18] system is a knowledge management system containing all pages that are dynamically generated from Informix Universal Server databases, i.e., a server-site program creates a page when the request is received from a client. Oracle, Microsoft's ASP, and the Unix PHP also support the dynamic serving of Web pages.

1.1.5 The emergence of meta-search engines

In order to overcome the deficiencies of search engines, many meta-search engines have been produced. What is a “Meta-search engine”? There are all kinds of terms in the literature. Examples include: meta-search engine, integrated search engine, broker, meta-broker, search manager, receptionist, meta-service, information agents, and query intermediary. In this dissertation, we use the term “meta-search engine”.

A meta-search engine is an information retrieval system that supports unified access to multiple (remote or local) search engines or information sources.

In the following, we will discuss what kind of features meta-search engines have:

(1) “Unlike the individual search engines and directories, meta-search engines do not have their own databases; they do not collect web pages; they do not accept URL additions; and they do not classify or review web sites.” [Liu98]

(2) A single query targets multiple information sources. Users of a meta-search engine do not need to discover new Internet sources, which will be gathered and later organized into the meta-search engines by people in specialized fields.

(3) Result merging. Different search engines have different coverage of Web information. Therefore, a meta-search engine can provide users with more complete relevant results, i.e., improving recall. Meta-search engines can re-rank all results from various sources. The information indexed by search engines might be out-of-date; while meta-search engines can check the validity of URLs.

(4) Consistent user interface. A meta-search engine provides users with a uniform query interface and maps user queries into the various formats supported by heterogeneous search engines.

There are a lot of general-purpose meta-search engines on the WWW: Inference Find [19], Cyber411 [20], Internet Sleuth [21], MAMMA [22], Profusion [GWG96], to name a few. They generally integrate a number of search engines such as AltaVista

[2], InfoSeek [3], HotBot [4], etc. In chapter 2, we will introduce and discuss some meta-search engines.

1.1.6 The SPOMSE meta-search engine

Because almost all scientific publication-oriented search engines are domain-specific, it is better for a comprehensive scientific meta-search engine to integrate as many scientific search engines as possible. In this dissertation, we will discuss a prototype system: SPOMSE (Scientific-Publication-Oriented Meta-Search Engine). Up to now, SPOMSE has integrated several scientific publication-oriented search engines, including NCSTRL [9], ACM-DL [7], DBLP [11], Elsevier [13], Kluwer [14], Cora [16], ERCIM [15], and IDEAL [8].

SPOMSE has several features:

(1) It can efficiently utilize the query capabilities of heterogeneous information sources through an adaptive data model (see chapter 3) and a constraint-based query translation algorithm (see chapter 4);
(2) It can provide users with adaptive, dynamically generated query input interfaces (see chapter 5) that will facilitate users formulating queries and utilize the query capabilities of heterogeneous search engines as much as possible.
(3) It can achieve higher retrieval precision than other general-purpose meta-search engines for users' specific information needs. (See the experiments in chapter 7)

1.2 A motivating example

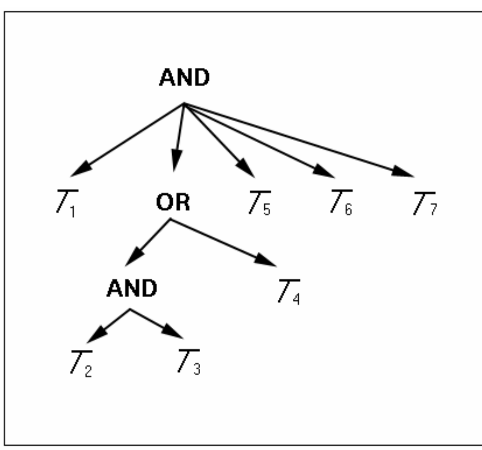
On the Web, there exist numerous kinds of search engines with differing query input user interfaces and query capabilities. The differences of search engines covering different domains are especially big. A meta-search engine provides a uniform user interface for users to input their information needs, then it translates user queries into the formats understood by the target search engines, and finally it merges all retrieved results and displays them to the users in a uniform style. One of the major problems challenging meta-search engines is the query translation problem. That is, how to translate the queries formulated by humans on the uniform query input interface of a meta-search engine into the specific formats understood by various heterogeneous search engines? Therefore, before tackling this problem, we first need to make clear: what are the mechanisms that Web search engines use to answer user queries?

In this section, we will present an example of a query for computer science publications and then show how some WWW search engines answer this query, thus demonstrating the great diversity among the query input user interfaces and query capabilities of different search engines and the difficulties of query translation.

Example

Suppose that a user wants to search for information as follows:
Q: ((Author is “Charlie Brown”) *AND* (((“Information Integration” in All fields) *AND* (Title contains “query”)) *OR* ((“Metadata”, “XML”) in Abstract))) in the Computer Science category, published during the period of 1995 to 1999, the results to be sorted by date.

This query is explicitly depicted in Figure 1.1.



T_1	Author is "Charlie Brown"
T_2	"Information Integration" in all fields
T_3	Title contains "query"
T_4	("Metadata", "XML") in abstract field
T_5	Date between 01.1995 and 12.1999
T_6	In computer science category
T_7	Sorted by date

Figure 1.1 Illustration of the example query *Q*

With the exponential explosion of information on the Web, IR systems are needed to help users precisely pinpoint and retrieve information. These kinds of queries are inevitably complicated and only two or three keywords are not enough to express users' information needs.

Now we choose three scientific publication-oriented search engines (i.e., NCSTRL, ACM-DL, IDEAL) and two general-purpose search engines (i.e., HotBot, AltaVista) to demonstrate how they answer this query. These five search engines have been selected for illustrating the problems to be addressed and solved because their query input interfaces and query capabilities are quite different and reflect the great heterogeneities of Web search engines and the difficulties in designing and implementing an efficient and effective meta-search engine with good flexibility and scalability. Figures 1.2-1.6 display the query input pages of these five search engines.

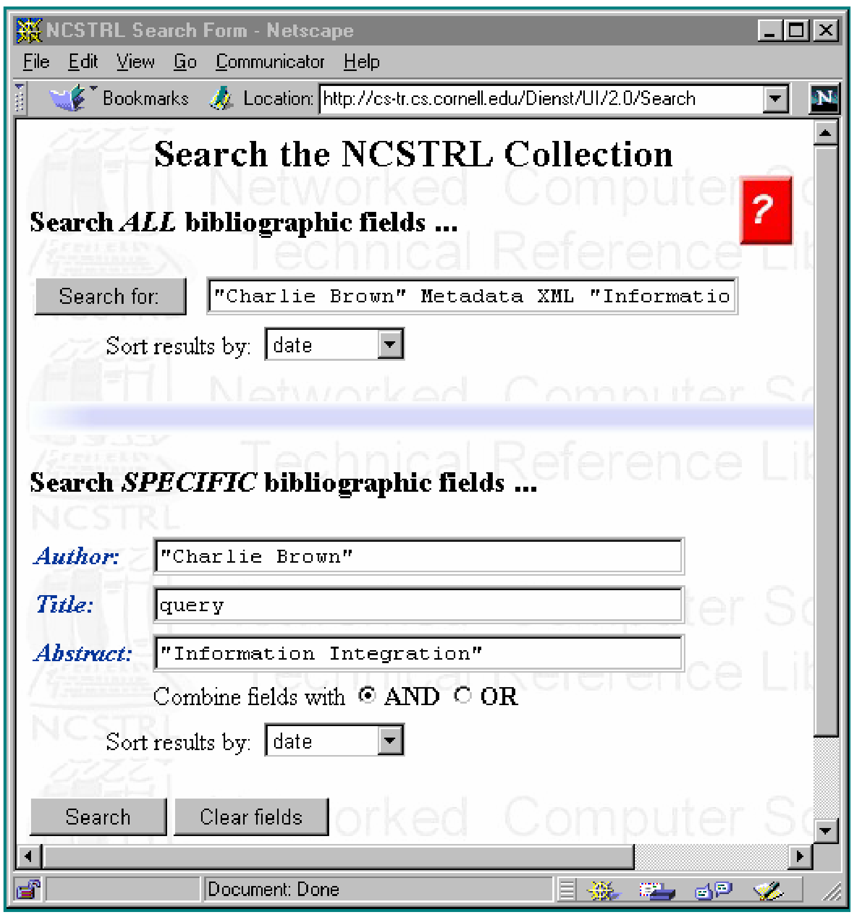


Figure 1.2 the query page of NCSTRL

Figure 1.2 displays the query input page of the NCSTRL (Networked Computer Science Technical Reference Library) search engine [9]. NCSTRL is an international collection of computer science research reports and papers from a number of participating institutions and archives. This page consists of two function-irrelevant parts. The first part (“Search ALL bibliographic fields …”) only contains an input-box and a result-sorting button. Therefore, when inputting the example query ***Q***, all searched keywords and phrases will be keyed into this input-box like this form (“Charlie Brown” “Information Integration” query Metadata XML). However, the constraint information, such as field modifiers (e.g., Title, Author, Abstract) and logical operators (i.e., AND, OR), will inevitably be discarded. Therefore, the precision of the result through this user interface will be low.

The second part (“Search SPECIFIC bibliographic fields …”) consists of (1) three input-boxes, each belonging to a specific field (“Author”, “Title”, or “Abstract”); (2) two radio buttons (“AND” and “OR”) for logically combining the above three input-boxes; (3) a result-sorting button. Obviously, queries inputted in the second part can achieve more precise results than queries inputted in the first part can do.

From Figure 1.2, we know that NCSTRL cannot completely support the original query ***Q***. In chapter 4, we will discuss how this query will be decomposed into three sub-queries that can be supported by NCSTRL.

Figure 1.3 displays the query page of the ACM Digital Library [7]. The ACM Digital Library now offers almost all ACM articles and proceedings. A query expression is built through the combination of terms (the specified words or phrases, wildcards '%'), search options (Contains phrase / Exactly Like, Stem expansion, Fuzzy expansion / Spelled Like, Sounds Like) and logical operators (AND, OR, NOT, NEAR). A query can be limited through the selection of the various categories presented. The refinement of a query is performed to further narrow down the result set obtained from a previous query. This result set is used as the scope within which the new query will run. A query can continue to be refined until the desired result is achieved.

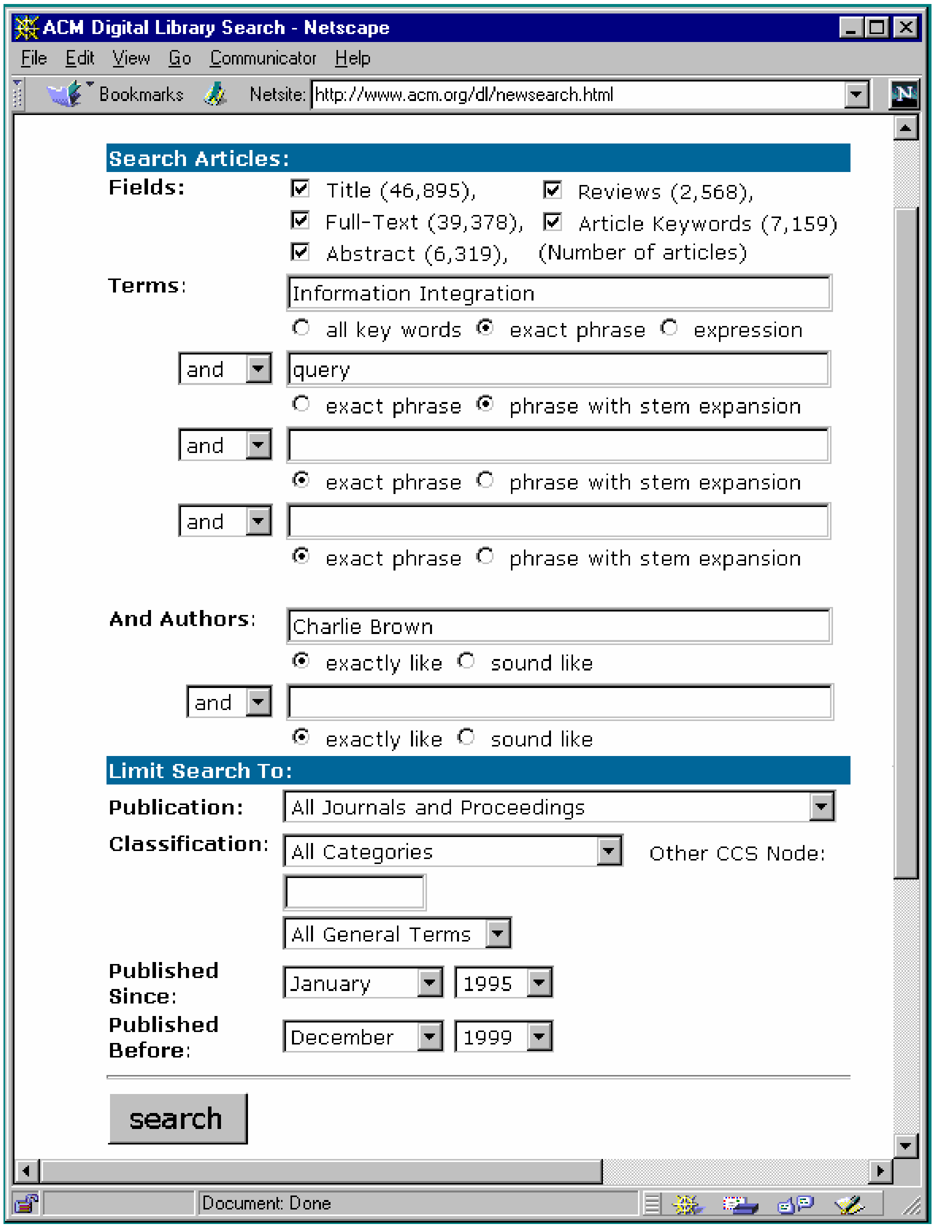


Figure 1.3 the query page of the ACM-Digital Library

The query input page in Figure 1.3 provides more controls than the one in Figure 1.2 does. It consists of five parts from top to bottom:

(1) There are five field modifiers: Title, Full-Text, Abstract, Review, Article Keywords. Each field modifier can be set by clicking a check box. Therefore, users can select several field modifiers to limit the keywords or phrases.

(2) There are four input-boxes for users to input keywords and phrases. The first input-box can accept three kinds of inputs, i.e. “all keywords”, “exact phrase”,

and “expression”. The other three input-boxes can accept two kinds of input: “exact phrase” and “phrase with stem expansion”. All four input-boxes are limited by the above-mentioned five field modifiers and can be combined by four logical operators (AND, OR, NOT, NEAR).

(3) There are two input-boxes for users to input authors’ information. Each input-box can accept two kinds of inputs: “exactly like” and “sound like”.

(4) There are some pull-down menus (also called *choices*) for users to limit the category.

(5) There are four pull-down menus for users to limit the range of publication date.

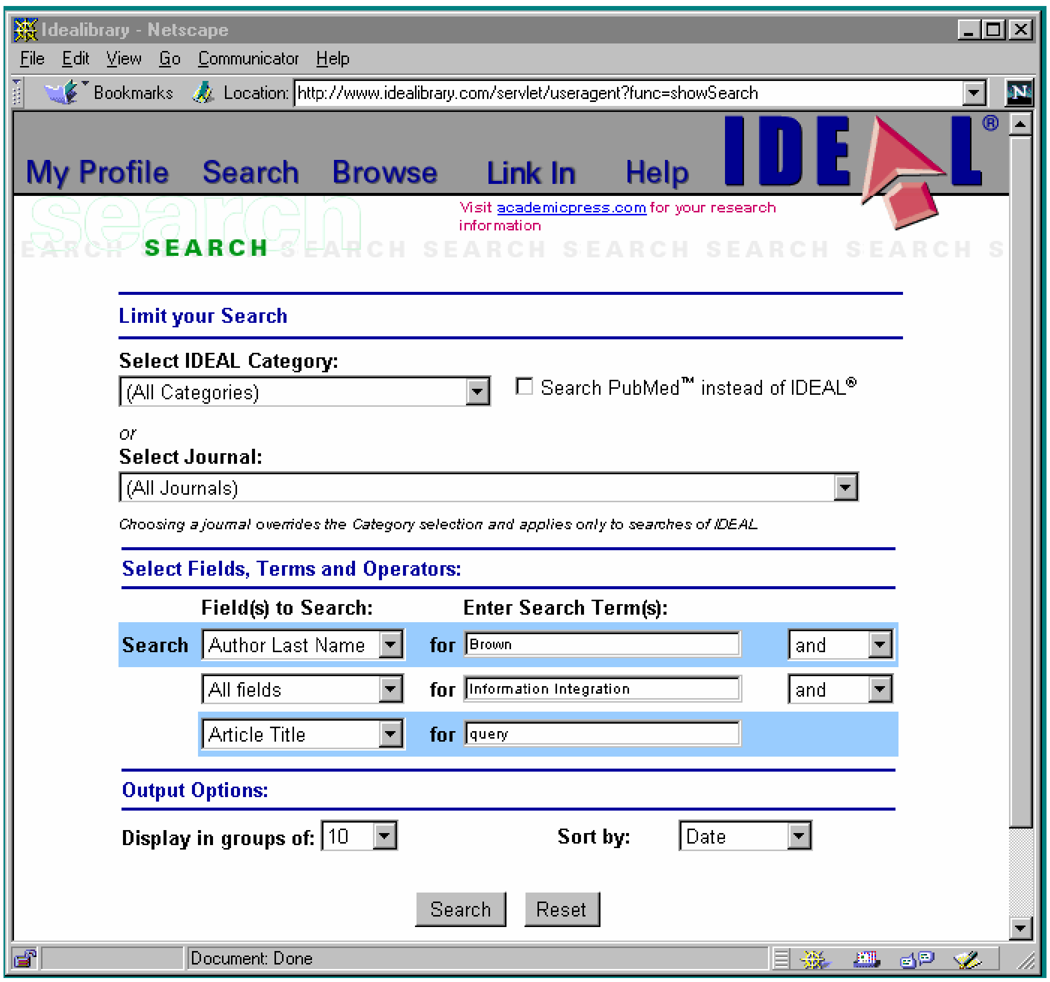


Figure 1.4 the query page of the IDEAL Search

Figure 1.4 displays the query input page of the IDEAL search service [8]. IDEAL (International Digital Electronic Access Library) is an online electronic library containing 174 Academic Press journals. IDEAL covers 11 categories (Biomedical Science, Business and Law, Engineering, Social Sciences, etc.). If users find that an article in the returned results is close to what they are searching for, clicking "More Like This" will perform a new search using the full article as the basis for the search.

The interface in Figure 1.4 consists of three parts:

(1) There are two pull-down menus for users to limit the searched category or searched journals.

(2) There are three input-boxes for users to input keywords, phrases, and authors' last names. There, users cannot search for information by using authors' full names.

(3) There are two pull-down menus, one is for result grouping size and another is for result sorting.

From Figures 1.2-1.4, we can see that there are a lot of functional discrepancies among different search engines. We can roughly divide all the controls in the user interfaces of search engines into three groups:

(1) Controls for users to limit the range of search, such as choosing a certain category or journal.

(2) Controls for users to input query expressions, such as input-boxes for keywords, pull-down menus for field modifiers, logical operators, etc.

(3) Controls for results output, such as sorting criteria and grouping size pull-down menus.

In chapter 3, we will systematically analyze the user interfaces of heterogeneous search engines and provide a formal data model (a conceptual framework) for describing the query capabilities of target sources.

Sometimes, people can get other useful information by searching general-purpose search engines for scientific publications, because this kind of search engine can index arbitrary web pages on the WWW. Most research organizations and researchers provide their homepages on the Web, from which users can find more complete and accurate publication information of an institute or a researcher. However, this kind of information cannot be acquired through publication-oriented search engines or digital libraries. In the following, we will discuss two examples of general-purpose search engines.

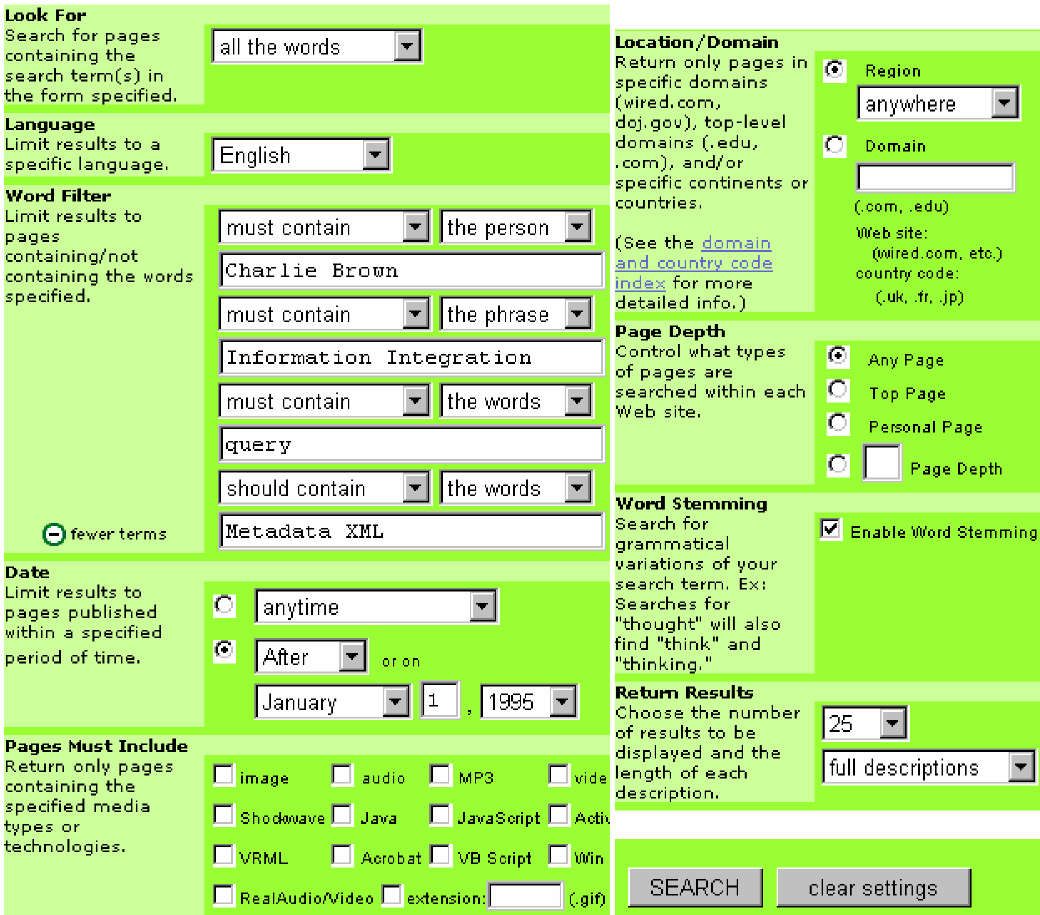


Figure 1.5 the query page of the HotBot search engine

In Figure 1.5, we can see the features of the user interface of the HotBot search engine [4], through which users can search for more specific information:

(1) Language supports. Users can search documents in a certain language, e.g., English, German, Spanish, etc.

(2) Users can use “must contain” to demand that the returned documents contain a keyword or phrase, and use “must not contain” to demand that the returned documents must not contain a keyword or phrase.

(3) Users can search for documents produced before or after a certain date.

(4) Users can search for documents that include some types of files.

(5) Users can search for documents in the websites of a certain country (.de, .jp, etc.) or domain (.edu, .org, etc.)

(6) Page depth.

In Figure 1.6, we can see that the AltaVista search engine [2] can support Boolean expression input. But users cannot use field modifiers (such as Abstract, Author, Title) in the query expression.

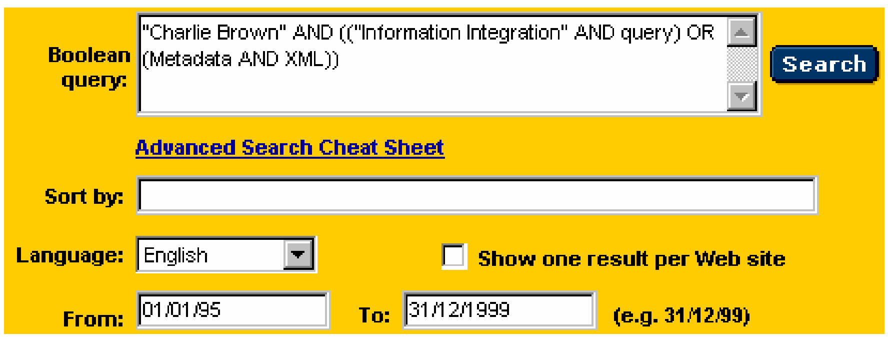


Figure 1.6 the advanced query page of the AltaVista search engine

Each search engine designs its query interface according to the characteristics of the database, the domain of the information, and the preferences of its designers. Because search engines are autonomous, each has its unique query interface and query capability. From the discussion of the above-mentioned five search engines, we know that even a human cannot easily express the original query in the user interfaces of various search engines. How can a computer automatically do it? In the next section, we will further discuss some problems encountered when integrating multiple heterogeneous search engines into a meta-search engine.

1.3 Problem statement

A meta-search engine provides a uniform user interface for users to formulate queries and then displays the merged results from heterogeneous sources in a uniform style.

From Figures 1.2-1.6, we can see that there are some commonalities among search engines. At the same time, many differences exist between them. Current meta-search engines use these common features to build the integrated query interface and cast away the discrepancies. This will inevitably lead to the loss of many important functions. "From the users' perspective the integration of different services is complete only if they are usable without losses of functionality compared to the single services and without handling difficulties when switching from one service to another" [23]. Making full use of the specific functions of search engines will alleviate such weaknesses.

Building a meta-search engine is not easy because different search engines are largely incompatible and do not allow for interoperability. Furthermore, the information sources on the Internet are too numerous. From the example in section 1.2, we know that there are some problems:

(1) **Data modeling.** There are great differences among the user interfaces and query capabilities of various search engines. How can we describe these various user interfaces and query capabilities and enable the meta-search engine to use them effectively and efficiently? At the same time, considering the exponential growth of information on the Web, how can a meta-search engine achieve great scalability and flexibility?

(2) **Source selection.** There are numerous kinds of search engines on the Web; users do not know which search engines can answer their queries. How can a meta-search engine decide which target search engines may return relevant results for a user query? This problem is also known as "search engine selection", "source selection" or "query routing" problem.

(3) **Query Translation.** When providing users of a meta-search engine with a uniform query construction interface, how can queries be translated from this uniform format into the formats supported by various target search engines?

(4) **User interface construction.** Formulating good queries can be a difficult task, especially in an information space unfamiliar to the user. There are great discrepancies among the user interfaces of search engines. How can users be provided with a uniform interface that masks the differences of heterogeneous search engines? A simple user interface cannot sufficiently utilize the specific functionalities of target sources. Static user interfaces are also not flexible and

scalable enough. How can we construct an adaptive, dynamically-generated user interface for a meta-search engine? The experiments in chapter 7 show that an adaptive, dynamically generated user interface can best satisfy users' information needs.

1.4 Contributions

This dissertation reports on how to construct adaptive meta-search engines for improving the retrieval precision of domain-specific Invisible Web meta-search under heterogeneous, autonomous environment. The main contributions of the dissertation include:

1. This dissertation proposes a novel **adaptive data model** (ADMIRE) for meta-search engines that can be used to formally and meticulously describe the user interfaces and query capabilities of heterogeneous, completely autonomous search engines and digital libraries on the Internet (Chapter 3). Current information integration systems mainly focus on the schema coordination in traditional multi-databases and federated or cooperative information systems, and they seldom consider the coordination of various constraints among the user interfaces and query models of information sources. However, for a Web meta-search engine, the target search engines are autonomous and their internal database schemas and indexing structure (and algorithms) are invisible to outsiders. A meta-search engine can only use the publicly accessible information of integrated remote sources (such as query input pages and result pages). The key feature of this data model is the focus on constraints between terms, term modifiers, attribute order, and the impact of logical operators. The query capabilities of a search engine or a meta-search engine can be denoted by a query expression that describes its query model (Sequential/Priority Boolean model or Vector-space/Probabilistic model), query terms with their constraints (field modifiers and term qualifiers), and the logical relationship between terms (AND, OR, NOT, NEAR). By using this data model, meta-search engines can utilize the functionality of each remote information source to the fullest extent.

2. This dissertation presents a novel **constraint-based query translation algorithm** (Chapter 4). When translating a query from a meta-search engine to a remote source, the mediator considers the functional and positional restrictions of terms, term modifiers and logical operators among the controls in the user interfaces to the underlying sources, thus allowing the meta-search engine to utilize the query capabilities of the specific sources as far as possible. In addition, a two-phase query subsuming mechanism is put forward to compensate for the functional discrepancies between sources, in order to make a more accurate query translation. Current systems employ query translation methods that focus on vocabulary (e.g., predicate rewriting) but not the syntactic structures of queries, and they have limited support for coping with special query constraints.
3. This dissertation presents a novel mechanism for **constructing adaptive, dynamically generated user interfaces** for meta-search engines (Chapter 5). The concept of control constraint rules has been proposed and applied to the user interface construction. Depending on the state of interaction between users and system, such meta-search engines adapt their interfaces to the concrete user interfaces of differing kinds of search engines (Boolean model with differing syntax, vector-space/probabilistic model, natural language support, etc.), so as to overcome the constraints of heterogeneous search engines and utilize the functionality of the individual search engines as much as possible. Current information integration systems adopt simple or static user interfaces that are not effective and flexible for accessing heterogeneous, autonomous information sources. The adaptive mechanism proposed in this thesis can facilitate the reduction of constraints, and consequently make the query translation more accurate and easy. Adaptive user interfaces can support the progressively self-refining construction of users' information requests, and can match user queries to the queries supported by target sources as far as possible.

This dissertation proposes a source selection method (Chapter 4) by utilizing the information of classification selection controls from the user interface and domain information about the integrated search engines. Therefore, users only have to enter information needs by manipulating these controls in the user interface of a meta-search engine, and the meta-search engine will automatically select target sources. Our source selection method can overcome deficiencies of the traditional statistics-based methods: 1. Sources with relevant documents are not searched if they are below thresholds; 2. They are practical for relatively static sources but could be problematic for widely distributed, autonomous, and dynamic sources.

This dissertation also tackles some issues on wrapper generation and result merging for Web information sources (Chapter 6). The adaptive meta-search engine architecture proposed in this dissertation has been applied to the information integration of scientific publications-oriented search engines. It can also be applied to other generic or specific domains of information integration, such as integrating all kinds of Web search engines (or search tools) and online repositories with quite different user interfaces and query models. With the help of source wrapping tools, they can also be used to integrate queryable information sources delivering semi-structured or non-structured data, such as product catalogues, weather reports, software directories, and so on.

This dissertation organically couples an adaptive data model for meta-search engines, a mechanism for constructing adaptive, dynamically generated user interfaces of meta-search engines, an algorithm for constraint-based query translation, and flexible wrapper generation for arbitrary Web information sources. It also systematically expounds the collaboration of these modules. The experiments (Chapter 7) show that an information integration system with an adaptive, dynamically generated user interface, coordinating the constraints among the heterogeneous sources, will greatly improve the effectiveness of integrated information searching, and will utilize the query capabilities of sources as much as possible.

1.5 Organization

The remaining chapters of this dissertation are organized as follows:

Chapter 2. Related work

This chapter begins by briefly introducing some background information on Information Retrieval. Then we discuss related work on information integration systems, and compare different methods of source selection, query translation, and query user interface construction. Then we discuss Web meta-search engines and Invisible Web catalogues. Finally, this chapter discusses relevant standards and protocols for Web meta-searching.

Chapter 3. ADMIRE data model

This chapter proposes a data model to formally describe the query capabilities of search engines. At the beginning, this chapter analyzes the user interfaces of Web search engines, and introduces some basic definitions and advanced features, and then gives the formal definition of query expressions and uses this definition to model three concrete domain-specific (scientific publication-oriented) search engines and a corresponding meta-search engine.

Chapter 4. Constraint-based query capability translation

This chapter first discusses the source selection problem. Then, a formal query translation algorithm is presented and the query translation problems are analyzed. Finally, it illustrates a concrete example of query translation and shows some problems.

Chapter 5. Adaptive user interface generation

This chapter begins by introducing and comparing three kinds of user interfaces for meta-search engines. It then defines the concept of control constraint rules, which can be used to construct adaptive, dynamically generated user interfaces to meta-search engines. It also discusses the mechanism of constructing adaptive user interfaces for meta-search engines.

Chapter 6. Implementation

This chapter discusses some technical issues on the implementation of our adaptive meta-search engine prototype, including system architecture, wrapper generation, result merging, and user interface design.

Chapter 7. Evaluation

This chapter discusses three groups of experiments. The first group contains three experiments comparing the efficiency of our prototype with three different types of user interfaces (i.e. simple, static mixed, and dynamically-generated). The second group contains two experiments comparing our meta-search engine prototype with four general-purpose meta-search engines (i.e., comparing the effectiveness of the Invisible Web with the Visible Web for the users' specific information needs). Finally, the third group of experiments checks the number of generated sub-queries and post-filters for eight complex queries.

Chapter 8. Conclusions

This chapter summarizes the contributions of this dissertation, describes some application spheres of Web meta-search technologies, and points out some directions of future research in related fields.

Chapter 2

Related Work

Exponential growth of information on the Web sets off an upsurge of research on designing efficient and effective Web information retrieval (IR) systems. Currently, popular search engines, such as Google and AltaVista, adopt a centralized database architecture. They apply traditional IR technologies to the Web environment. At the same time, they also consider peculiar features of the Web documents, for example, the link usage in the hypertext [BP98, Klei99], that are not features of documents in the sense of traditional IR. In view of the limitations of Web search engines that we have discussed in chapter 1, new multi-database search mechanisms, such as distributed search or meta-search, have been researched and developed to make up for the deficiencies of search engines. Because a meta-search engine is built on top of target search engines, its query construction interface, query capabilities, and results display interface will be quite similar to the underlying search engines. Some users even cannot perceive the differences between search engines and meta-search engines. Therefore, the developer of a meta-search engine should know well the functionalities of search engines and keep pace with their development; otherwise, the quality of a meta-search engine cannot be guaranteed.

The state of the art of research on meta-searching technologies can be divided into four categories: (1) information integration systems; (2) Web meta-search engines; (3) Invisible Web catalogues; and (4) standards and protocols. Each of these four categories contains many research topics. In the following, we only discuss and compare the topics that are relevant to the research focuses of this thesis. As mentioned in chapter 1, this thesis addresses three problems of Web meta-searching: data modeling, query translation, and intelligent user interface construction. Just as what will be discussed in the following sections, current Web meta-search engines and Invisible Web catalogues only provide users with a very simple query input interface (usually an input-box with some other auxiliary controls). They only utilize the basic query functionalities of underlying search services, and their capability for data modeling and query translation is primitive. Therefore, in this thesis, Web meta-search engines and Invisible Web catalogues will not be discussed and compared as focal points. Many standards and protocols have been laid down to improve Web meta-search. They demand all participating search services comply with strict rules. However, in our meta-search engine the target sources are completely autonomous. Information integration systems (from the database community, focusing on managing data from multiple databases) have many similarities with meta-search engine systems (from the IR community, focusing on searching information from distributed collections). In this chapter, we put more emphasis on discussing and comparing information integration systems from the point of view of data modeling and query translation.

This chapter is organized as follows. Section 2.1 starts by introducing background information on Information Retrieval. Then, section 2.2 discusses information integration systems relevant to our system and compares different methods of query translation, source selection, and query user interface construction. In section 2.3, the user interfaces and query capabilities of several general-purpose meta-search engines on the Web are discussed. In section 2.4, we discuss Invisible Web catalogues. Finally, section 2.5 reviews standards and protocols relevant to Web meta-search technologies.

2.1 Background material

In this section, some basic concepts of research on Information Retrieval are briefly introduced.

Information retrieval systems

Information retrieval systems are software tools with which users can submit queries and receive references to documents contained in a specific corpus or database. A typical information retrieval system responds to a query with a ranked list of documents. Queries are representations of users' needs for information. They contain words (also called terms), and sometimes may include other information, such as how the words relate to one another. Documents are the actual objects of the users' searching activities.

Index

Typically, the full text of all documents is combined into a data structure called an inverted index that maps words to a set of documents which contain them. Each word appearing one or more times in the corpus has a corresponding entry in the inverted index. Three transformations are often applied to words in the corpus to make the index structure more useful for searching and more space-efficient: (1) Case differences among words are standardized. This means all capitalization variance is eliminated and every word is represented entirely in lower case letters. (2) Stop words are eliminated. These small words, such as 'the', 'and', 'if', usually occur many times throughout the collection, and thus would result in extremely long lists if included in the index. Some search engines also support phrase searching when the stop words are indexed as well. For example, the phrase "Gone with the wind" can be retrieved even though there are two stop words in it: "with" and "the". (3) Suffixes or variations in the form of a word are stripped. The suffix removal process is called stemming. Along with every transformed word in the inverted index is a list of pointers to each document where that word occurs. Other information can be stored in the index, such as the total number of occurrences of the term in all documents combined, the number of occurrences of the term in each document where it appears, and even the exact location of each occurrence of the word within the page might be included.

Boolean search

The most common method for searching information in a document repository is by using Boolean searches [Sal91]. In this type of search, a number of keywords combined with Boolean operators (such as “and”, “or”, and “not”) are specified, and the result consists of the documents that satisfy the given Boolean expression. The retrieved documents are all equally ranked with respect to relevance and the number of retrieved documents can only be changed by reformulating the query [AS96]. In order to solve these problems, the Boolean retrieval has been extended and refined. Expanded term weighting operations make ranking of documents possible, where the terms in the document could be weighted according to their frequency in the document [Sal83]. In this thesis, we have focused on supporting Boolean queries at the front-end. The Boolean query model is used by most commercial search services and traditional library systems to access their text databases.

Vector-space and probability based indexing and retrieving

As more and more information retrieval systems adopt vector-space or probability based indexing and retrieving models instead of the Boolean model, we briefly introduce the relevant properties of Probabilistic and Vector-space models as well. A Probabilistic model [Fuhr92] considers the probability that a term or concept appears in a document, or that a document satisfies the information need. A Bayesian inference net is a good representation and processing framework for this type of IR model [Rijs79]. In the Vector-space model [WZRW87], representations of documents and queries are converted into vectors. The features of these vectors are usually words in the document or query, after stemming and removing stop-words. The vectors are weighted to give emphasis to terms that exemplify meaning, and are useful in retrieval. In a retrieval operation process, the query vector is compared to each document vector. Those that are closest to the query are considered to be similar, and are returned.

Tf*Idf weighting

Whenever a user submits a query to a search engine, the index is consulted and the information for each query term is looked up in the inverted index. Search engines using the common Tf*Idf (Term Frequency times Inverse Document Frequency) [Sal89] ranking algorithm exploit two important qualities of natural language text to perform accurate retrieval [WMB94]: (1) Frequency – if a term occurs frequently in a document, that document is considered more relevant to a query than other documents with fewer occurrences of the same term. (2) Scarcity – in a multiple word query, the rarer terms (those that occur in very few documents) receive more substantial weight in determining document relevance. Tf*Idf weighting is used to combine the frequency and scarcity information when ranking documents for a query.

Evaluation

The standard criteria for evaluating information retrieval systems are precision and recall. Precision is the proportion of search results for a query that are actually relevant; it measures the accuracy of the system. Recall is the proportion of all relevant search results that appear in the list of results and measures the thoroughness of the system.

CGI

The Common Gateway Interface (CGI) [49] is a standard for interfacing external applications with information servers, such as HTTP or Web servers. A CGI program is executed in real-time, so that it can output dynamic information. CGI specifies how data are sent to the gateway program (as environment variables or as data read from standard input), and how the gateway program returns data to the client program (the user's Web browsers) that originally sent the data. A CGI program can be written in any language that allows it to be executed on the operating system, such as C/C++, PERL, TCL, Unix shell, and Applet Script. The remote Web search engines integrated in our SPOMSE meta-search engine prototype use CGI programs to answer user queries.

2.2 Information integration systems

Since the beginning of last decade of twenty centuries, there has been great interest in building information agents that can integrate information from multiple data sources [LMR90, SL90, TTC+90, ASD+91, IC91, ACHK93, HM93, RRM93, CC94, GGMT94, GR94, SAD+94, CHS+95, CLC95, FSF95, GGM95, GMHI+95, Hei95, KLSS95, LHS+95, LP95, LSH95, SAB+95, ACPS96, AKS96, CYC+96, Etz96, GWG96, HKWY96, LPL96, LRO96, MKSI96, WHC+96, TRV96, ABT97, BBB+97, BW97, CPW+97, DH97, DM97, GGM97, GMPQ+97, HKWY97, LP97, NR97, SC97, TRS97, TRV97, Ull97, VP97, YL97, AB98, AFT98, AKMP98, BBE98, BKP+98, BT98, CBC98, CDSS98, DFGL98, HFAN98, MLY+98, NGT98, SV98, UFA98, VZ98, CGMP99, FAD+99, FLMS99, Fuhr99, Gal99, GGMT99, GMLY99, GMW99, GMY99, HT99, IFF+99, LL99, LXLN99, MK99, Par99, PHW+99, TRÖH99, YLGMU99, AH00, CDTW00, CGM00, GW00, LC00, PTK00, SE00, BFPV01, GIG01, IGS01, KW01, WMYL01, YMWL01, ZRZB01, CGJM02, IG02, MYL02, ZRV+02, CC03, GIS03]. Today, the research on information integration is still a very hot topic due to the explosive growth of information on the Internet. Unlike general-purpose meta-search engines, information integration systems focus more on the description of data structures and query languages, and the utilization of query capabilities. These information integration systems integrate all kinds of relational databases, object-oriented databases, semi-structured information sources, and also search engines.

Compared with these systems, our meta-search engine has both similarities and differences with them. On the one hand, because our SPOMSE meta-search engine integrates scientific publication-oriented search engines, repositories, and digital libraries, our system is more like the information integration systems discussed in this section than the general-purpose meta-search engines that will be discussed in section 2.3. On the other hand, because we only can get the information from the user interfaces (i.e., query input CGI forms and results pages) of the remote autonomous information sources and their internal database schemas and indexing schemes are completely invisible to us, our meta-search engine is different from traditional information integration systems, such as multi-databases and federated databases. The query input interface of an information source usually has more operational limitations than the actual query capability of the information source. In paper [CBC98], Chidlovskii et al. also pointed out this kind of difference: "Any Web search service is connected to an underlying search engine through, for example, a cgi-script. However, the user deals with the Web page query language rather than with the underlying search engine. The expressive power of the Web page query language can obviously not be higher than that of the underlying search engine. Nevertheless, even in the presence of a powerful search engine the Web page designer is often driven by the application needs which may result in a Web page query language which is far different from the underlying query language". In paper [GMY99], Garcia-Molina and Yerneni listed four reasons for query capabilities of sources: "1. Sources may choose to provide simple query interfaces through search forms even if they are actually capable of answering more complex queries; 2. Certain attributes may be hidden for security reasons. Such attributes may not appear in the query results and/or they may not appear in the query condition specification; 3. Sources may only allow conditions on attributes for which they have indexes so that certain query response-time profiles can be maintained; 4. There may be published application programming interfaces (APIs) that are in current use, so they continue to support such interfaces even when they are restrictive."

The representative information integration systems include CARNOT [WHC+96], CoBase [CC94, CYC+96], DIIM [NR97], DIOM [LPL96, LP97], DISCO [TRV96, TRV97], FUSION [SC97], GARLIC [CHS+95, HKWY97, HTRO99], HERMES [ACPS96], Information Manifold [KLSS95], InfoSleuth [BBB+97], IRO-DB [FSF95], Myriad [LHS+95, LSH95], OBSERVER [MKSI96], Pegasus [SAD+94], SIMS [ACHK93, AKS96], and TSIMMIS [GMHI+95]. Such systems allow database-like querying of semi-structured Web sources through wrappers around the Web sources, and also provide integrated access to multiple data sources. Besides, there is a lot of research work on data warehousing technologies that construct customized data collections derived from multiple distributed sources. Here we don't expand the discussion on data warehouse and its relevant technologies. In this thesis, we only consider the information integration systems that store and manage meta-information (not actual data) from heterogeneous, autonomous remote sources.

The autonomy and heterogeneity of information sources present some differences from traditional distributed databases. These differences are caused by constraints which are characteristic of the underlying environment; for example, different sources may differ in their query-handling ability, cost models may not be known, and data conversions may incur large hidden costs. In [BBE98], Bouguettaya et al. pointed out that one of the major challenges in integrating multiple heterogeneous information sources is in understanding and translating the data from all the data sources into a common context. The main difficulty in this process is the presence of semantic heterogeneity among the data and meta-data (schema) in the different data sources. This semantic heterogeneity is caused by the design autonomy of the data sources.

2.2.1 Data modeling and systems comparison

In the following, we will discuss and compare some research projects with aims similar to those of our SPOMSE project.

Information Manifold

The Information Manifold system [LSK95, LRO96, KLSS95] is a global information integration system (developed at AT&T Research) where the external information sources are described against a unified world-view by query expressions. It uses capability records that specify 5 tuples of information with respect to each source (input set, outputs of the source, the selections the source can apply, the minimum and maximum number of inputs allowed); and it gives an algorithm for query planning. "IM (Information Manifold) is a knowledge based information retrieval and management environment that enables one to interact with the Web in terms of knowledge about information, rather than dealing strictly with primitive parcels of information" [Kir96]. The system provides clients with means to superimpose a tailored conceptual organization on an unstructured information space, enriching the usefulness of and access to that information.

However, in such a tightly-integrated model, the definition of the world-view relations and descriptions of the external information sources are easy to change schematically if new information sources are integrated into the system and they cannot be described relative to the existing world-view relations. Furthermore, when dealing with query translation, this system has very limited consideration of some special constraints, such as the limitations of term modifiers, logical operators and the order of terms. Our SPOMSE meta-search engine prototype describes all kinds of constraints between the query models (as embodied in the user interfaces) of various sources, and therefore can utilize the functionality of each source to the fullest extent.

TSIMMIS

The TSIMMIS system – **T**he **S**tanford-**I**BM **M**anager of **M**ultiple **I**nformation **S**ources [GMPQ+97, PGMU96] uses OEM (Object Exchange Model) [PGMW95] as a basis for information integration and adopts a centralized mediator/wrapper architecture. The OEM is a simple OO model and each object consists of four elements: label, type, value, and object-id. Because of the simple model, the TSIMMIS system can easily generate wrappers and mediators by using MSL (Mediator Specification Language). The emphasis in the TSIMMIS system is that of automatic generation of translators and mediators for accessing and combining information in heterogeneous data sources.

However, its query performance (lacking scalability and flexibility on query processing) is not so strong and the OEM data model is not well suited for complex objects. Compared with this model, the ADMIRE data model presented in this thesis can be employed to wrap complex sources. Furthermore, as pointed out in paper [YCGMU99], "the TSIMMIS mediator does not describe its capabilities to the user, so the user has to submit queries in a trial-and-error scenario".

DIOM

The Distributed Interoperable Object Model (DIOM) project [LP95, LPL96, LP97] has developed a distributed mediation framework based on the ODMG-93 data model. A network of specialized information mediators is deployed to facilitate application access to the data in the data sources. A meta-mediator can be instantiated to build multiple specialized mediators. The goal of this project is to provide a scalable platform for uniform access to autonomous and heterogeneous systems based on evolving and composable mediators. The DIOM query mediation architecture can be generalized into a five-phase process: query routing, query decomposition, parallel access plan generation, sub-query translation and execution and query result assembly.

However, the DIOM model focuses mainly on the integration of schemas of RDBMSs (Relational Database Management Systems) and OODBMSs (Object-Oriented Database Management Systems). Furthermore, it has limited support for describing the specific constraints in the query interfaces and query capabilities of disparate information sources and tackling constraints problems when translating queries from the mediator to a specific wrapper. Our SPOMSE meta-search engine prototype can utilize the query capabilities of target information sources as much as possible by employing a constraint-based query translation algorithm.

InfoSleuth

In order to support mediated interoperation of data and services over the Internet, the InfoSleuth project [BBB+97] integrates agent technology, ontologies, and information brokerage technology with traditional approaches to querying (SQL) and schema mapping. InforSleuth is comprised of a network of cooperating agents communicating by means of agent query language KQML [FFMM94]. Users specify requests and queries over specified ontologies via applet-based user interfaces. These user queries are internally represented by the database query language SQL and the knowledge representation language, and routed by mediation and brokerage agents to specialized agents for data retrieval from distributed sources. A resource agent provides a mapping from the common ontology to the database schema and language native to its resource, and executes the requests specific to that resource.

The InfoSleuth project relies upon using ontologies to capture the database schema (e.g., relational, object-oriented, hierarchical) and conceptual models (e.g., E-R models, object models, business process models). However, the designers of Web meta-search engines cannot get such kind of information from "uncooperative" sources. Furthermore, the InfoSleuth system does not consider the specific constraints in the query interfaces when translating queries from the mediator to a specific wrapper.

Others

The DISCO project [TRV96, TRV97] provides support for integrating unstable data sources in a dynamic environment. Mediation in DISCO is based on the ODMG-93 standard object model. The GARLIC [CHS+95, HKWY97, TRÖH99] project, developed at the IBM Almaden Research Center, is also based on ODMG's OO data model. The data from the wrapped data sources is represented as objects. The Pegasus [SAD+94] project focuses on integration of relational databases, multimedia databases, and legacy applications. It aims at the seamless integration of external schemas with the local databases. The IRO-DB (Interoperable Relational and Object-Oriented Databases) project [FSF95] developed tools for unified access to a number of relational and OO databases. It is based on the ODMG standard data model and the query language OQL. The OBSERVER project [MKSI96] represents an approach for query processing in global information systems. In the SIMS project [ACHK93, AKS96], a model of the application domain is created using a knowledge representation system to establish a fixed vocabulary describing objects in the domain, their attributes, and relationships. Myriad [LHS+95, LSH95] is a federated database project developed at the University of Minnesota. The federation is defined as an integrated database with a global schema consisting of a set of global relations. An SQL-like language is used to query the integrated database schema. These projects and systems seldom consider the coordination of various constraints among the query models and user interfaces of information sources and do not compute and export the query capabilities of their mediators. Their methods focused mainly on the domain and schema coordination in traditional multi-databases and federated or cooperative information systems.

Most multi-database systems rely on building a single global schema to encompass the differences among the multiple local database schemas. The mapping from each local schema to the global schema is usually expressed in a common SQL-like language, such as HOSQL in the Pegasus system [SAD+94]. Although using a single global schema through data integration can achieve full transparency for uniform access, component databases have very restricted autonomy and scalability. In addition, such systems are focusing on querying only structured databases. In the case of Web meta-search engines, this method is impractical or impossible because component information sources are completely autonomous and heterogeneous.

The federated approach [SL90] improves the autonomy and the flexibility of multi-database management by relying on multiple import schemas and the customized integration at various levels. However, this approach also cannot scale well when new sources are added into an existing multi-database system, because the heterogeneities are resolved at the schema integration stage. In addition, the component schemas cannot evolve without the consent from the integrated schema. The Carnot [WHC+96] federated database project did static data integration, and was not designed to operate in a dynamic environment where information sources change over time, and where new information sources can be added autonomously and without formal control. In section 2.5, we will discuss some standards and protocols that are also intended to facilitate information integration over distributed information sources.

For a Web meta-search engine, the target search engines are completely autonomous and their internal database schemas and indexing structure and algorithms are invisible to outsiders. In chapter 3, we will present the ADMIRE data model that can be employed by meta-search engines and information integration systems to analyze and describe the user interfaces, the various constraints existing among them, and the query capabilities of Web search engines.

2.2.2 Query translation

Although the distributed query-processing problem has been well studied in the domain of structured (especially relational) databases, there is very little research in processing distributed queries that involve autonomous Web information sources. The internal database schemas and indexing schemes of these autonomous information sources are completely invisible to outsiders. The only usable things for the designers of meta-search engines are the query input pages and results pages on the user interfaces of remote information sources.

Different information systems adopt different data models and query languages. Therefore, a meta-search engine must consider the translation of queries across heterogeneous sources. There are all kinds of query translation methods. However, most of these systems (such as HERMES [SAB+95], MODM [HM93], Pegasus [ASD+91, SAD+94], etc.) do not address the problem of different and limited query capabilities in the underlying sources because they assume that those sources are full-fledged databases that can answer any queries over their schema. The recent interest in the integration of arbitrary information sources, including databases, file systems, the Web, and many legacy systems, invalidates the assumption that all underlying sources can answer any query over the data they export and forces researchers to resolve the mismatch between the query capabilities provided by these sources. HERMES [SAB+95] proposes a rule language for the specification of mediators in which an explicit set of parameterized calls can be made and then sent to the sources. At run-time, the parameters are instantiated by specific values and the corresponding calls are made. Thus, HERMES guarantees that all queries sent to the wrappers are supported. Unfortunately, this solution reduces the interface between wrappers and mediators to a very simple form (the particular parameterized calls), and does not fully utilize the sources' query power.

Papers [CBC98] and [CGMP99] apply user-defined mapping rules to subsume queries for translation between different sources. They describe some problems involved in predicate rewriting, such as the "contains" predicate and word patterns, the "equals" predicate and phrase patterns, proximity operators, etc. They focus on vocabulary but not the syntactic structure of queries. Compared with their work, we propose a more generic model for translating arbitrary queries supported by various sources. Our two-phase method for coping with query subsuming (relaxing and decomposing) and post-processing (tightening with common filters and composing with special filters) can coordinate functional discrepancies among heterogeneous information sources.

DISCO [TRV97] describes the set of supported queries using context-free grammars. This technique reduces the efficiency of capabilities-based rewriting because it treats queries as "strings". Many papers, such as [LRO96], [VP97] and [YLGMU99], describe the query capabilities of sources and deal with the query translation problem. They discuss more on context-free, conjunctive queries and have very limited support for coping with special constraints, such as the limitations of term modifiers, logical operators and the order of terms. From Figures 1.2-1.6 in chapter 1, we know there is great diversity among sources. Sometimes even a very subtle difference will render the query translation impossible. Our work sufficiently describes all kinds of constraints between the query models (as embodied in the user interfaces) of various sources, and therefore can utilize the functionality of each source to the fullest extent.

Adali and Bufi [AB98] used Church-Rosser systems to characterize the query capabilities of information sources and uses "Attribute Preference Ordering" to realize query relaxing. Garcia-Molina et al. [GMLY99] proposed a scheme called "GenCompact" for generating capability-sensitive plans for queries on Internet sources. These two papers try to describe the query capabilities of sources and to translate queries across sources in a generic view. However, they have limited consideration for post-processing inexact results and describing some specific query constraints. We maintain that the heterogeneity of information sources inevitably renders the query translation inaccurate, and that post-processing of results is necessary to make up for the inaccuracy.

In chapter 4 and chapter 7, we will see that some filters ("Common filters" and "Special filters", see Definition 4.3 and Definition 4.4 in chapter 4) cannot be applied to result post-processing due to extraordinary processing costs or impossibility. Such filters will be skipped and the results of a transformed query with "skipped" filters will inevitably have differences with the results of the original query. Therefore, for some queries, "perfect" (100%) translation cannot be achieved because some constraints cannot be precisely translated. In [CGM00], Chang and Garcia-Molina presented a mechanism for approximately translating Boolean query constraints across heterogeneous information sources, which adopts a customizable "closeness" metric for the translation that combines both precision and recall.

2.2.3 Source selection

Many efforts have been devoted to source selection problems, such as [GGMT94, CLC95, GGM95, LPL96, DH97, YL97, DFGL98, MLY+98, Fuhr99, GGMT99, HT99, LXLN99, GIG01, IGS01, WMYL01, YMWL01, CGJM02, IG02, MYL02, CC03, GIS03]. Most of them provide automatic source selection by using source summaries and statistical information, such as Medoc Broker [DFGL98], GlOSS [GGMT94, GGM95, GGMT99], meta-index [DH97], and [CLC95, YL97, MLY+98, Fuhr99, MYL02]. Papers [WMYL01, YMWL01] present a method for ranking sources by employing the linkage information between documents to determine the degree of relevance of a document with respect to a given query.

In paper [DH97], Dreilinger and Howe use a meta-index approach which is a matrix (where the number of rows is the number of terms that have been used and the number of columns is the number of search engines). The meta-index tracks the effectiveness of each search engine in responding to previous queries. The meta-index grows as new terms are encountered. This method judges if a search engine will support a keyword based on experience. In D-WISE (Distributed Web Index and Search Engine) [YL97], the representative of a target search engine consists of the document frequency of each term in the database and the number of documents in the database. The representatives of all databases are used to compute the ranking score of each target search engine with respect to a user query. The ranking scores of this approach are relative scores that will be difficult to determine the real value of a database with respect to a user query. In the CORI Net (Collection Retrieval Inference Network) project [CLC95], the representative of a database consists of two pieces of information for each distinct term in the database: the document frequency and the number of databases containing the term. The ranking score of a database with respect to a user query is an estimated combined probability that the database contains useful documents due to each query term. The GlOSS (Glossary-of-Servers Server) project [GGMT94, GGM95, GGMT99] has presented a solution to the text-database discovery problem by integrating the indexes of all constituent databases to generate a meta-index. For each database and each word, the number of documents containing that word is included in the meta-index. When a query is submitted to GlOSS, relevant databases are selected by using the meta-index information. The GlOSS system demands that each of the remote search engines must cooperate with the meta-search engine by supplying up-to-date index information. Therefore, this system needs a prohibitive amount of administrative complexity. In [HT99, GIG01, IGS01, IG02, GIS03], techniques are proposed to automate the extraction of content summaries from "uncooperative" searchable text databases.

The above-mentioned approaches may be practical for relatively static sources, but they could be problematic for widely distributed, autonomous, and dynamic sources. Because the contents of a search engine's collection and index always change, yesterday a search engine may have found 0 hits for the term "Artificial Intelligence", and today it may add some new papers on AI, but the meta-search engine still excludes this search engine on the basis of the history records. However, the method of using experience can quicken source selection. For example, a search engine focusing on publications on mathematics cannot support some terms from chemistry fields. Therefore, a thesaurus can be built to record some often-used terms from various fields and map these terms to some search engines.

Another method is to provide users with complete control, so users can explicitly state which search engines are to be used. This will cause a problem: all users must be aware of the domains of expertise of each search engine.

Papers [CGJM02, CC03] propose an early user interaction approach that invites user collaboration in query formulation and query categorization. This approach can be viewed as a complement to existing collection selection techniques. Liu et al. [LPL96] proposed an approach for query routing that is similar to our approach. This approach is to map the domain model terminology to the source model terminology by applying a set of catalog mapping operators that utilize the metadata information maintained in the information source catalog and the interface repository. Our source selection method utilizes CSC (Classification Selection Control, see Definition 3.2 in chapter 3) information from the user interface and domain information about the integrated search engines. Therefore, users only have to enter information needs by manipulating CSC controls in the user interface of a meta-search engine, and the meta-search engine will automatically select target sources. Our source selection method not only can be used effectively in dynamic environments but also can overcome another deficiency of the statistics-based methods: sources with relevant documents are not searched if they are below thresholds.

2.2.4 Query user interface construction

Most current information integration systems only adopt simple, static query input interfaces [WHC+96, CYC+96, LP97, TRV97, SC97, HKWY97, HTRO99, ACPS96, KLSS95, BBB+97, LSH95, MKSI96, SAD+94, AKS96, GMHI+95]. There are some researches on designing more sophisticated user interfaces for integration information exploration [RRM93, GR94, BW97, CPW+97]. However, these systems have very limited support for coordinating various constraints among the controls on the user interfaces of heterogeneous information sources. Paper [BKP+98] describes an extensible constructor tool that helps information experts (e.g., librarians) create specialized query synthesizers for heterogeneous digital library environments. These query synthesizers can be used by end-users to specify queries. These manually created user interfaces can coordinate the constraints among the controls on the user interfaces of heterogeneous information sources, but they will inevitably consume much human resources. In paper [YLGMU99], Yerneni et al. presented algorithms to compute the set of mediator-supported queries based on the capability limitations of integrated target sources. Although this paper does not cope with the query input user interface constructing issues, the proposed method can be used to assist in constructing the user interfaces of information integration systems. Our SPOMSE meta-search engine employs adaptive, dynamically-generated query construction interfaces. Adaptive mechanisms will facilitate the reduction of constraints, and consequently make the query translation more accurate and easy. They can achieve such advantages as supporting the progressively self-refining construction of users' information needs; efficiently coordinate conflicts among heterogeneous sources; matching user queries to the queries supported by target sources as far as possible.

2.2.5 Others

Some researches [DM97, FLMS99, GMW99, IFF+99, GW00] have been conducted on combining the query facilities of traditional databases with existing Web search engines. They use wrappers to make information in Web pages act logically as database elements, and then enable the using of well-structured languages like SQL or OQL on the unstructured or semi-structured documents. These researches are not the focus of this thesis.

Some researches [CHS+95, ACPS96, TRV96, HKWY97, TRV97, AFT98, NGT98, SV98, UFA98, VZ98, TRÖH99, IFF+99, CDTW00, LC00, BFPV01, ZRZB01, ZRV+02] cope with problems of cost-based query optimization for accessing heterogeneous information sources. These kinds of cost-based query optimization problems are very important for large-scale information integration systems. Query optimization methods can be employed by meta-search engines and information integration systems to improve the efficiency of query evaluation and result post-processing. Cost-based query optimization is not the focus of this thesis.

2.3 Meta-search engines

In section 1.1.3, we discussed some deficiencies of WWW search engines. In this section, we discuss several general-purpose meta-search engines, such as SavvySearch [24], AskJeeves [25], MetaCrawler [26], Dogpile [27], Highway 61 [28], I.SEE [29], etc. These meta-search engines can more or less overcome such deficiencies. Figure 2.1 and Figure 2.2 display the query input interfaces of these meta-search engines.

SavvySeach (see Figure 2.1(a)) searches several hundreds of search engines, Web directories, auctions, storefronts, news sources, discussion groups, and reference sites. It permits users to customize their favorite query interface, such as selecting some search engines and customizing their ranking. **AskJeeves** (see Figure 2.1(b)) is a question answering meta-search engine that supports plain English queries. Unlike the traditional search engines that only use keywords to match documents, it tries to understand the user by presenting users with one or more closely related questions to which it knows the answer. When users input a question or some keywords to "AskJeeves", it will suggest some similar questions that other people have often asked and for which the system has the correct answers. **MetaCrawler** (see Figure 2.1(c)) [SE95a, SE95b] was originally developed at the University of Washington. MetaCrawler queries other search engines, organizes the results into a uniform format, ranks them by relevance and returns them to the user.

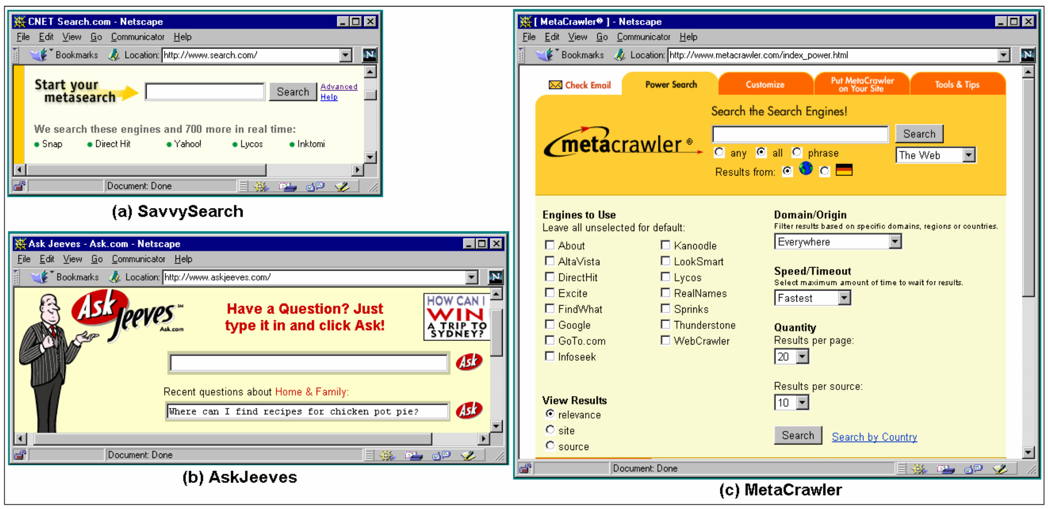


Figure 2.1 Query pages of SavvySearch, AskJeeves, MetaCrawler

Dogpile (see Figure 2.2(a)) supports many of the Web's most popular search tools, such as search engines, Usenet, FTP, Stock Quotes, Yellow Pages, Auction, etc. It also permits users to set searching order and choose some specific categories. **Highway 61** (see Figure 2.2(b)) supports "link verification". Because the Web grows and changes so fast, many URLs may disappear or the pages change days after they are indexed by a search engine. Therefore, it benefits users if a meta-search engine can verify whether the URLs are still valid. Unfortunately, this will greatly influence the response time of meta-search engines. "Highway 61" provides several levels of user patience with interesting labels: (1) "Hurry up! You losers!" (2) "Please try and make it quick." (3) "I'm a reasonable person." (4) "Time is a relative thing …" (5) "Take your time, I'm going to the bathroom…". **I.SEE** (see Figure 2.2(c)) [ABT97] offers a uniform interface that resembles the interfaces provided by the search engines. The difference is that the I.SEE interface combines many of the capabilities available from different engines. It also provides a passage to any number of search engines selected by the user. I.SEE contains explicit pull-down menus for specifying different query options such as case sensitivity, category, resource type, etc. This interface contains an amalgamation of the options supported by the search engines.

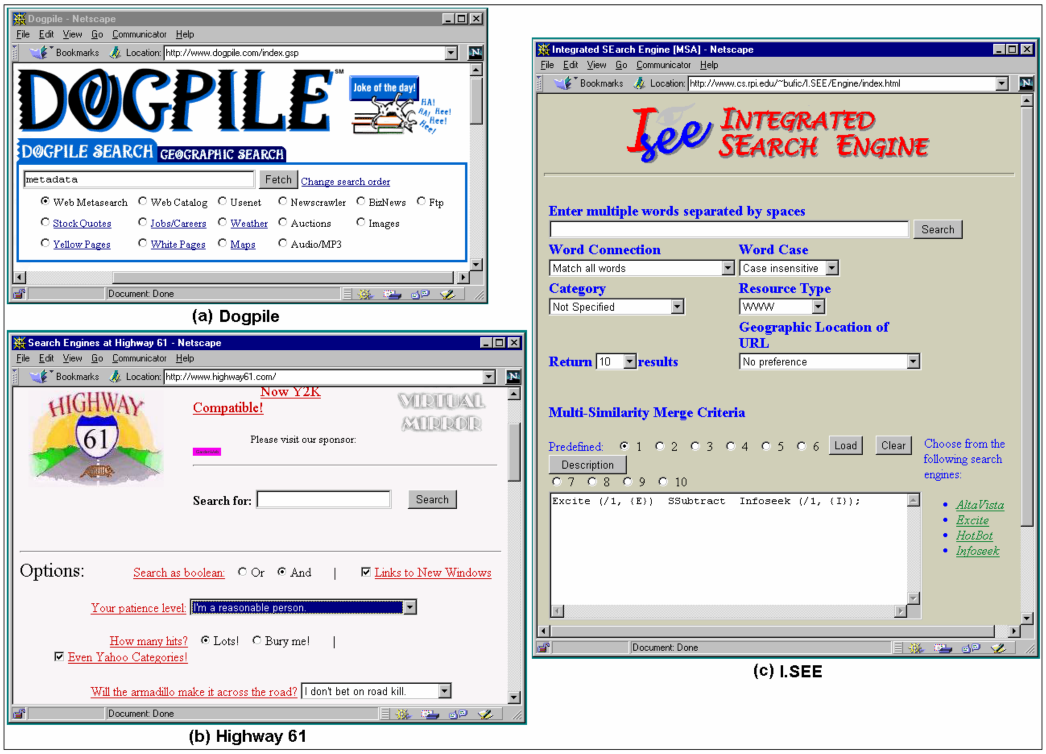


Figure 2.2 Query pages of Dogpile, Highway 61, I.SEE

The above-mentioned meta-search engines visit multiple remote search engines in parallel and merge the results in a uniform style. Each of them has its own features. For example, the AskJeeves focuses on natural question answering. Some other meta-search engines focus on processing retrieved results. For example, the Vivísimo meta-search engine (http://vivisimo.com) developed by Carnegie Mellon University integrates and categorizes textual information on-the-fly into meaningful hierarchical folders. Therefore it is also called a document clustering engine.

There are other meta-search engines, such as “Cyber411” [45], “Internet Sleuth” [46], “ONESEEK” [47], etc. Although these meta-search engines integrate a lot of WWW search engines, their user interfaces only use a simple user interface (an input-box for query input; some search engines may have other controls for users to select domain, search engines, result size, and patience level, etc.) and discard some of the rich functionalities of specific search engines. It is difficult for users to input complicated queries and retrieve specific information. This weakness is especially obvious when users want to search for specialized information, such as scientific information. In addition, today's meta-search engines do not have adaptive mechanism for differing domains and query capabilities.

Our SPOMSE meta-search engine prototype provides users with an adaptive, dynamically generated user interface through control constraint rules (see Chapter 5). Compared with the simple user interface adopted by the above-mentioned meta-search engines, the adaptive, dynamically-generated user interface can achieve higher precision. In addition, a constraint-based query translation algorithm (see Chapter 4) has been employed in our system to coordinate the functional disparities between the meta-search engine and the target search engine. In this way, the meta-search engine can utilize the query capabilities of target search engines as much as possible.

2.4 Invisible Web catalogues

In the last section, we have introduced several general-purpose meta-search engines. They visit multiple Web search engines and integrate the results. Some of them also visit local databases. In this section, we introduce some special-purpose Web information source catalogues. They provide a uniform query interface for searching specialized databases. Figure 2.3 displays the cropped screenshots of four Invisible Web catalogues: **Infomine Multiple Database Search** [30], **Lycos Invisible Web Catalog** [31], **AlphaSearch** [32], and **WebData** [33]. These catalogues serve as database portals, specializing in finding, categorizing and organizing online databases, resource collections, electronic journals and books, online library card catalogs, and directories of researchers. They only provide a simple query interface for users to search "hidden" databases. People can benefit from their efforts in gathering and classifying the information sources on the Web. However, the query construction functionalities and the query translation capabilities of these catalogues are weak and they have no mechanisms for differing query capabilities of target sources. Our SPOMSE meta-search engine prototype overcomes the shortcoming of weak query input functionality by providing users with an adaptive, dynamically-generated query construction interface. Furthermore, it can utilize the query capabilities of target information sources to a most extent by employing a constraint-based query translation algorithm.

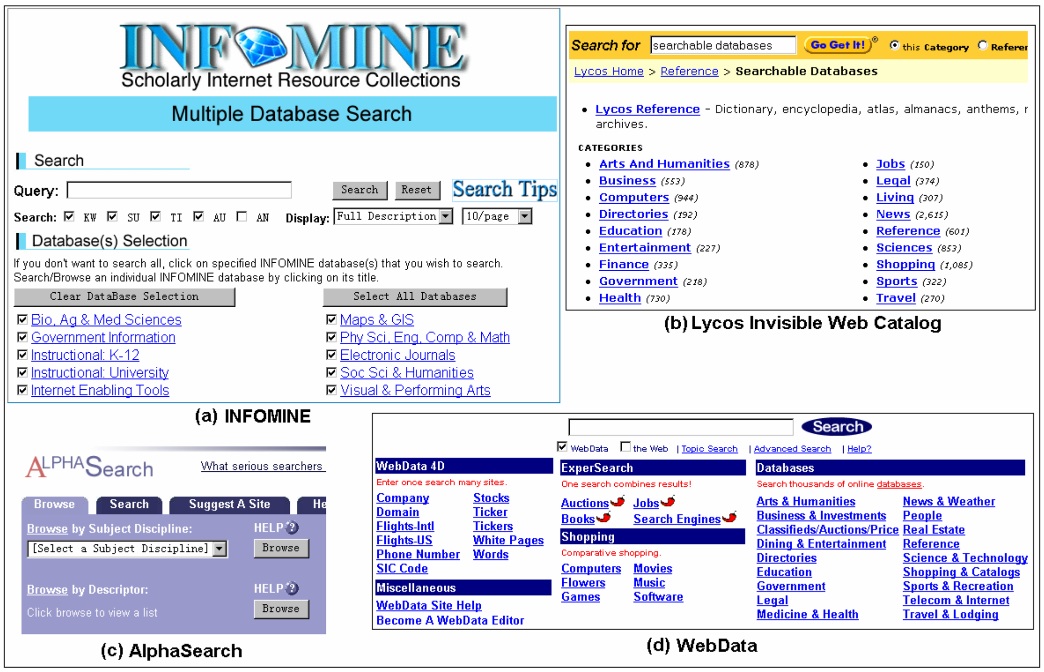


Figure 2.3 Search pages of some Invisible Web Catalogs

2.5 Standards and protocols

There has been considerable effort towards defining all kinds of standards for supporting and improving meta-search on the Web, such as Z39.50[Z95], ISO8777[ISO93], STARTS[GC+97], CCL[Neg79] and later Z39.58[Z93], CORBA [34], etc. Of course, if all people would compose documents using a standard format (e.g., XML [35] and RDF [36]) and all search engines would adopt a uniform interface and query model (e.g., Z39.50), constructing an information integration system would be very easy. However, for a number of reasons, such as a large amount of legacy information, information producers' unwillingness to comply with strict rules, online businesses might prefer to be visited manually rather than mechanically (for example, some search engines delivering advertisements to users try every means to avoid their information being automatically extracted and searched by software agents), and great differences from one domain to another, these standards are not being applied extensively. As a consequence, there is great diversity in the user interfaces and query models of information sources for different domains or subjects that is very difficult to unify.

Meta-search engines can be built by utilizing some attribute sets (such as the metadata elements of Dublin Core, Z39.50-1995 Bib-1 and GILS attribute set), standards of query languages (e.g., the type-101 query of the Z39.50-1995 standards, STARTS protocol) and some information formats (e.g., Harvest SOIFs).

Z39.50 Protocol

Z39.50 [Z95] specifies a client/server-based protocol for Information Retrieval. It specifies procedures and structures for a client to search a database provided by a server, retrieve database records identified by a search, scan a term list, and sort a result set. Access control, resource control, extended services, and a "help" facility are also supported. The protocol addresses communication between corresponding information retrieval applications, the client and server (which may reside on different computers); it does not address interaction between the client and the end-user.
This standard fully specifies and mandates support of the type-1 query, expressed by individual search terms, each with a set of attributes, specifying, for example, type of term (subject, name, etc.), whether it is truncated, and its structure. The server is responsible for mapping attributes to the logical design of the database. The attributes associated with a search term belong to a particular attribute set, whose definition is registered, that is, assigned a unique and globally recognized attribute-set-id, an Object Identifier, which is included within the query. The attribute-set bib-1 of Z39.50 specifies various attributes useful for bibliographic queries. Additional attribute sets may be registered outside of the standard. This standard is helpful for distributed information retrieval. If an information source adopts Z39.50 standard, the information integration system will be able to easily and efficiently utilize the query capabilities of this source.

GILS

The goal of the Global Information Locator Service (GILS) is to make it easy for people to find information of all kinds, in all media, in all languages, and over time. GILS adopts existing open standards that achieve this interoperability by allowing reference to common semantics for characterizing information resources. The GILS Profile provides the specifications for the overall GILS application relating to the GILS Core, which is a subset of all GILS Locator Records, and completely specifies the use of Z39.50 in this application. Sensitive to the world's many languages, as well as legal and financial issues, GILS adopts the ANSI Z39.50 standard to specify how electronic network searches can be expressed and how results are returned. GILS-compliance is a particular way in which servers support searching for the characteristics of any kind of information, at any level of aggregation [Chri96].

STARTS

STARTS [GCGMP96] is a protocol proposed by Stanford University for Internet retrieval and search. The goal of STARTS is to facilitate the main three tasks that a meta-searcher performs: choosing the best sources to evaluate a query; evaluating the query at these sources; merging the query results from these sources. STARTS specifies a query language that is based on a simple subset of the Z39.50-1995 type-101 query language and the GILS attribute set. A query consists of two parts: a filter expression and a ranking expression. A filter expression is Boolean in nature and defines the documents that qualify for the answer. The ranking expression associates a score with these documents and ranks them accordingly.

The Dublin Core and the Warwick Framework

The Dublin Core (DC for short) is a metadata element set intended to facilitate discovery of electronic resources. Originally conceived for author-generated description of Web resources, it has attracted the attention of formal resource description communities such as museums, libraries, government agencies, and commercial organizations (http://purl.oclc.org/dc/). DC is an attempt to formulate a simple yet usable set of metadata elements to describe the essential features of networked documents. Up to now, DC defines fifteen elements: Title, Creator, Subject, Description, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, and Rights.

"The Warwick Workshop was convened to build on the Dublin Core program and provide a more concrete and operationally usable formulation of the Dublin Core, in order to promote greater interoperability among content providers, content catalogers and indexers, and automated resource discovery and description systems."[LLD96] The result of this workshop is a proposal for a container architecture: the Warwick Framework. The framework is a mechanism for aggregating logically, and perhaps physically, distinct packages of metadata. It allows the designers of individual metadata sets to focus on their specific requirements, without concerns for generalization to ultimately unbounded scope.

XML / RDF

"The eXtensible Markup Language -- XML -- is a simple dialect of SGML whose goal is to enable generic marked-up documents to be handled on the Web as easily as HTML." [37] HTML is a layout language for presenting textual documents whereas XML is a meta-language just like SGML for the structure and semantics of information. "XML specifies neither semantics nor a tag set. In fact, XML is really a meta-language for describing markup languages. In other words, XML provides a facility to define tags and the structural relationships between them. Since there's no predefined tag set, there can't be any preconceived semantics. All of the semantics of an XML document will either be defined by the applications that process them or by stylesheets." [38] Just like Alon Levy [Levy99] pointed out: "XML without agreed upon DTDs does nothing to support integration at the semantic level. The names and meanings of the tags used in XML documents are arbitrary. As a result, the emergence of XML is fueling activity in various communities to agree on DTDs". Data-exchange standards such as XML and other approaches (e.g., OntoSeek [GMV99], On2broker [FAD+99], etc.) will simplify the information extraction and integration, but they force the data consumer to accept the producer's ontological decisions and the entire collection of Web pages must be transformed into ontological form. Such problems will require meta-searchers to translate between different DTDs. Now more and more people have used XSchema [48] to express constraints on XML documents, instead of using DTD to do that. Although XSchema has many advantages (such as the same syntax with XML, extensibility, supporting inheritance, and so on) over DTD, meta-searchers are still required to translate between different XSchemas.

The Resource Description Framework (RDF) is the W3C (The World Wide Web Consortium [39]) recommendation for defining the architecture necessary to support the interchange of Web metadata; it enables automated processing of Web resources. It accommodates the diversity of semantics and structure needed by various communities. Typical applications of RDF are resource discovery, search engines, catalogs and taxonomies, agents, digital signatures, and annotations. RDF uses XML as the encoding syntax for metadata. But the RDF model is independent of XML (RDF data may or may not be stored as XML; it may be directly stored in a DB as DB tables or in files as binary streams).

Dienst Protocol

The Dienst project [LSDK95] is a conceptual architecture for digital libraries, a protocol for communication in that architecture, and software system providing Internet access to distributed, decentralized document collections. Dienst is based on a document model that incorporates unique document names, multiple document formats, and multiple document decompositions. Interoperability among Dienst servers provides the user with a single logical document collection, even though the actual collection is distributed across multiple servers. The NCSTRL search engine [9] uses this protocol.

Harvest

The Harvest system [BDM+95] includes a set of tools for gathering and accessing information from heterogeneous sources, building and searching indexes of this information, and replicating the information throughout the Internet. The Harvest gatherers collect and extract indexing information from one or more sources. Then, the brokers retrieve this information from one or more gatherers, or from other brokers. The brokers provide a querying interface to the gathered information.

The above-mentioned standards and protocols require that all the distributed sources are to be constructed as homogeneous as possible. While in our meta-search engine, all integrated remote information sources (search engines, digital libraries, repositories, semi-structured document collections, etc.) are completely autonomous and the meta-search engine only uses the publicly accessible information (query input pages and output pages) of these sources.

Chapter 3

ADMIRE Data Model

Confronted with the great heterogeneity of information sources on the Internet, it is very important and critical to design an efficient data model for meta-search engines (or information integration systems) to describe the data structures, user interfaces, query capabilities, and domain knowledge of information sources. Considering the mutability and the fast growth of Internet information sources, and in order to accommodate the frequent emergence of new search engines and the updating of old ones, the data model should be sufficiently flexible and scalable.

This chapter proposes a data model that can be employed to formally describe the query capabilities and query input user interfaces of search engines. This chapter is organized as follows. First, in section 3.1, we briefly discuss the heterogeneity in data sources on five aspects (i.e., syntactic, semantic, content, capability, interface). Then in section 3.2, we present the ADMIRE data model that can be employed by meta-search engines and information integration systems to describe the user interfaces, the various constraints existing among them, and the query capabilities of Web search engines. Then in section 3.3, we use this data model as an inter-lingua to describe the user interfaces and query capabilities of three search engines and one meta-search engine. Finally, we conclude this chapter with some closing remarks.

3.1 Heterogeneity in information sources

The efficiency of a meta-search engine or an information integration system depends on the extent to which various conflicts existing among heterogeneous information sources are coordinated. In the following, we will discuss five kinds of data conflicts from the syntactic, semantic, content, query capability, and user interface aspects.

3.1.1 Syntactic conflicts

There are many syntactic conflicts such as the following. This kind of conflicts can be solved by using conversion functions and mapping tables.

(1) Data formats. For example, when different search engines return their results, we can see that they use different *date* formats. Some systems use "*September 8, 2000*", others use "*09/08/00*", "*08/09/00*", "*08.09.00*", "*Sep. 2000*", or "*08092000*", etc. Another example is the *Author* format. Some systems use full names, such as "Bill Gates", other user "B. Gates", "Gates, Bill", or "Gates, B.", etc.

(2) Labeling. Different systems use different names for synonyms. For example, some systems use "*all fields*" to denote this field modifier, some use "*anywhere*". For the title field modifier, Yahoo uses "*t:*", while Cora uses "*title:*". Some use standard names and some use abbreviations (e.g., "*kilometer*"/"*km*", "*Article title*"/"*Title*", etc.).

(3) Organization (or layout) of controls in the user interface or organization of fields in the results pages.

(4) Range of terms. In the user interfaces of different sources, the restrictions for each input term are different. For example, some terms can only be limited by a field (e.g., see Figure 1.2, in the second query input form of NCSTRL, there are three input-boxes with each belonging to a certain field: *author, title, or abstract*), some terms can be limited by a subset of all fields (e.g., see Figure 1.3, the query input page of ACM-DL has five check boxes for users to select several restricting fields for the input terms), while some terms can be limited by all fields (e.g., Elsevier service).

3.1.2 Semantic conflicts

There are several kinds of semantic conflicts such as:

(1) Field naming. This is different from the labeling conflicts. For example, in Dublin Core, there is only one metadata for authors: "*Creator*". While in USMARC [Cra84] (MARC is the abbreviation of Machine-Readable

Cataloging), there are two metadata for authors: "*Corporate author*" or "*Individual author*". Therefore, when translating a query from a Dublin Core compatible search engine to a USMARC compatible search service, there needs to be coordination between these two "*Author*" fields.

(2) Units&Scaling. Different information retrieval systems have different ranking methods. For example, ACM-DL may assign the value 11 to an entry. Cora search engine may assign 0.9156 to another entry. How can you compare the relevance of these two entries? In section 6.3, we will discuss some problems on result-merging.

3.1.3 Content conflicts

There are some content conflicts such as:

(1) Domain. There are many disciplines, subjects, topics and branches. For example, ACM-DL and NCSTRL only provide publications on computer science. While IDEAL and Elsevier provide publications on various disciplines. Some specialized search engines only focus on the document collections of very specific subjects, such as "Machine Learning", "UML", or the homepages of computer scientists, and so on.

(2) Languages.

(3) Genre. Technical reports, conference papers or posters, journal articles, theses, etc.

(4) Document formats.

(5) Quality. Peer reviewed material, newsletters, etc.

3.1.4 Capability conflicts

There are some capability conflicts such as:

(1) Different retrieval models and query languages. Some sources support Boolean-based queries, some support vector-space-based queries, some support natural language queries, some support probabilistic model based queries.

(2) Rich controls. Some sources provide rich functions for users to formulate queries, such as all kinds of term modifiers, logical operators, etc.
(3) Indexed term processing. Some sources automatically drop stop-words (e.g., and, with, etc.). Some sources support fuzzy expansion, stemming, right-/left-truncation, or wildcards, and so on.
(4) Performance. Some sources have faster response and more indexed documents than other ones.
(5) Quality level. Some sources only provide bibliographic information, some provide abstracts of publications, some provide full papers, and some provide review information for publications.
(6) Payment/rights/security services.

3.1.5 Interface conflicts

There are some interface conflicts such as:

(1) UI programming. Some sources provide static HTML form user interface, some provide dynamic HTML form user interface, some provide HTML form user interface with JavaScript, some provide java applet user interface, and some serve as application software installed on the clients' PCs.
(2) Personalization. Some sources provide customizing services for users to personalize their user profiles.
(3) Controls realization. Different sources use different ways to implement the same concept. For example, ACM-DL uses check boxes to implement field modifiers (see Figure 1.3), the IDEAL search service uses pull-down menus to implement them (see Figure 1.4), while NCSTRL uses labels to implement them (see Figure 1.2).
(4) Retrieval process. Some sources can return all results for a user query, while some sources require users to visit their web sites more than once to get complete results. Some sources can let users refine their queries after results come.

3.1.6 How can we solve the heterogeneity problems?

In the above, we have described some concrete conflicts existing among distributed, heterogeneous search engines from five aspects. Besides, there are many other kinds of conflicts that will not be discussed in this dissertation. Other researchers have already dealt with the problems of semantic interoperability in information sources, such as [HM85, HK87, SL90, HM93, KS94, AKS96, KM96, MGKS96, PGMU96, Gal99]. Their methods focused mainly on the domain and schema coordination in traditional multi-databases and federated or cooperative information systems. However, for a Web meta-search engine, the target search engines are completely autonomous and their internal database schemas and indexing structure and algorithms are invisible to outsiders. Therefore, the designers of meta-search engines can only utilize the information on the user interfaces of target search engines.

In this chapter, we will discuss and tackle the problem of describing various kinds of control constraints existing on the user interfaces of heterogeneous search engines, so as to enable meta-search engines utilize the query capabilities of target search engines as much as possible.

The way to solve these conflicts is to reduce or eliminate conceptual and terminological confusion and reach a shared understanding. Therefore, a unifying conceptual framework needs to be developed to describe the characteristics (user interfaces and query capabilities) of heterogeneous search engines. In the following section, we present the ADMIRE data model that provides a vocabulary of terms and their definitions (see Definitions 3.1-3.25) from domain and function points of view.

3.2 ADMIRE: an Adaptive Data Model for Integrating Retrieval Engines

In this section, we introduce the ***ADMIRE*** data model – an ***A***daptive ***D***ata ***M***odel for ***I***ntegrating ***R***etrieval ***E***ngines. First, in section 3.2.1, we analyze the user interfaces of search engines and divide the controls in user interfaces into three groups (i.e., classification selection controls, result display controls, and query input controls) by functionalities. Then in sections 3.2.2-3.2.4, we discuss these three groups of controls separately and provide basic definitions. Based on these basic definitions, in section 3.2.5, we introduce some advanced features of search engines, and in section 3.2.6, we provide the definition of ***query expression***.

3.2.1 Analysis of User Interfaces of Information Sources

In the following we analyze the user interface of an information source in order to find some commonalities among heterogeneous sources, and provide some formal definitions for the basic components in user interfaces.

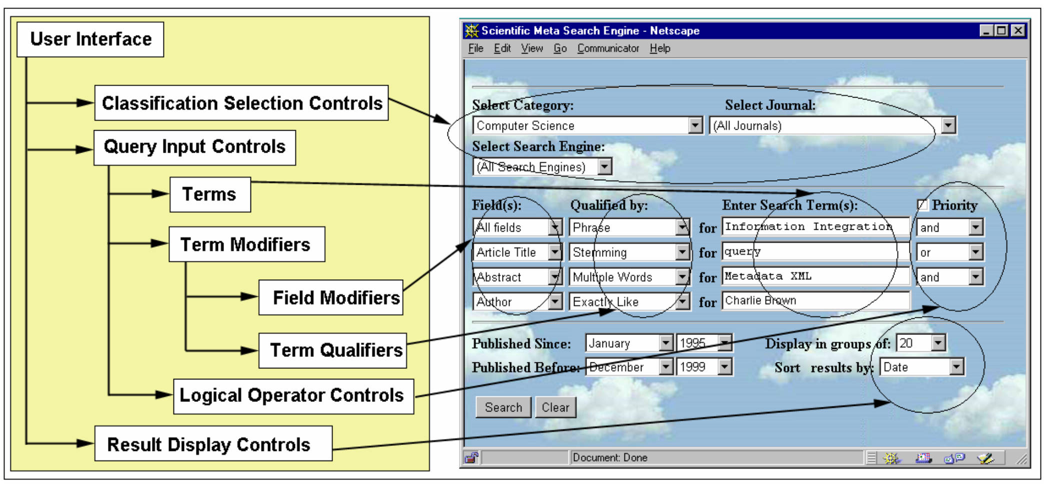


Figure 3.1 Analysis of the user interface of an information source

Although there are a lot of differences among user interfaces to search engines, we can generally divide the controls on user interfaces into three groups: **classification selection controls**, **result display controls**, and **query input controls** (see Figure 3.1).

Each group can also be divided into several sub-groups or types. For example, in Figure 3.1, there are three types of classification selection controls on the upper part of the query input page. They are used for selecting categories, journals, and search engines, respectively. On the bottom part of this page, there are two types of result display controls for users to select grouping size and sorting criteria. On the middle part of this page, we can also see that there are four types of query input controls (i.e., terms, field modifiers, term qualifiers, and logical operator controls).

The classification of the above-mentioned controls on the query input page of a search engine is based on the application of "Taxonomy" methods. Taxonomy (from Greek, *taxis* meaning arrangement, order or division and *nomos* meaning law) is the science of classification according to a pre-defined system, with the resulting catalog used to provide a conceptual framework for discussion, analysis, or information retrieval.

In Definition 3.1, we provide two basic definitions: "Concepts" and "Global Taxonomy". Although there are many similarities between "Global Taxonomy" and "Global Schema" (or "Global View") mentioned in other literature, some differences exist between them. In a traditional multi-database system, a single global schema is employed to encompass the differences among the schemas of all target local databases. In a federated database system, there are coordination and communication among multiple local databases. However, for a Web meta-search engine, all target search engines and information sources are completely autonomous. When new search engines are to be integrated into the meta-search engine, or integrated search engines change their user interfaces or schemas, the "Global Taxonomy" of the meta-search engine has to change.

Definition 3.1 Concepts and Global Taxonomy (**GlobalTaxonomy**)

When designing a meta-search engine or an information integration system, people need a globally predefined taxonomy to classify the user interface controls and the items of these controls of all candidate target search engines. The labels of these controls and items are called **concepts**. These concepts are hierarchically organized and separated into subgroups that are mutually exclusive, unambiguous, and taken together, comprehensive. Many controls will evolve and change during the development of a meta-search engine (such as adding new search engines, removing obsolete search engines, search engines being updated, and so on). Therefore, these taxonomies are evolving and dynamic. We call such a global, predefined but dynamic classification a **GlobalTaxonomy**.

The concepts of a GlobalTaxonomy are hierarchically organized like a rooted tree (see Figure 3.2). For a special-purpose meta-search engine, the concepts are domain oriented. For example, a scientific publication oriented meta-search engine has concepts including “Author”, “Abstract”, “Citation”, “Editors”, etc. But a meta-search engine for vehicle searching has concepts including “Cylinder”, “Model”, “Maker”, etc.

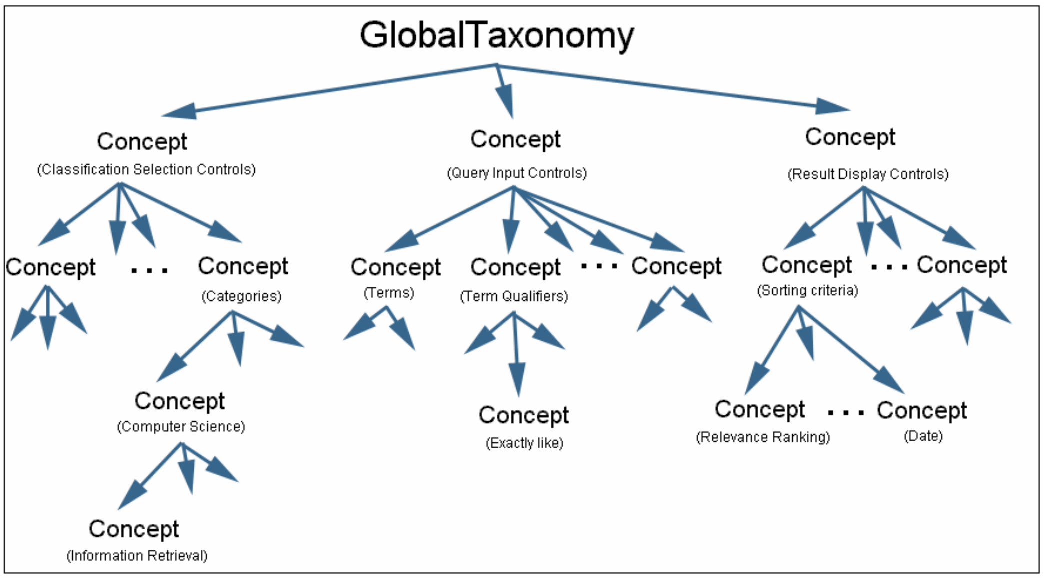


Figure 3.2 Example of a GlobalTaxonomy

The classification systems in library catalogues such as the Dewey Decimal Classification System [40] and the Library of Congress Subject Classification System [41] are already well-defined. Furthermore, such structured domain information and knowledge as the ACM Computing Classification System [42], AMS Mathematics Subject Classification [43] and other ontologies and taxonomies can be incorporated in any system to help users in their searches.

In the following sections, we will provide formal definitions for the controls on the query input page in Figure 3.1. In these definitions, the phrase “search tool” is employed to mean various search engines, meta-search engines, or information sources. First, we introduce classification selection controls.

3.2.2 Classification Selection Controls

Definition 3.2 Classification Selection Control (**CSC**) and CSC Items (**CSCItems**)

A classification selection control (**CSC**) is a control component on the user interface of a search tool. By selecting one or more items (called **CSCItems**) of a CSC, users can limit their information needs to certain information domains, subjects, categories, etc. The values of CSCItems cannot be modified by users.

Definition 3.3 CSC Types (**CSCTypes**) and CSC Taxonomy (**CSCTaxonomy**)

In a meta-search engine and its integrated search engines, there are several types of CSCs (called **CSCTypes**).
CSCTypes = $\{CSCType_i\}$, where ($1 \le i \le N_{CSCTypes}$, $N_{CSCTypes}$ is the number of CSCTypes)

According to the GlobalTaxonomy, each type of CSC contains a group of concepts that compose the taxonomy of this CSC (called **CSCTaxonomy**).
For each $CSCType_i$, there is a corresponding $CSCTaxonomy_i$.

Definition 3.4 Classification Selection Controls Set (**CSCSet**)

All CSCs of a search tool constitute its CSC set (called **CSCSet**):
CSCSet = $\{CSC_i\}$, where ($1 \le i \le N_{CSCs}$, N_{CSCs} is the number of CSCs)

Each CSC belongs to a CSCType (supposed to be $CSCType_k$) and contains several CSCItems. Each of these CSCItems corresponds to a concept in the corresponding $CSCTaxonomy_k$.
$CSC_i = \{CSCItem_{i,j}\}$, where ($1 \le j \le N^i_{CSCItems}$, $N^i_{CSCItems}$ is the number of CSCItems in CSC_i)

In the following, we provide a concrete example for a CSCSet of a scientific publication oriented meta-search engine. This CSCSet contains 4 CSCs, i.e., CSCSet = {CSC_1, CSC_2, CSC_3, CSC_4} = {*Category CSC*, *Journal CSC*, *Search Engine CSC*, *Language CSC*}.

In a *meta-search* engine, the number of CSCs usually equals the number of CSCTypes, i.e., each CSC belongs to a CSCType. However, a *target search* engine may only contain some types of CSCs and each CSC may only contain part of the CSCItems of the corresponding CSCType.

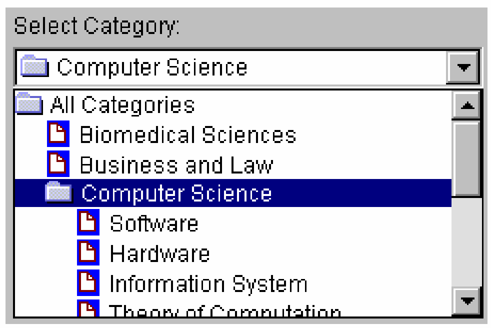


Figure 3.3 an example of a category CSC and its CSCItems

Figure 3.3 illustrates the CSC_1, i.e., the Category CSC of this CSCSet: CSC_1 = *Category CSC* = {$CSCItem_{1,1}$, $CSCItem_{1,2}$, $CSCItem_{1,3}$, $CSCItem_{1,4}$, $CSCItem_{1,5}$, $CSCItem_{1,6}$, … …} = {"All Category", "Biomedical Sciences", "Business and Law", "Computer Science", "Software", "Hardware", … …}. From this figure, one can recognize that the CSC_1 is implemented in the user interface of a search tool by using a tree control component in which its CSCItems are hierarchically organized. In Definitions 3.5-3.7, we will discuss tree-structured CSCs in detail. Therefore, by using such data structures, the information about the hierarchical organization of CSCItems can be recorded internally.

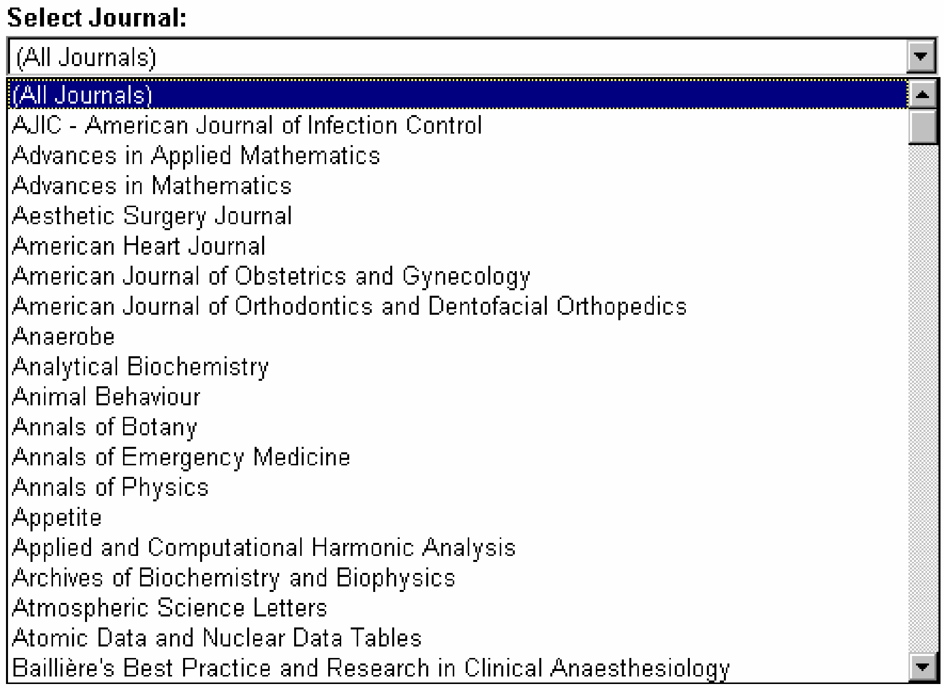


Figure 3.4 an example of a journal CSC

Figure 3.4 displays an example of a Journal CSC of the IDEAL search service (Figure 1.4 displays the whole query page). From this figure, one can recognize that the CSCItems of this CSC is organized as a one-dimensional list instead of a tree. However, this Journal CSC is internally recorded as a simple tree structure, i.e., a parent node (All Journals) with its children nodes.

CSCs can be used by users to control their search by specifying their domain-specific information needs, search goal and preferences.

As mentioned earlier, for a domain-rich meta-search engine, the CSCItems of some CSCs can be hierarchically organized like a tree. In the following, we will provide the definition of “TreeCSC” and discuss some problems of constructing TreeCSCs for meta-search engines from the corresponding TreeCSCs of target search engines.

Definition 3.5 Tree-structured Classification Selection Control (**TreeCSC**)

A **TreeCSC** is a CSC whose CSCItems are hierarchically categorized as the nodes of a rooted tree according to the structure of its corresponding CSCTaxonomy. Edges connecting CSCItems denote specialization. An edge from CSCItem N_1 to another (lower-level) CSCItem N_2 denotes that N_2 is a sub-concept (e.g., sub-category, sub-domain, sub-discipline, sub-subject, etc.) of N_1.

A TreeCSC consists of a root CSCItem with k (k≥0) **SubTreeCSCs**. The syntax of a TreeCSC can be recursively defined as follows:

TreeCSC ::= <CSCItem> | <CSCItem> {IncludeOrNot SubTreeCSCs }
SubTreeCSCs ::= (, TreeCSC)*
IncludeOrNot ::= +|–

where

'*' refers to zero or more contiguous occurrences of a given pattern.

'|' denotes "Alternative".

"IncludeOrNot = +" means that the search tool user interface supports classification selections on the parent CSCItem of the corresponding SubTreeCSCs, i.e., the search tool itself has the corresponding CSCItem in its CSCTaxonomy.

"IncludeOrNot = –" means that the search tool user interface does not support classification selections on the parent CSCItem of the corresponding SubTreeCSCs, i.e., the search tool itself does not have the corresponding CSCItem in its CSCTaxonomy.

Figure 3.5(a) displays a concrete example of a TreeCSC. In Figure 3.5(b), it is described according to the syntax defined in Definition 3.5. Figure 3.3 is the cropped screenshot of the implementation of this TreeCSC. There are no '–' signs for any CSCItems in this TreeCSC.

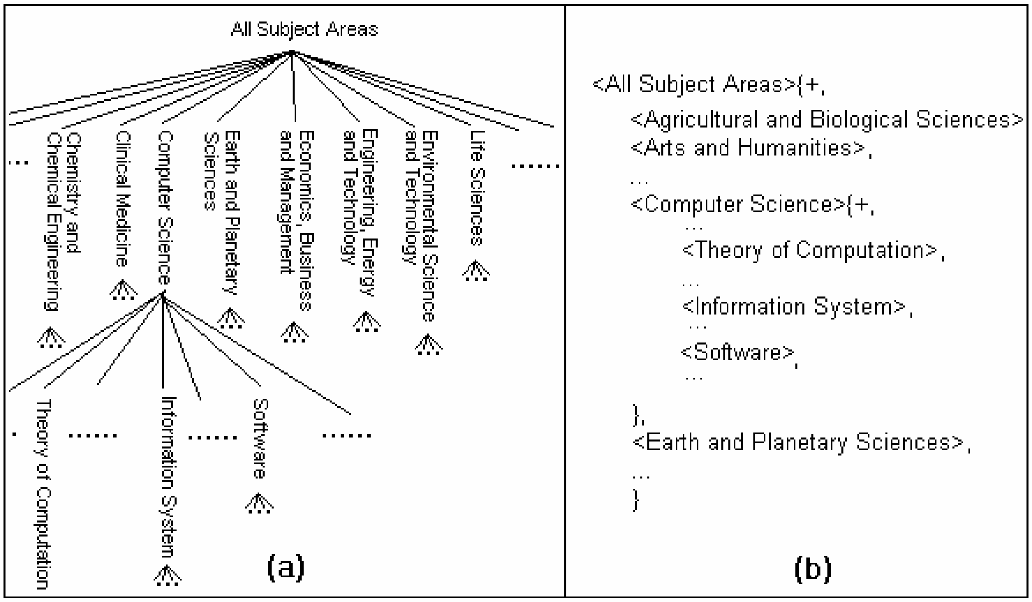


Figure 3.5 an example of a category TreeCSC with '+' signs

Next we provide another example of a TreeCSC that has '–' signs in its CSCItems. Figure 3.6(a) displays a cropped screenshot of a CSC on the user interface of a search engine. This CSC has only two CSCItems: <Computer Graphics> and <Database>. According to the global taxonomy, there will be a CSCItem <Computer Science> that is the parent CSCItem of the above-mentioned two CSCItems. But this CSCItem <Computer Science> cannot be supported by this search engine. Therefore we can virtually construct the TreeCSC like Figure 3.6(b). Figure 3.6(c) displays the syntax of this TreeCSC. Later we will continue to discuss this example. In the following, before we provide the definition of "Virtual CSCItems" in Definition 3.7, we first provide the definition of "Ancestor and Descendant CSCItems" in Definition 3.6.

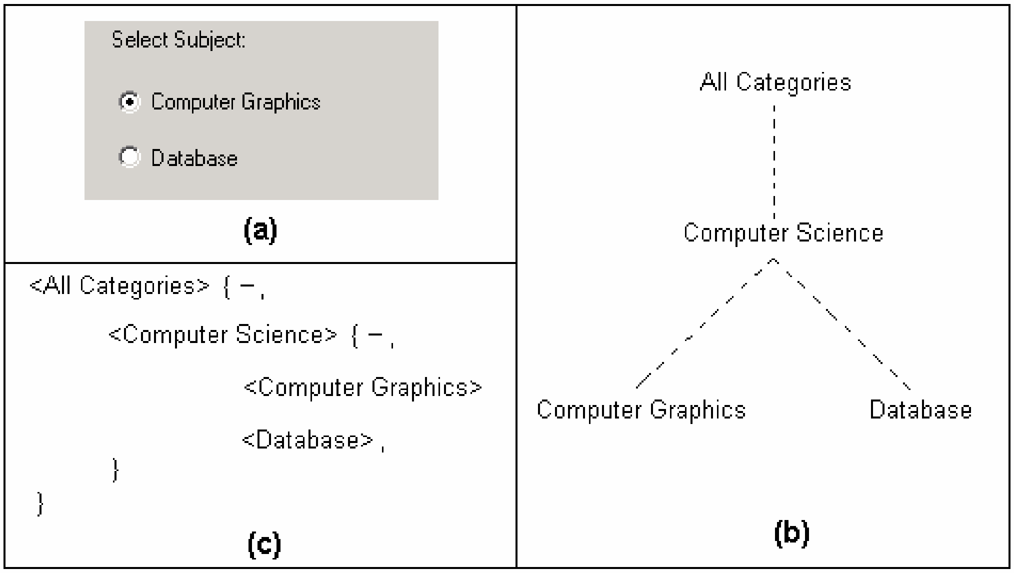


Figure 3.6 an example of a category TreeCSC with '–' signs

Definition 3.6 Ancestor and **Descendant CSCItems**

In a TreeCSC, CSCItem A is an **ancestor CSCItem** of CSCItem B if A has a higher position than B and there is a direct path (not passing through a CSCItem higher than or on the same level with CSCItem A) from A to B. At the same time, CSCItem B is called a **descendant CSCItem** of CSCItem A.

A TreeCSC of a meta-search engine can be regarded as the union of relevant TreeCSCs of all integrated target search engines. Figure 3.7(a) illustrates the construction of a meta-search engine's TreeCSC. The left side of Figure 3.7(a) displays five TreeCSCs (i.e., $STree_1$, $STree_2$, $STree_3$, $STree_4$, and $STree_5$) of different search engines. The middle part of Figure 3.7(a) is the synthesized TreeCSC (named "Mtree") of a meta-search engine and it can be regarded as a CSCTaxonomy of all the integrated search engines. The right part of Figure 3.7(a) displays how the TreeCSC of the meta-search engine is constructed by the five TreeCSCs of search engines. Before explaining Figures 3.7(b) and 3.7(c), we first provide a definition of "Virtual CSCItems" and an operational definition of "normalizing TreeCSCs of search engines".

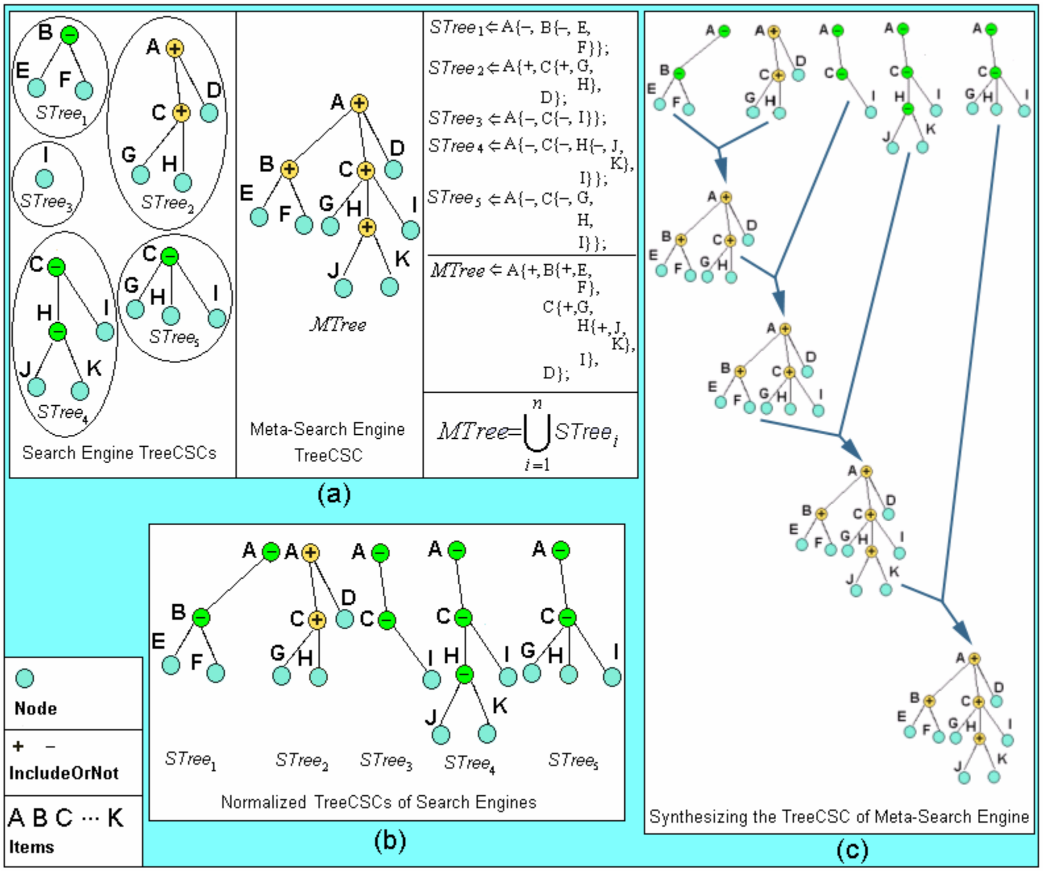


Figure 3.7 Construction of the meta-search engine's TreeCSC

Definition 3.7 Virtual CSCItems and **Normalization of TreeCSCs of search engines**

Suppose that the i^th TreeCSC of the meta-search engine (M) is $TreeCSC_{M,i}$, and the j^th target search engine's corresponding k^th TreeCSC is $TreeCSC_{Sj,k}$. $TreeCSC_{M,i}$ and $TreeCSC_{Sj,k}$ belong to the same CSC type $CSCType_l$. When $TreeCSC_{Sj,k}$ is to be integrated into the meta-search engine, according to the corresponding $CSCTaxonomy_l$, $TreeCSC_{Sj,k}$ may be lacking some ancestor CSCItems due to its incompleteness. We call these lacking ancestor CSCItems "**Virtual CSCItems**" of $TreeCSC_{Sj,k}$, each of which will be labeled with a '–' sign (see Definition 3.3). We refer to such a process that adds virtual CSCItems as "**normalizing TreeCSCs of search engines**" (see Figure 3.7(b)).

We continue to use the previously-discussed example of a category TreeCSC (see Figure 3.6) to illustrate these two definitions. Suppose that the category TreeCSC of search engine S_j can only support queries on the categories of “Computer Graphics” and “Database”, but it does not support the “Computer Science” category (see Figure 3.6(a)). According to the predefined taxonomy of the meta-search engine for this TreeCSC, there will be a virtual parent CSCItem “Computer Science” with a number of child CSCItems that include “Computer Graphics” and “Database”. So in this case, this virtual CSCItem “Computer Science” of S_j will have a ‘–’ sign. In the same way, another virtual CSCItem “All Categories” will be added as the parent CSCItem of the virtual CSCItem “Computer Science” in this search engine according to the predefined taxonomy. This CSCItem will also be labeled with a ‘–’ sign. The original TreeCSC ({<Computer Graphics >, < Database >}) will be normalized as a new TreeCSC (with two virtual CSCItems):
<All Categories> {–, < Computer Science > {–, <Computer Graphics >, < Database >}}.

The above example has also been illustrated in Figure 3.7. The TreeCSC $STree_1$ has two CSCItems (E, F). It is normalized as (A {–, B {–, E, F}}) according to the global taxonomy.

Figure 3.7(b) displays five normalized CSCtrees of the five original target sources' TreeCSCs. We can see that some virtual ancestor CSCItems will be added to the TreeCSC of each search engine according to the global taxonomy of the meta-search engine. Figure 3.7(c) displays the method of synthesizing these five normalized TreeCSCs into the TreeCSC of the meta-search engine. Table 3.1 displays the algorithm (in pseudo-code) for constructing the TreeCSCs of the meta-search engine.

Table 3.1 Meta-search engine's TreeCSC construction algorithm

Input: *TreeCSCs of all target search engines*
Output: *TreeCSCs of the meta-search engine*
For ($i = 1$; $i<=N_{CSCs}$; i++) //Suppose there are N_{CSCs} TreeCSCs in the meta-search engine
$TreeCSC_{M,\ i} = \varnothing$; //Set empty value to the i^{th} TreeCSC of the meta-search engine
For ($j = 1$; $j <=n$; j++) //Suppose there are n target search engines
Normalize j^{th} search engine's k^{th} $TreeCSC_{Sj,\ k}$ (that corresponds to the $TreeCSC_{M,\ i}$) by adding virtual ancestor CSCItems;
$TreeCSC_{M,\ i}$ += $TreeCSC_{Sj,\ k}$;

Note: here "+=" is an overridden function for integrating a TreeCSC (i.e., $TreeCSC_{Sj,\ k}$) of a target search engine into the meta-search engine's TreeCSC (i.e., $TreeCSC_{M,\ i}$), i.e., merging these two TreeCSCs by removing the duplicate CSCItems and inserting the new CSCItems of $TreeCSC_{Sj,\ k}$ into $TreeCSC_{M,\ i}$.

When two or more TreeCSCs of different search engines are synthesized into a TreeCSC of the meta-search engine, the virtual ancestor CSCItems (with sign '–') will be materialized (with sign '+') in the meta-search engine by changing sign '–' to sign '+'. For example, the 'B' CSCItem in Figure 3.7 is a virtual CSCItem in the $TreeCSC_1$ and is not in any other TreeCSCs, but when it is synthesized with other TreeCSCs, the CSCItem will be assigned the sign '+'. This means that users can formulate queries that have the selection of this 'B' CSCItem of this meta-search engine. When a user formulates a query that has the selection of the 'B' CSCItem, this query will be translated into two sub-queries; with one selecting the 'E' CSCItem and the other selecting the 'F' CSCItem. These two sub-queries will then be submitted to the target source.

After integrating all target search engines into a meta-search engine, a non-leaf CSCItem of a TreeCSC in the meta-search engine contains all its CSCItems that correspond to the concepts in the global taxonomy. However, it may not contain all possibilities of the real world domain as expressed in the corresponding part of the global taxonomy. For example, suppose in the realistic situation, the 'B' CSCItem has four child CSCItems: 'B_1', 'B_2', 'B_3', and 'B_4'. But there are only two CSCItems 'B_1' and 'B_2' in the target search engines. So in the meta-search engine, 'B' CSCItem only displays these two child CSCItems. We can use visual metaphors to express the "incompleteness" of the 'B' CSCItem, for example, using a folder icon with unsaturated color. Therefore, when users see this icon, they can understand that the meta-search engine only supports part of this category. The reason is that all the integrated target search engines cannot support this entire category, i.e., the entire sub-domain of the global taxonomy.

In order to record the information of each integrated target search engine for query routing and source selecting, a source mapping table (see Definition 3.8) is created.

Definition 3.8 Source Mapping Table

Source mapping table can be used to record the mapping situations between the CSCItems of a meta-search engine's CSCs and the z (z>0) integrated search engines. $\forall CSCItem\ I \in CSC_i$, $\exists k(z \geq k \geq 1)$ search engines that support *I*.

For example, the CSCItem <Zoological Journal of the Linnean Society> in Journal CSC can only be mapped to the IDEAL search engine; while the CSCItem <Computer Sciences> in Category CSC can be mapped to most search engines that have been integrated in our scientific publication-oriented meta-search engine. Table 3.2 demonstrates how a source mapping table looks like. The left two columns in this table specify the CSCs and their CSCItems of the meta-search engine. Each row of the right three columns in this table specifies the information of a CSCItem (and wich search engine and CSC it belongs to) that corresponds to one CSCItem of the meta-search engine. In chapter 4, we will introduce a source selection algorithm by using this source mapping table. In chapter 5, when discussing the adaptive user interface construction for meta-search engines, we will also touch upon the functionality of this table.

Table 3.2 A source mapping table example

Meta-search engine		**Target search engines**		
CSCs	**CSCItems**	**Search Engines**	**CSCs**	**CSCItems**
CSC_1	$CSCItem_{1,1}$	S_1	CSC_1	$CSCItem_{1,2}$
		S_2	CSC_3	$CSCItem_{3,5}$
		S_5	CSC_2	$CSCItem_{2,1}$
		…	…	…
	$CSCItem_{1,2}$	S_2	CSC_1	$CSCItem_{1,2}$
	…	…	…	…
CSC_2	$CSCItem_{2,1}$	S_1	CSC_3	$CSCItem_{3,1}$
		S_3	CSC_2	$CSCItem_{2,2}$
	…	…	…	…
…	…	…	…	…

3.2.3 Result Display Controls

When users submit a query to a search engine, the search engine will return a number of results (hits) to the users' computer. Perhaps the number of the returned results is more than ten thousand or even a million. Users do not want all the results to be listed in a page; so they hope that the search engine can only return 10 or 20 hits each time when they send an additional request. This will lower the network traffic and users' waiting time. In addition, the investigation on the log files of user queries of a large search engine [SHMM99] shows that most users are only looking at the first 10-20 hits. Of course, users hope these 10-20 hits will contain what they need most. Therefore, the search engine can sort the results according to users' sorting criteria, such as the results can be sorted by the relevance to the input keywords or by dates from newer to older, and so on. Users may also hope that the results can be clustered by subjects or authors from a publication-oriented search engine. Some users hope that each hit is returned with detailed description; while other users only need brief description or the URL to a Web page (or a link to a file).

Definition 3.9 Result Display Control (**RDC**) and RDC Item (**RDCItem**)

A result display control (**RDC**) is a control component on the user interface of a search tool. By selecting items (called **RDCItem**) of a RDC, users can control the formats, sizes or sorting methods of the query results. The RDCItems of one RDC are a one-dimensional set that cannot be modified by users.

Definition 3.10 RDC Types (**RDCTypes**) and RDC Taxonomy (**RDCTaxonomy**)

In a meta-search engine and its integrated search engines, there are several types of RDCs (called **RDCTypes**).

RDCTypes = $\{RDCType_i\}$, where ($1 \le i \le N_{RDCTypes}$, $N_{RDCTypes}$ is the number of RDCTypes)

According to the GlobalTaxonomy, each type of RDC contains a group of concepts that compose the taxonomy of this RDC (called **RDCTaxonomy**).

For each $RDCType_i$, there is a corresponding $RDCTaxonomy_i$.

Definition 3.11 Result Display Controls Set (**RDCSet**)

All RDCs of a search tool compose the RDCs set (called **RDCSet**):

RDCSet = $\{RDC_i\}$, where ($1 \le i \le N_{RDCs}$, N_{RDCs} is the number of result display controls)

Each RDC belongs to an RDCType (suppose it to be $RDCType_k$) and contains several RDCItems. Each of these RDCItems corresponds to a concept in the corresponding $RDCTaxonomy_k$.

$RDC_i = \{RDCItem_{i,j}\}$, where ($1 \le j \le N^i_{RDCItems}$, $N^i_{RDCItems}$ is the number of RDCItems in the ith RDC)

In the following, we provide a concrete example for an RDCSet of a scientific publication oriented meta-search engine (see Figure 3.8). This RDCSet contains 3 RDCs, i.e., RDCSet = $\{RDC_1, RDC_2, RDC_3\}$ = {*Sorting Criteria RDC*, *Grouping Size*

RDC, *Description RDC*}. RDC is often implemented by using check boxes, radio-boxes, or pull-down menus.

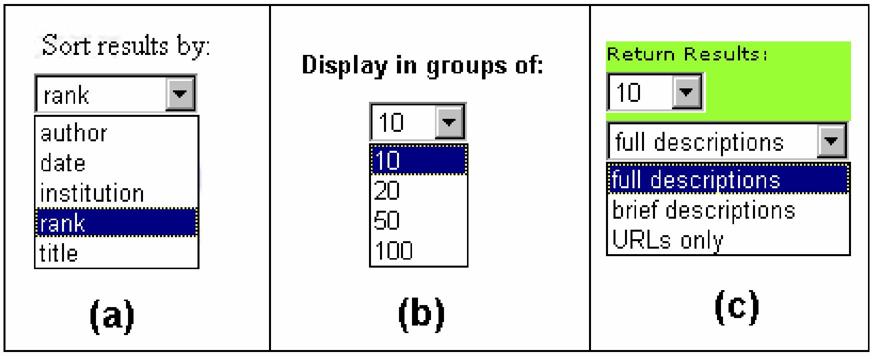


Figure 3.8 RDC examples

Figure 3.8(a) illustrates a choice control that lets users select one of the methods for sorting results. RDC_1 = *Sorting Criteria RDC* = {$RDCItem_{1,1}$, $RDCItem_{1,2}$, $RDCItem_{1,3}$, $RDCItem_{1,4}$, $RDCItem_{1,5}$} = {<Author>, <Date>, <Institution>, <Relevance ranking>, <Title>}. As for the <Relevance ranking> in the Sorting Criteria RDC, because each search engine has its own algorithm for computing relevance, we cannot rearrange all items when merging results from various search engines. In section 6.3, we will discuss some issues on result merging, such as result sorting, duplicate removing and results dynamic displaying.

Figure 3.8(b) illustrates a choice control that limits the number of returned hits per page. RDC_2 = *Grouping Size RDC* = {$RDCItem_{2,1}$, $RDCItem_{2,2}$, $RDCItem_{2,3}$, $RDCItem_{2,4}$, $RDCItem_{2,5}$} = {<10>, <20>, <30>, <50>, <100>}.

Figure 3.8(c) illustrates a choice control that decides in which format each hit will be displayed. RDC_3 = *Description RDC* = {$RDCItem_{3,1}$, $RDCItem_{3,2}$, $RDCItem_{3,3}$} = {<full>, <brief>, <URL>}.

3.2.4 Query Input Controls

In the above two subsections, we have discussed “Classification Selection Controls” and “Result Display Controls”. These two kinds of controls are for users to express their information needs on domain aspect and their favorite results display styles. In this subsection, we will discuss some controls that are for users to express their concrete and specific information needs from the lexical and semantic point of view.

From Figure 3.1, we can see that "Query Input Controls" (QIC) consists of three parts: "Terms", "Term Modifiers", and "Logical Operator Controls". In the following, we will introduce them respectively. Finally, we will provide the definition for "QICs".

3.2.4.1 Terms

Definition 3.12 Term (**T**) and Terms Set (**TSet**)

A term (**T**) is the content keyed into an input box on the user interface of a search tool, which is different from the usual meaning of "term" because a term defined here can be a single keyword, multiple words, a phrase, or a Boolean expression. In some cases, the input term may support truncation or stemming*. It may be case-sensitive, and might drop stop-words, hyphens, diacritics and special characters.

All terms of a search tool compose the term set (called **TSet**):
TSet = $\{T_i\}$, where ($1 \le i \le N_{Ts}$, N_{Ts} is the number of terms)

*Note: Truncation of a word "abc" means it matches any words starting with this truncation "abc". Stemming of a word "abc" means it matches any words with the same stem as "abc" under some stemming algorithm [Lovi68, Port80]. Some information retrieval systems do not use stemming algorithms when indexing documents.

In order to better illustrate the concept of terms and compare the terms of different search engines, we extract all terms from the three search engines (see Figures 1.2, 1.3, 1.4) introduced in Chapter 1, and generate a new figure as Figure 3.9.

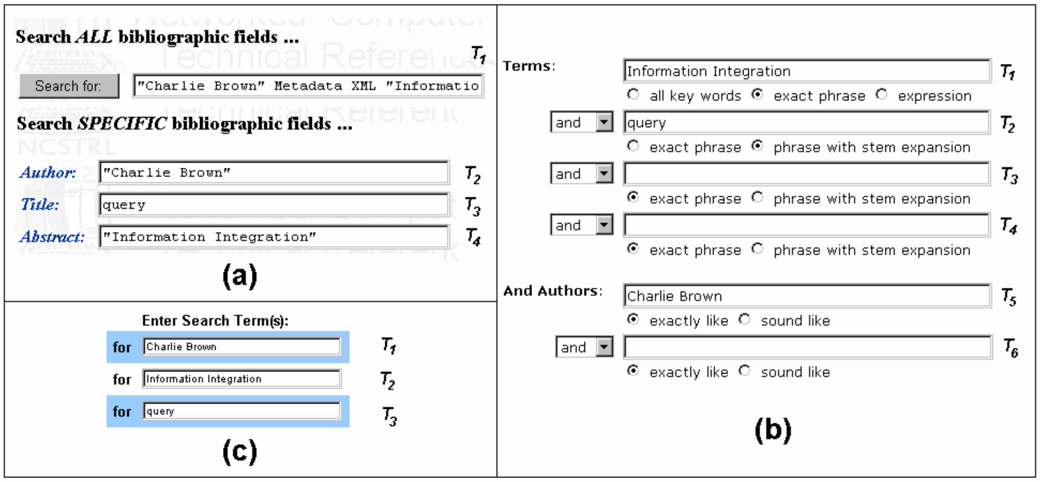


Figure 3.9 Terms examples

Figure 3.9(a) shows that the NCSTRL search engine provides four terms $\{T_1, T_2, T_3, T_4\}$; the first term and the other three terms, respectively, belong to two separate forms. In this figure, each term has concrete value:

- T_1 = "Charlie Brown" Metadata XML "Information Integration" query;
- T_2 = "Charlie Brown";
- T_3 = query;
- T_4 = "Information Integration";

From Figure 3.9(b), we can see that ACM-DL has six terms $\{T_1, \ldots, T_6\}$. In the first four terms, users can input keywords and phrases, and in the last two terms, users can input authors' names.

From Figure 3.9(c), we see that the IDEAL Search provides three terms $\{T_1, T_2, T_3\}$, in which users can input any keywords, phrases, and authors' names.

In addition, a term can be a Boolean expression, for example, in the advanced query page of the AltaVista search engine (see Figure 1.6), the first term T_1 = ("Charlie Brown" AND (("Information Integration" AND query) OR (Metadata AND XML)));

After the discussion of terms, now we discuss term modifiers. From Figure 3.1, we can see that "Term Modifiers" consists of two parts: "Field Modifiers" and "Term Qualifiers". In the following, we will discuss them separately.

3.2.4.2 Field Modifiers

Definition 3.13 Field Modifier (**FM**), Field Items (**FieldItems**) and Field Modifiers Set (**FMSet**)

A field modifier (**FM**) is a control component on the user interface of a search tool. By selecting one or more items (called **FieldItems**) of an FM, users can limit the scope of a term, i.e. it requires that the provided term be contained in the appointed part of the result. "Fielded search" usually means that keywords provided by users should be found in certain parts of a publication.

All field modifiers of a search tool compose the field modifiers set (called **FMSet**):
FMSet = { FM_i }, where ($1 \leq i \leq N_{FMs}$, N_{FMs} is the number of field modifiers)

Each field modifier contains one or more FieldItems. A FieldItem is also called a "field" for short.
$FM_i = \{FieldItem_{i,j}\}$, where ($1 \leq j \leq N^i_{FieldItems}$, $N^i_{FieldItems}$ is the number of FieldItems in the i^{th} field modifier)

Because each field modifier is associated with a term, we also use "Field(T_i)" to denote one concrete selected *field item* from the "FM_i", i.e., Field(T_i) $\in FM_i$.

In the scientific publication domain, we can stipulate the field items of one field modifier as follows:
$FM_i \subseteq \{FieldItem_{i,1}, FieldItem_{i,2}, FieldItem_{i,3}, \dots \dots\}$ = {<Title>, <Full-Text>, <Review>, <Article Keywords>, <Abstract>, <Author>, <Affiliation>, <Date>, <ISBN>, <ISSN>, <Journal Title>, <Citation>, <Editor>, <Anywhere>}

Figure 3.10 displays three concrete examples of field modifiers in different scientific publication oriented search engines.

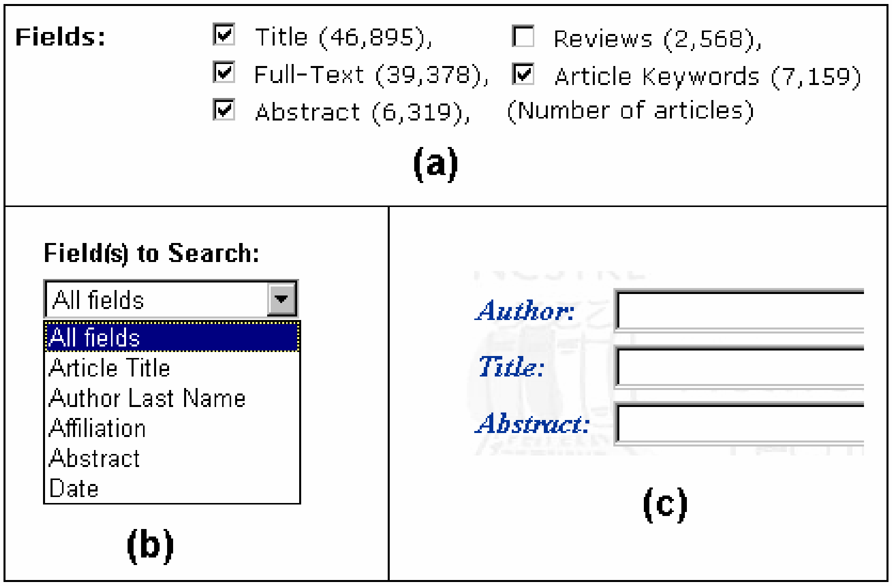


Figure 3.10 various implementations of field modifiers

In Figure 1.3 (also Figure 3.10(a)) we can see that the first four terms in the user interface of ACM-DL can be limited by arbitrary fields (Subset of five fields, i.e., FM_1, FM_2, FM_3, FM_4 $\subseteq$ {<Title>, <Full-Text>, <Review>, <Article Keywords>, <Abstract>}); while the fifth term and the sixth term can only be modified by the <Author> field (i.e., FM_5, FM_6 = {<Author>}).

In Figure 1.2 (also Figure 3.10(c)), in the first querying form of NCSTRL, the term can be limited by several fields, i.e., FM_1 $\subseteq$ {<Author>,<Title>,<Abstract>}; while in the second form, each term can only be limited by a certain field (FM_2 = {<Author>}; FM_3 = {<Title>}; FM_4 = {<Abstract>}).

Figure 1.4 (also Figure 3.10(b)) shows that each term of IDEAL Search can be limited by one of {<Title>, <Abstract>, <Author>, <Affiliation>, <Date>} or all these 5 fields. FM_1, FM_2, FM_3 $\in$ {{<Title>}, {<Abstract>}, {<Author>}, {<Affiliation>}, {<Date>}, {*<Title>, <Abstract>, <Author>, <Affiliation>, <Date>*}}. The terms in this search engine cannot be limited by an arbitrary subset of all fields (e.g., {<Title>, <Abstract>}); while ACM-DL can.

The field modifiers of different search engines can be implemented by using different methods, for example, by check boxes (ACM-DL, see Figure 3.10(a)), pull-down menus (IDEAL, see Figure 3.10(b)), or designated input-box (NCSTRL, see Figure 3.10(c)).

The above-mentioned field modifiers are in the scientific publication domain. There are also other field modifiers in other domains. For example, in the "Car selling" domain, fields can be "Maker", "Model", "Cylinder", "Door", etc. For the "Weather forecast" domain, the fields can be "Temperature", "Relative humidity", "Wind", etc.

3.2.4.3 Term Qualifiers

Definition 3.14 Term Qualifier (**TQ**), Qualifier Items (**QualifierItems**) and Term Qualifiers Set (**TQSet**)

A term qualifier (**TQ**) is a control component on the user interface of a search tool. By selecting one or more items (called **QualifierItems**) of a TQ, users can limit the quality and form of the term input by users.

All term qualifiers of a search tool compose the term qualifiers set (called **TQSet**): TQSet = { TQ_i }, where ($1 \le i \le N_{TQs}$, N_{TQs} is the number of term qualifiers)

Each term qualifier contains one or more QualifierItems.
$TQ_i = \{QualifierItem_{i,j}\}$, where ($1 \le j \le N^i_{QualifierItems}$, $N^i_{QualifierItems}$ is the number of QualifierItems in the ith term qualifier)

Because each term qualifier is associated with a term, we also use "Qualifier(T_i)" to denote one concrete selected *qualifier item* from the "TQ_i", i.e., Qualifier(T_i) $\in$ TQ_i.

In the scientific publication domain, we can stipulate the qualifier items of one term qualifier as follows:
$TQ_i \subseteq \{QualifierItem_{i,1}, QualifierItem_{i,2}, QualifierItem_{i,3}, \ldots \ldots\}$ = {<Exactly Like>, <Multiple Words>, <Using Stem Expansion>, <Phrase>, <Expression>, <Sound Like>, <Spelled Like>, <Before>, <After>}

For ACM-DL, users can select qualifiers from the query page (see Figure 1.3). The qualifier of the first term has three possible qualifier items, i.e., TQ_1 = {$QualifierItem_{1,1}$, $QualifierItem_{1,2}$, $QualifierItem_{1,3}$} = {<Multiple Words>, <Phrase>, <Expression>}; while each qualifier of the other five terms has two qualifier items, i.e., TQ_2, TQ_3, TQ_4 = {<Using Stem Expansion>, <Phrase>}, TQ_5, TQ_6 = {<Exactly Like>, <Sound Like>}.

For NCSTRL and IDEAL Search, there is no qualifier control for users to select. However, from the help files one can learn that the functions of qualifiers can be embodied in the terms. For example, the phrase qualifier can be applied by using quotation marks (“”); Wildcard signs like '*' and '?' can be used to express stemming expansion. We can define some rules in specific wrappers for translating query expressions. For some search engines, the qualifiers of some terms have the item <Expression>. In [CGMP99], there is a deep-going discussion about the conversion of qualifiers and qualified terms. However, it does not consider the constraints imposed by fields.

In the above, we have discussed field modifiers and term qualifiers. Our query translation algorithm proposed in chapter 4 will sufficiently consider and coordinate this kind of constraint information. In the following, we will discuss logical operators that will also be considered in query translation.

3.2.4.4 Logical Operator Controls

Definition 3.15 Logical Operator Control (**L**), Logical Operator (**θ**) and Logical Operator Controls Set (**LSet**)

A logical operator control (**L**) is a control component on the user interface of a search tool, one of which items (called a **logical operator** or θ; usually it can be ‘AND’, ‘OR’, or ‘AND NOT’) can be used to logically combine two terms to perform a search, the results of which are then evaluated for relevance.

All logical operator controls of a search tool compose the logical operator controls set (called **LSet**):
LSet = $\{L_i\}$, where ($1 \le i \le N_{Ls}$, N_{Ls} is the number of logical operator controls)

Each logical operator control contains one or more logical operators:
$L_i = \{\theta_{i,j}\}$, where ($1 \le j \le N^i_{LogicalOperators}$, $N^i_{LogicalOperators}$ is the number of LogicalOperators in the i^th^ logical operator control)

Usually, Boolean information retrieval systems allow 4 kinds of logical operators:
$L_i \subseteq \{\theta_{i,1}, \theta_{i,2}, \theta_{i,3}, \theta_{i,4}\} = \{\wedge, \vee, \neg, \sim\}$ where $\wedge$ means AND; $\vee$ means OR; $\neg$ means NOT; ~ means NEAR.

The <NOT> logical operator is not a unary operator, i.e., it cannot be used alone. It serves as a connector between two terms: A NOT B = A AND (NOT B).

Because the proximity operator <NEAR> is often used together with these Boolean operators by some information sources, in this thesis we treat <NEAR> like a boolean operator.

Each logical operator control has its own "sphere of action"; it can only join two terms. Some search engines have strict restrictions for logical operator controls. For example, the two logical operator controls of NCSTRL must have the same value and can only be "AND" or "OR" (see the second query form in Figure 1.2). This will greatly restrict the query translation process. We must decompose the original query expression into equivalent sub-query expressions or minimal subsuming query sub-expressions (see Chapter 4). In [CGMP96], there is a detailed discussion about the general transformation of users' queries into a subsuming query by the conversion of DNF (Disjunctive Normal Form) and CNF (Conjunctive Normal Form). However, it does not cope with the constraints of fields and qualifiers on the terms.

Definition 3.16 Query Input Controls (**QICs**)

All terms (TSet), field modifiers (FMSet), term qualifiers (TQSet), and logical operator controls (LSet) of a search tool constitute a group of **query input controls** (QICs), through which users can express their information needs (queries).

$QIC_i \in$ {TSet $\cup$ FMSet $\cup$ TQSet $\cup$ LSet};

QICs = {QIC_i},

where ($1 \le i \le N_{QICs}$, N_{QICs} is the number of query input controls, $N_{QICs} = N_{Ts} + N_{FMs} + N_{TQs} + N_{Ls}$).

//TSet and N_{Ts}(see Definition 3.12), FMSet and N_{FMs}(see Definition 3.13),

//TQSet and N_{TQs}(see Definition 3.14), LSet and N_{Ls} (see Definition 3.15).

In an adaptive, progressive user interface, the number of query input controls may change during users' querying. A Boolean retrieval model based search engine usually has richer QICs than a vector space retrieval model based search engine.

Definition 3.17 Controls Set (**CTRLSET**)

All controls on the user interface of a search tool constitute a controls set:

CTRLSET = CSCs+QICs+RDCs.

The meta-search engine should describe the information of target search engines as much as possible. However, it does not mean that all controls will be displayed on the user interface of the meta-search engine; the controls in the user interface of a meta-search engine should be organized just like a common search engine. In chapter 5, we will discuss how to construct the user interface of a meta-search engine. Based on the above description, a meta-search engine can utilize the capabilities of specific search engines as much as possible.

3.2.5 Advanced Features

In sections 3.2.2-3.2.4, we have discussed basic components in the query user interfaces of Web search engines. However, such controls in a query user interface are not applied in an isolated, independent way. In this section, we discuss some advanced features that are derived from the structural relationships of all controls. By combining single controls, we can get useful information for the construction of meta-search engines.

First, we provide a definition of "Boolean Expression" that can be extracted from a group of mutually-relevant controls in the QICs.

Definition 3.18 Boolean Expression (**Exp**)

A Boolean expression ***Exp*** is constructed by k (k≥1) terms combined by (k-1) logical operators. $Exp = T_1\theta_1T_2\theta_2\ldots\theta_{k-1}T_k$, where

- Field(T_i) $\in$ FM_i; 1≤i≤k;
- Qualifier(T_i) $\in$ TQ_i; 1≤i≤k;
- $\theta_j \in L_j$; 1≤j≤(k–1).

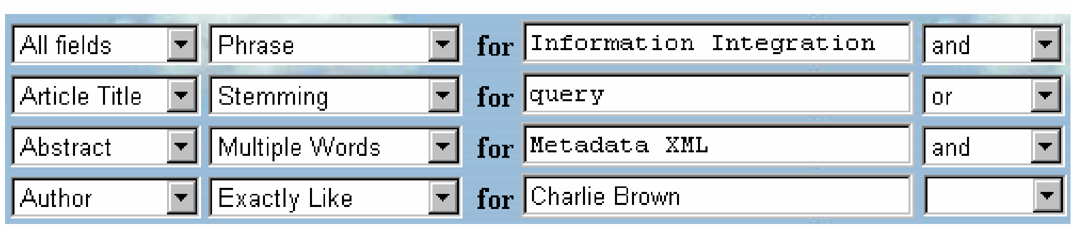


Figure 3.11 an example of Boolean expression

The query example in Figure 3.11 can be considered as a Boolean expression. Here we let

- Term A as (*Phrase "Information Integration" can be found anywhere*);
- Term B as (*Article title contains "query" or its stemming form*);
- Term C as (*Abstract contains two single words "Metadata" and "XML"*);

- Term D as (*Authors contain "Charlie Brown"*).

So this Boolean expression is (A and B or C and D). How does a meta-search engine or a search engine explain this expression? Does it equal (((A and B) or C) and D)? Or does it equal ((A and B) or (C and D))? Different systems have different explanations. In the following, we will discuss them separately.

Definition 3.19 Sequential Boolean Expression (**SEQ**)

Some search engines interpret the Boolean expression *Exp* by order. If *Exp* = $(T_1\theta_1T_2\theta_2\ldots\theta_{k-1}T_k) = ((\ldots(T_1\theta_1T_2)\ \theta_2T_3)\ldots)\theta_{k-1}T_k)$, we call the Boolean expression *Exp* a *Sequential Boolean Expression* denoted by ***SEQ***(*Exp*).

Some information sources support sequential Boolean expressions; while in other Boolean systems, some logical operators have precedence over other logical operators. For example, 'AND' has higher priority than 'OR'. Now we provide the definition of "Priority Operators".

Definition 3.20 Priority Operators

We use the $\prec$ sign as a priority operator, and $\alpha\prec\beta$ denotes that α has precedence over β. We can also use the following signs to denote all other kinds of priority operators accordingly: $\succ$ (lower priority), $\preccurlyeq$ (higher or equal priority), $\succcurlyeq$ (lower or equal priority), $\nprec$ (not higher priority), $\nsucc$ (not lower priority).

Definition 3.21 Priority Boolean Expression (**PRI**)

Some search engines interpret a Boolean expression *Exp* by some stipulated priority rules, such as $\wedge\prec\vee$, $\neg\prec\vee$, $\sim\prec\vee$, $\wedge\nprec\neg$, Etc. In this case we call the Boolean expression *Exp* a *Priority Boolean Expression* denoted by ***PRI***(*Exp*).

For example, if the query in Figure 3.11 is interpreted by priority: and $\prec$ or, this Boolean expression ***PRI*** (A and B or C and D) = ((A and B) or (C and D)).

Because both vector-space model and probabilistic model (see section 2.1) are similar in the query input (just a list of keywords without logical operators), we use the same definition VSQ to express the query capabilities of such information retrieval systems.

Definition 3.22 Vector-Space Query Expression (**VSQ**)

We call a query expression Exp_i supported by a vector-space model (or probabilistic model) based search engine as *Vector Space Query* denoted by ***VSQ***(Exp_i).

This kind of expression is quite simple (no strict syntax and no sophisticated features) and is usually just a list of keywords, for example: ***VSQ***("Intelligent Information Retrieval").

Definition 3.23 Bracket Controls

Most current search engines cannot support complicated Boolean expressions. Boolean expression ((**A** *AND* **B**) *OR* (**C** *AND* **D**)) can be interpreted by priority, but not in sequential order. While ((**B** *OR* **C** *OR* **D**) *AND* **A**) can be interpreted in sequential order but not by priority. However, some queries cannot be expressed by the sequential and priority methods, e.g., ((**A** *OR* **B**) *AND* (**C** *OR* **D**)). Therefore, the system should provide a bracket mechanism so that users can generate such Boolean expressions by manipulating certain controls.

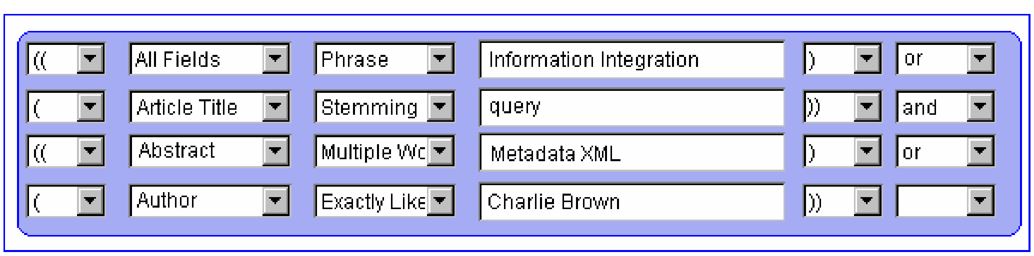


Figure 3.12 Brackets support

In Figure 3.12, the system provides several bracket controls for users to construct complicated Boolean expressions. Here (A or B and C or D) can be explained as ((A or B) and (C or D)) with the help of bracket controls.

One may think that such parenthetical expressions are unnecessary for experts who can type such expressions faster and with more flexibility than by using the bracket widgets on the query form. However, this is true only if the terms in a query expression have no limiting modifiers (such as <Author>, <Stemming>, etc); for example, in Figure 1.6, in the AltaVista search engine, users can type a complete query expression in a text-input box. But how do users express the following information in a box in Figure 1.6? The author's name is exactly "Charlie Brown", "query" or its stemming words must be in <Title> field, and so on. A constraints-coordinated search engine or meta-search engine should enable users to express specific information needs.

Definition 3.24 Special Constraints

A special constraint is an unusual situation that cannot be described by previous definitions (Definition 3.1-3.23). We use C_{FM}, C_{TQ}, C_θ to denote special constraints for field modifiers, term qualifiers, and logical operator controls.

There are very few such special constraints on the heterogeneous search engines. This kind of constraint is extremely rare on the Web. For example, there is only one case from about more than ten search engines we have investigated. From Figure 1.2, there is a special constraint for its logical operator controls in the NCSTRL search engine: C_θ is "$\theta_1 = \theta_2$". We have not found any special constraints on the field modifiers and term qualifiers. Because of the autonomy of distributed information sources, the owner of one source can design the user interface and data model in whatever way they want; and in order to achieve a perfect query translation between the meta-search engine and a target source, the data model should consider this kind of special constraints.

3.2.6 Query Expression

Up to now, we have defined and discussed the basic components and some advanced features in the user interfaces of search tools. Based on these definitions, we can formally describe the user interface and query capability of a search engine as "Query Expression" in Definition 3.25.

The query capability of a search tool can be expressed as the union of several Boolean (or VSQ) expressions, CSCs, and RDCs. In the following definition, we use the sign '$\cup$' to denote a set of several components of the same type (e.g., Boolean / VSQ expressions, CSCs, or RDCs).

Definition 3.25 Query Expression (**Q**)

A query expression ***Q*** can be denoted as:

$$Q = (\bigcup_{i=1}^{n} (\Phi_i (Exp_i (T_{i_1} \theta_{i_1} ... \theta_{i_{k-1}} T_{i_k})))) \cup (\bigcup_{j=1}^{u} CSC_j) \cup (\bigcup_{l=1}^{v} RDC_l) \text{, where}$$

- $n \geq 1$, $u \geq 0$, $v \geq 0$;
- $\Phi_i \in \{$***SEQ***, ***PRI***, ***VSQ***$\}$;
- Field(T_{i_l}) $\in FM_{i_l}$, $1 \leq i_l \leq i_k$, C_{FM};
- Qualifier(T_{i_l}) $\in TQ_{i_l}$, $1 \leq i_l \leq i_k$, C_{TQ};
- $\theta_{i_l} \in L_{i_l}$, $1 \leq i_l \leq i_{k-1}$, C_θ;
- $CSC_j \in$ CSCSet; $RDC_l \in$ RDCSet;

Query expression Q can also be denoted as a set of the following form: Q $\{Exp_1, \ldots, Exp_n, CSC_1, \ldots, CSC_u, RDC_1, \ldots, RDC_v\}$.

So far, we have used a bottom-up strategy to analyze the query input interfaces of search engines and to build up the ADMIRE data model. We can use such representations of query expressions (**Definition 3.25**) to describe the user interfaces and query capabilities of heterogeneous Web search engines.

In the next section, several concrete Web information sources will be described by using this definition.

3.3 Wrapper/Mediator Modeling

In this section, we provide some concrete examples of using Definition 3.25 to model the query capabilities and user interfaces of search engines or meta-search engines. We have built a meta-search engine to search for scientific publications on the Web. This meta-search engine employs a "Mediator-Wrapper" architecture [Wie92] that has been used by many information integration systems. Figure 3.13 illustrates the mediator/wrapper architecture of our meta-search engine. We will discuss in detail this architecture and some issues on wrapper generation and result merging in chapter 6.

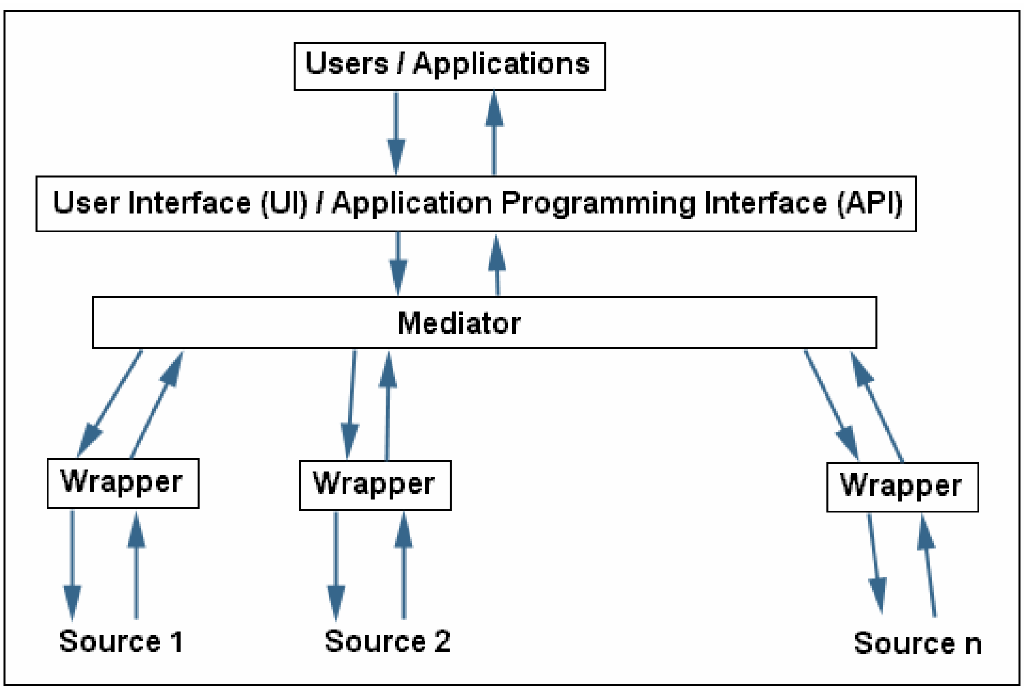


Figure 3.13 Mediator / Wrappers architecture

Many institutions, organizations and publishing companies provide search services for users to search through publication information on the Web, for example, ACM-DL, NCSTRL, IDEAL, Kluwer, Elsevier, etc. Each search service collects its own publications. In the following, we provide a description of some publication search engines. First, we use query expression (**Definition 3.25**) to describe the query capabilities and the query input interface of ACM digital library.

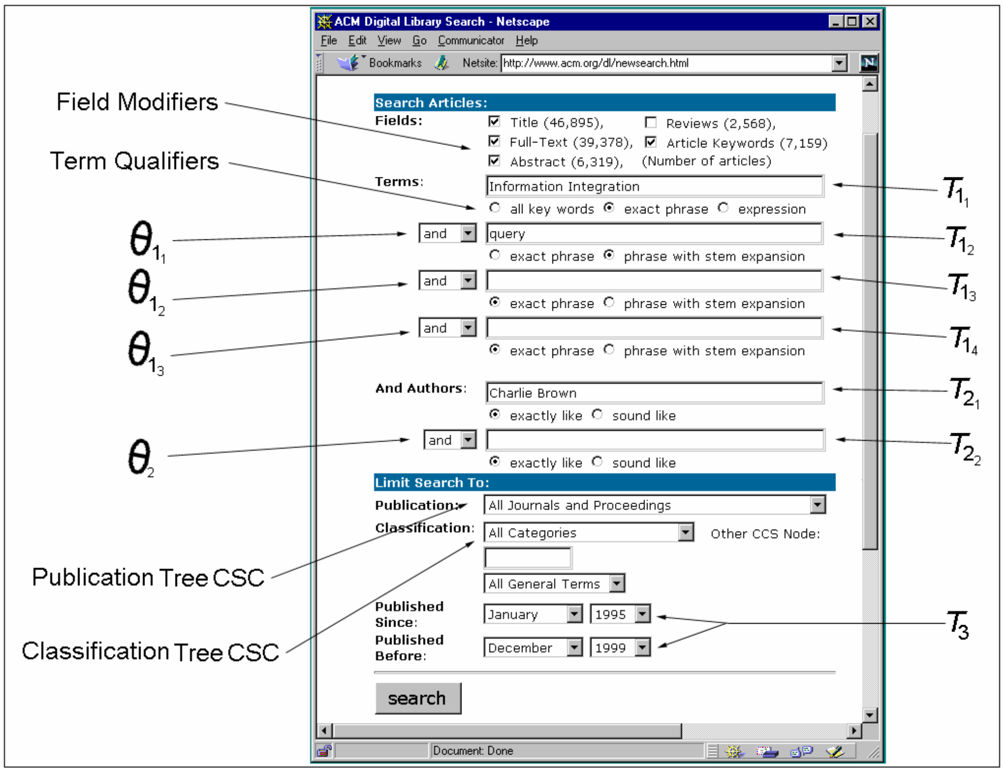


Figure 3.14 Query expression illustration of ACM-Digital Library

The query expression of ACM-Digital Library (see Figure 3.14) can be denoted as follows:

Q^{w}_{acm} = {**PRI**($Exp_1(T_{1_1}\theta_{1_1}T_{1_2}\theta_{1_2}T_{1_3}\theta_{1_3}T_{1_4})$), **PRI**($Exp_2(T_{2_1}\theta_2 T_{2_2})$), T_3, Publication TreeCSC, Classification TreeCSC}, where

- Field(T_{1_i}) $\subseteq$ {<Title>, <Full-Text>, <Review>, <Article Keywords>, <Abstract>}, 1≤i≤4;
- Field(T_{2_i}) = <Author>, 1≤i≤2;

• Field(T_3) = <Date>;
• Qualifier(T_{1_1}) $\in$ {<Multiple Words>, <Phrase>, <Expression>};
• Qualifier(T_{1_i}) $\in$ {<Using Stem Expansion>, <Phrase>}, 2≤i≤4;
• Qualifier(T_{2_i}) $\in$ {<Exactly Like>, <Sound Like>}, 1≤i≤2;
• $\theta_{1_i} \in \{\wedge, \vee, \neg, \sim\}$, 1≤i≤3;
• $\theta_2 \in \{\wedge, \vee\}$;
• Publication TreeCSC = <All Publication> {–, <All Journals and Proceedings of the ACM>{+, <Communications of the ACM>, <Computing Surveys>, …}}
• Classification TreeCSC = <All Categories> {–, <Computer Science&Technology>{+, <Software>, <Theory of Computation>, …}}

Here term T_3 is limited by the <Date> field. However, from Figure 3.14, we can recognize that there are 4 pull-down menus (two for “Month”, two for “Year”) instead of an input-box for this term. The field modifier of term T_{1_i} can be a sub-set of all fields, such as {<Title>, <Abstract>}, and so on.

In the following we describe the query capabilities and the query input interface of NCSTRL search engine. From Figure 3.15, we can see that it consists of two CGI forms.

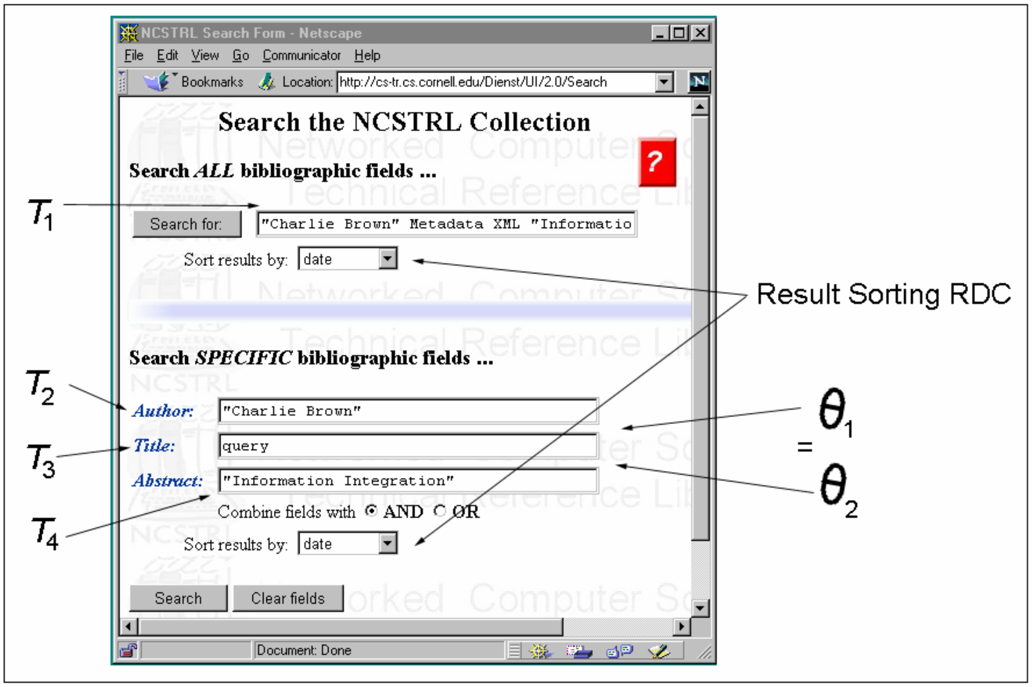


Figure 3.15 Query expression illustration of NCSTRL

The query expression of NCSTRL (see Figure 3.15) can be denoted as follow:

$Q^{w}_{ncstrl,1}$ ={ T_1, Result Sorting RDC, Category TreeCSC}, where

- Field(T_1) ={<Author>,<Title>,<Abstract>};

$Q^{w}_{ncstrl,2}$ ={**SEQ**(*Exp*($T_2\theta_1T_3\theta_2T_4$)), Result Sorting RDC, Category TreeCSC}, where

- Field(T_2) = <Author>; Field(T_3) = <Title>; Field(T_4) = <Abstract>;
- Qualifier(T_i)∈ {<Exactly Like>, <Multiple Words>, <Using Stem Expansion>, <Phrase>}, 2≤i≤4;
- $\theta_i \in \{\wedge, \vee\}$, 1≤i≤2, $\theta_1 = \theta_2$;
- Result Sorting RDC = {<Rank>, <Author>, <Date>, <Institution>, <Title>}
- Category TreeCSC = <All Categories> {–, <Computer Science&Technology>};

Each search engine of ACM-DL and IDEAL (see Figure 3.16) has only one query form. Therefore, there is only one query expression for each of them. In NCSTRL, there are two separate query forms. The first query form allows users to search keywords in all fields; while in the second one, users must limit the keywords to a specific field.

Finally, we describe the query capabilities and the query input interface of IDEAL search service (see Figure 3.16).

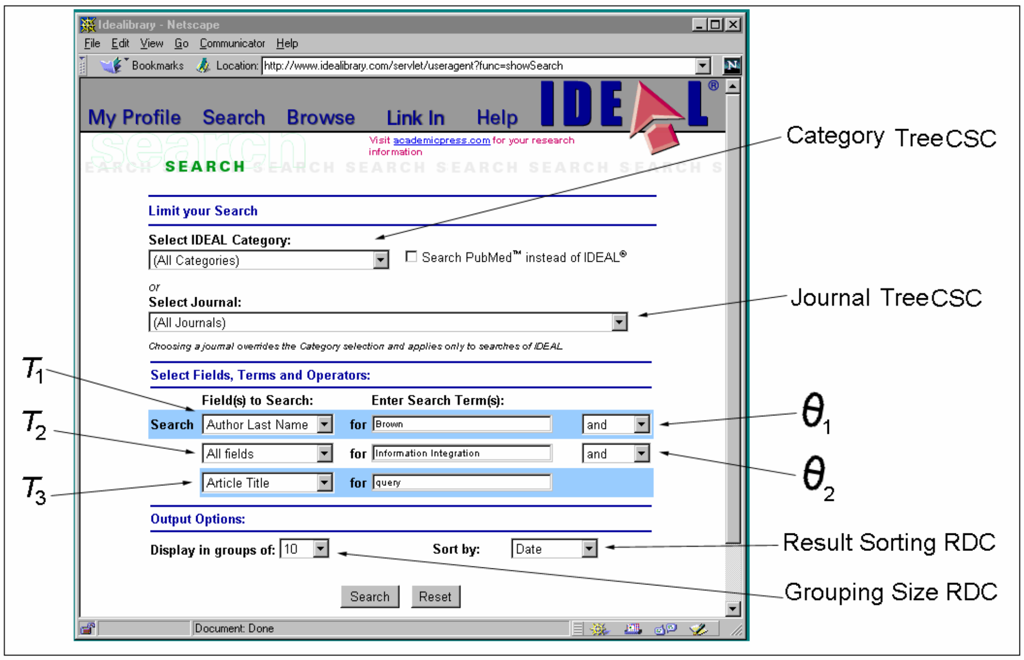


Figure 3.16 Query expression illustration of IDEAL

Q^{w}_{ideal} = {**SEQ**($Exp(T_1\theta_1T_2\theta_2T_3)$), Category TreeCSC, Journal TreeCSC, Grouping Size RDC, Result Sorting RDC}, where

- Field(T_i)∈ {<Anywhere>, <Title>, <Abstract>, <Author>, <Affiliation>, <Date>}, $1 \le i \le 3$;
- Qualifier(T_i)∈ {<Exactly Like>, <Multiple Words>, <Using Stem Expansion>, <Phrase>}, $1 \le i \le 3$;
- $\theta_i \in \{\wedge, \vee, \neg\}$, $1 \le i \le 2$;

• Category TreeCSC = <All Categories> {+, <Biomedical Sciences>, <Business and Law>, <Computer Science>, <Economics and Financing>, …}
• Journal TreeCSC = <All Publication> {–, <All Journals and Proceedings of the IDEAL>{+, <Advances in Applied Mathematics>, <Animal Behavior>, …}}
• Grouping Size RDC = {10, 20, 50, 100}
• Result Sorting RDC = {<Relevance ranking>, <Date>, <Journal>}

From the above three examples, we know that there are many commonalities and differences among search engines. Some provide rich and complicated functions, while others have simple functions. Controls of some search engines are less limited, while controls of other search engines are limited by all kinds of constraints. Only if all these features are understood by a meta-search engine can it achieve a better performance.

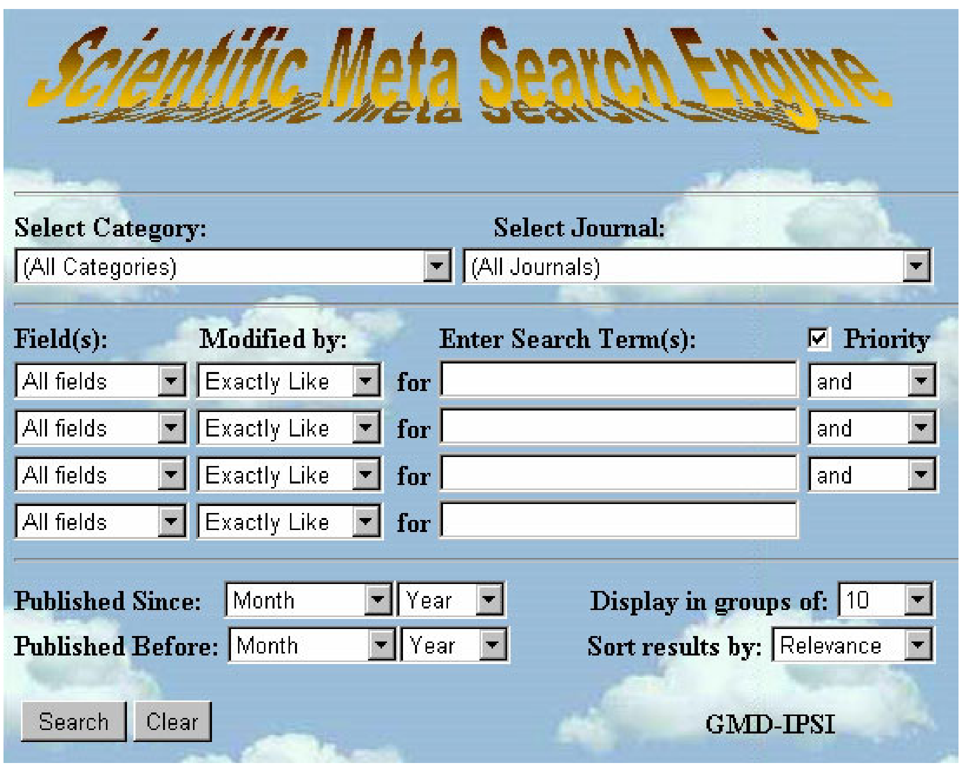


Figure 3.17 The query page of a meta-search engine

Figure 3.17 displays the query interface of our scientific publication oriented meta-search engine prototype - SPOMSE (HTML static UI version). This mediator integrates several scientific search engines such as the above-mentioned three ones and Elsevier, Kluwer, etc.

In Figure 3.17, there is a check box on the right in the center part of the query page. If it is checked, the query expression will be interpreted by priority; otherwise, the query expression will be interpreted sequentially. The query expression of the mediator can be described as follows:

$Q^m = \{\Phi(Exp(T_1\theta_1T_2\theta_2T_3\theta_3T_4))$, T_5, Category TreeCSC, Publication TreeCSC, Grouping Size RDC, Result Sorting Criteria RDC}, where

- $\Phi \in$ {**PRI**, **SEQ**};
- Field(T_i)$\in FM_i$={<Title>, <Full-Text>, <Review>, <Article Keywords>, <Abstract>, <Author>, <Affiliation>, <ISBN>, <ISSN>, <Journal Title>, <Citation>, <Editor>, <Anywhere>}, $1 \leq i \leq 4$;
- Field(T_5) = <Date>;
- Qualifier(T_i)$\in TQ_i$= {<Exactly Like>, <Multiple Words>, <Using Stem Expansion>, <Phrase>, <Expression>, <Sound Like>, <Spelled Like>}, $1 \leq i \leq 4$;
- $\theta_j \in \{\wedge, \vee, \neg, \sim\}$, $1 \leq j \leq 3$;
- MediatorTreeCSC = $\bigcup_{i=1}^{n} WrapperTreeCSC_i$; //Suppose there are n target search engines
- MediatorRDC = $\bigcup_{i=1}^{n} WrapperRDC_i$;

Note: The field modifiers of a meta-search engine can be the Dublin Core metadata element set. Here because our meta-search engine prototype has been built on the top of several concrete bibliographic databases that have not adopted the Dublin Core standard yet, we only use their current field modifiers.

Here we provide the query capability description of a static user interface of a meta-search engine. Actually, users' queries vary greatly from simple to complex; therefore, static user interfaces are not efficient for users to input queries, especially complex queries. In chapter 5, we will introduce a method for constructing adaptive, progressive user interfaces for meta-search engines. This kind of dynamically-generated user interfaces can solve the above-mentioned problems to some extent.

In the above, we use 'Query Expression' (see Definition 3.25) to describe some wrappers for several concrete domain-specific search engines and a mediator. By using this method, the user interfaces and query capabilities of the meta-search engine and target search engines can be sufficiently described. However, this will inevitably increase the difficulty and complexity of the query translation from a meta-search engine to target search engines. In chapter 4, we will discuss a constraint-based query translation algorithm in detail.

3.4 Closing remarks

So far, we have conceptually introduced the ADMIRE data model that can be employed to formally describe the user interfaces and the query capabilities of heterogeneous information sources on the Web. This data model presents a methodology for constructing a shared and common understanding of some domain that can be communicated between different application systems and people. This shared understanding can provide machine-processible semantics of information sources. The source wrapping can be done by humans using formal definitions (see Definitions 3.1-3.25) to analyze, extract and describe the syntactic, semantic, functional information from the query input pages and then enable software agents to understand the query capabilities of sources, thus facilitating the automatic query translation across heterogeneous information retrieval systems.

The ADMIRE data model can be employed by meta-search engines and information integration systems to automate the specification of query input interfaces and query capabilities of remote target sources with the help of automatic document extraction tools and heuristics (or Machine Learning) technologies (Document analysis and extraction technologies are outside the scope of this thesis; in section 6.2.4, we will introduce some existing document extraction tools). This data model can also allow users to build dynamic and personal views over a growing number of information sources. In chapter 6, we will discuss some technical issues on wrapper generation and result merging with respect to a sample implementation of this data model within our SPOMSE prototype.

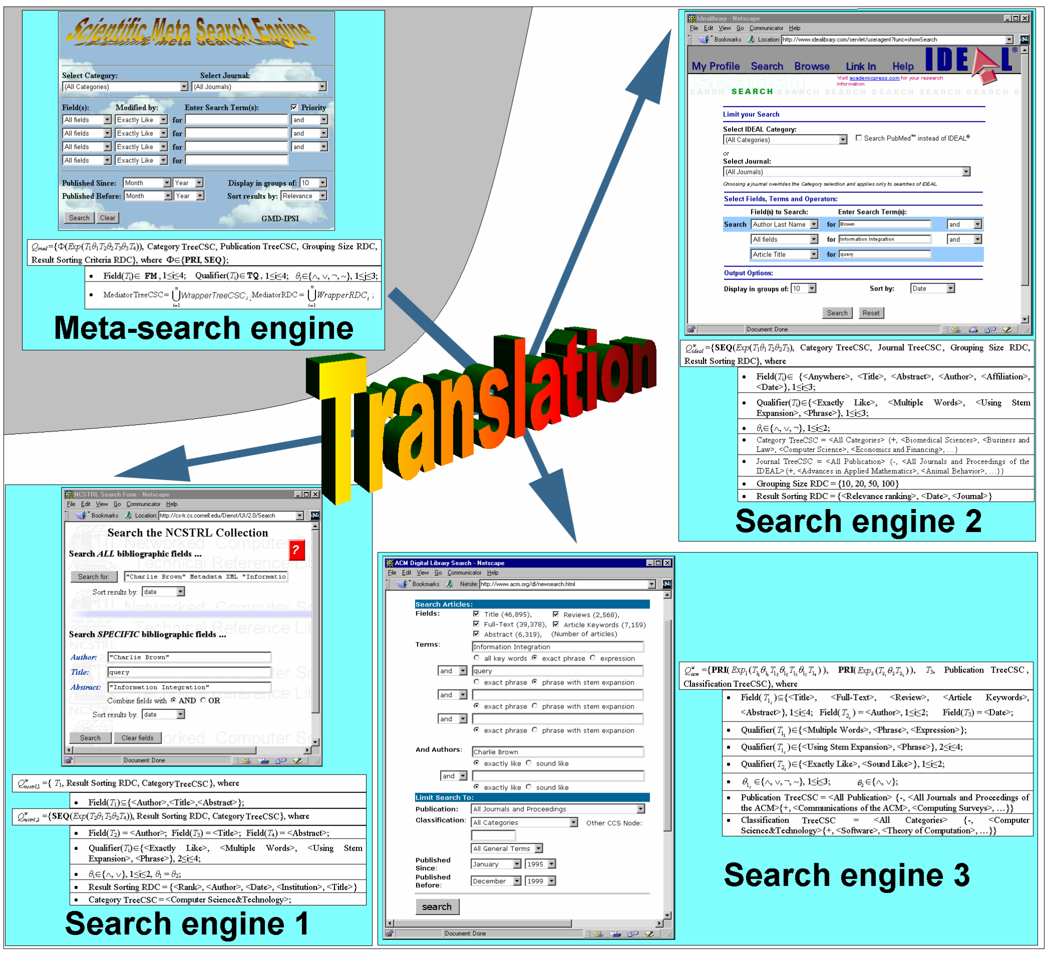


Figure 3.18 Sketch application map of the ADMIRE data model

As displayed in Figure 3.18, query expressions are used to describe wrappers for three scientific publications-oriented search engines and a concrete mediator for meta-search engines (or information integration systems). After that, the arrows (from the meta-search engine to search engines) in this figure are what we are concerned about now. This problem can be expressed as: how can the query expressions of meta-search engines be automatically translated into the query expressions of specific search engines? In chapter 4, we will answer this question.

Obviously, the static query input interface for a meta-search engine is not flexible, scalable, and efficient enough. In chapter 5, we will discuss how we can construct adaptive, dynamically-generated query input interfaces for a meta-search engine.

Chapter 4

Constraint-based Query Capability Translation

In chapter 3, we have introduced the ADMIRE data model that can be employed to describe the query capabilities and user interfaces of search engines and meta-search engines. Based on this data model, we will now discuss the problem of translating queries formulated on the query input user interfaces of a meta-search engine into the formats understood by heterogeneous search engines.

The query translation process consists of three steps: (1) when users submit a query, the meta-search engine first checks which search engines may have relevant answers to the query by means of source selection methods; (2) then it maps the query to wrappers for all selected sources and dispatches the transformed queries to the corresponding sources; (3) after results are retrieved, it post-processes the results from each source; and finally, all results are merged and displayed to the user. Many papers regard the source selection problem separate from the query translation problem because their source selection methods are based on keyword statistics. Our source selection approach is based on the users' information needs on the domain aspect, so it can be regarded as part of the query translation problem.

From previous chapters, we know that various information sources have very different query input interfaces and query capabilities. For example, in Figure 3.18, the three target search engines (i.e., ACM-DL, NCSTRL, IDEAL) have different user interfaces. It is not easy to automatically translate queries that are input on the user interface of the meta-search engine (see Figure 3.18) to the formats that can be accepted by the user interfaces of these three search engines.

When translating a query from a meta-search engine to a remote target search engine, the mediator will consider the functional restrictions of terms, term modifiers, and logical operators among the controls on the user interfaces to the underlying sources, so the meta-search engine can utilize the query capabilities of the specific sources as much as possible. In addition, a two-phase query subsuming mechanism has been applied to compensate for the functional discrepancies between sources, in order to achieve a more accurate query translation.

This chapter begins by discussing the source selection problem. Then in section 4.2, we give a formal query translation algorithm and analyze the query translation problems. Finally, in section 4.3, we illustrate a concrete example for query translation and show some of the problems.

4.1 Source Selection

In this section, we discuss the source selection problem which is also called the query routing problem. This problem is the prelude of the query translation process. Without source selection, the query translation in a large-scale meta-search engine or an information integration system is pointless. For example, when users want to search for publications on orthodontics and then the meta-search engine makes efforts to map this query into a search engine for MP3 music or car sales, nothing relevant will be returned. In addition, searching all sources for a user query will consume valuable computer and communication resources. As more and more information sources charge users for searching, this makes the source selection problem more important. The ultimate goal for source selection is to minimize the number of information sources to which the query is broadcast.

In the following, we first describe the source selection problem and provide a formal definition. Then a source selection algorithm is given.

When users have completed a query construction and submit it, if this query contains domain and category information, the meta-search engine can judge which search engines will be selected to answer the query. For the purpose of fast response and saving CPU resource and network bandwidth, a meta-search engine only chooses some potentially relevant search engines to answer a user query.

If the number of relevant search engines is large, a priority order will also be decided. For example, suppose that 500 search engines are selected because they may return relevant results. Then the meta-search engine forks 500 threads with each thread doing a lot of work for a search engine (such as query translating, query sending, results receiving, results post-processing, etc.). Forking 500 threads simultaneously will consume a lot of CPU resources. It is even worse for a common PC. Therefore, the meta-search engine can divide these selected search engines into 10-20 groups according to the running priorities. When a thread finishes, another thread with lower-priority will be started. Later on we will introduce the calculation of similarity points for selected search engines.

Definition 4.1 Selecting Sources Through CSCs

When mapping a query, if the CSCItem <A> (see Definition 3.2) in a TreeCSC (see Definition 3.5) of the meta-search engine is the same as the CSCItem <B> of the TreeCSC of a target search engine or is the descendant CSCItem of the CSCItem <B>, the CSCItem <A> will match to this search engine, i.e., this search engine can support queries that have the selection of this CSCItem.

In definition 4.1, we can see that when users choose some CSCItems from CSCs of a meta-search engine, the mediator will decide which target search engines are relevant to these selected CSCItems by using source mapping tables (see Definition 3.8). In the next sub-section, we will provide an algorithm for selecting sources.

The source selection algorithm (in pseudo-code) is listed in Table 4.1:

Table 4.1 Source selection algorithm

Input: *all TreeCSCs of all information sources*

Output: *all potentially relevant information sources, each with a similarity point*

1. *SetofSelectedSources* ←ϕ; // ϕ denotes empty set.
2. *bUserSelectCSCItem* ←FALSE; // This Boolean variable is used to denote whether a user has manipulated any classification selection controls to limit his/her information needs.
3. FOR (each TreeCSC)
4.   IF (the user selects some CSCItems)
5.   FOR (each of the CSCItems the user selects)
6.   IF (the CSCItem that the user selects is (or is a descendent of) a CSCItem of corresponding TreeCSC of each target source)
7.   *SetofSelectedSources* ← this source;
8.   Rearrange the order of *SetofSelectedSources* by computing similarity points (e.g., see Figure 4.1)
9.   *bUserSelectCSCItem* ←TRUE;
10. IF (*bUserSelectCSCItem* == FALSE) // This means that if the user query does not contain any domain and category restrictions on search results, the meta-search engine will select all integrated target sources.
11.   *SetofSelectedSources* ←All target sources;

In line 8 of Table 4.1, the order of the selected sources will be rearranged by *similarity points*. We stipulate the following computing rule: if the selected CSCItem (or default value) can be exactly supported by the target source (i.e., there is a corresponding CSCItem in this search engine), the similarity point is 0; Otherwise, if the selected CSCItem can only be supported by a CSCItem of the target source that is parent CSCItem of this selected CSCItem, then the similarity point is 1; Otherwise, if the selected CSCItem can only be supported by a CSCItem of the target source that is grand-parent CSCItem of this selected CSCItem, then the similarity point is 2; and so on. The above algorithm adds the similarity points of all TreeCSCs for each selected target source. The source with the least point will be executed with highest priority because it can support the user query the best and may return the most relevant results.

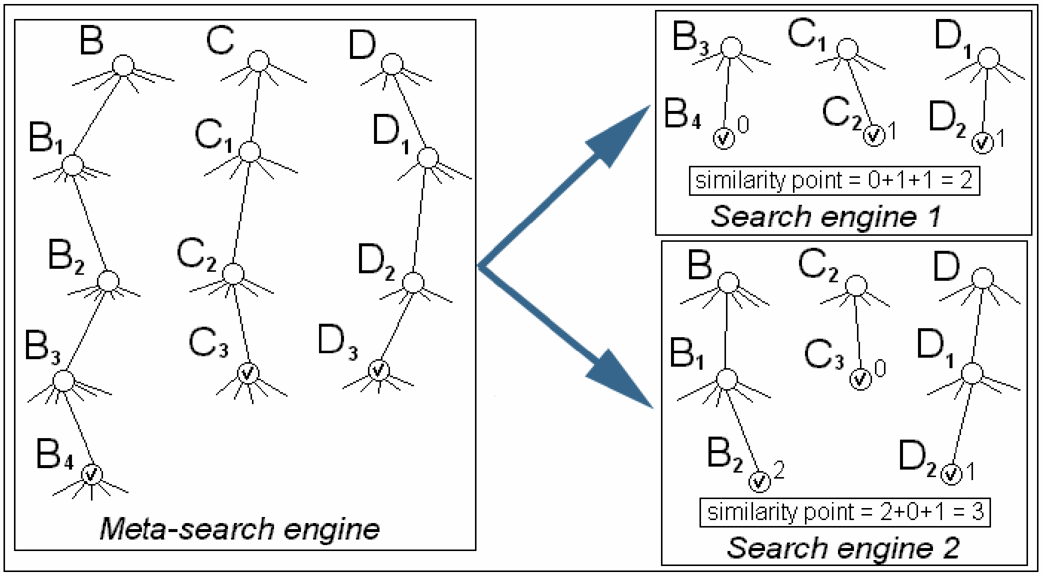


Figure 4.1 Computing similarity points of target sources

In the following we provide an example in Figure 4.1. Users select the ‘B_4’, ‘C_3’, and ‘D_3’ nodes from three different TreeCSCs when inputting a query on a meta-search engine. Now we compute the similarity points of two target search engines. For the first search engine, it can only support query on the ‘B_4’, ‘C_2’, and ‘D_2’ nodes, so its similarity point is 2 (= 0+1+1). For the second search engine, it supports query on the ‘B_2’, ‘C_3’, and ‘D_2’ nodes, so its similarity point is 3 (= 2+0+1). Therefore, the first search engine may support the query better. Although we use multi-threads technology to run several searches on different target search engines simultaneously, but if there are too many selected sources (e.g., 100 or more), the meta-search engine should run the searches in groups, so as to lessen the load of CPU.

4.2 Constraint-based query translation analysis

In this section, we provide a constraint-based query translation algorithm and analyze the process of query translation. First, section 4.2.1 provides a formal, overall description of the applied algorithm. Section 4.2.2 provides an illustrating example and a definition for query disjunctivizing. Section 4.2.3 discusses the translation from a conjunctive query into a single target query and how the common filters and the special filters are generated and how they work. Section 4.2.4 provides two detailed examples to explain this translation method. Section 4.2.5 discusses the translation from an arbitrary query into several targets. Finally, section 4.2.6 summarizes this constraint-based query translation algorithm.

4.2.1 Constraint-based query translation algorithm

In Chapter 3, we have introduced the concept of *query expressions* (see Definition 3.25 in section 3.2.6) that can be employed to describe the query capabilities of search engines and meta-search engines (Section 3.3 gives several concrete examples of applying query expressions). Now in this section we discuss how to translate a query expression $Q_{mediator}$ from a meta-search engine into the query expression $Q_{wrapper}$ understood by a search engine. Without loss of generality, we suppose $Q_{wrapper}$ to be:

$$Q_{wrapper} = (\bigcup_{i=1}^{n}(\Phi_i(Exp_i(T_{i_1}\theta_{i_1}...\theta_{i_{k-1}}T_{i_k})))) \cup (\bigcup_{j=1}^{u} CSC_j) \cup (\bigcup_{l=1}^{v} RDC_l);$$

- where $n \geq 1$, u, $v \geq 0$;
- $\Phi_i \in \{$***SEQ***, ***PRI***, ***VSQ***$\}$;
- Field(T_{i_l})$\in FM_{i_l}$, $1 \leq i_l \leq i_k$;
- Qualifier(T_{i_l})$\in TQ_{i_l}$, $1 \leq i_l \leq i_k$;
- $\theta_{i_l} \in L_{i_l}$, $1 \leq i_l \leq i_{k-1}$;

• $CSC_j \in$ CSCSet;
• $RDC_l \in$ RDCSet;

In the above query expression, each CSC will be used for source selection at the beginning of the query translation. In section 4.1, we have introduced the source selection algorithm: for each CSC in the meta-search engine, the meta-search engine will check if the user has selected one or more CSCItems. If the user has done so and the selected CSCItem(s) can be mapped into the corresponding CSC of the target source, the meta-search engine will fill in the query parameters of the source using the user-selected value or the value of the user-selected CSCItem's ancestor; otherwise, it will fill in the default value (it is often the value of the root CSCItem). In the same way, for each RDC in the meta-search engine, the value of user-selected RDCItem will be passed to the corresponding RDC of the target source. If the target source cannot support the value of a user-selected RDCItem, the meta-search engine will have to post-process the retrieved results from this target source. For example, when users input a query requiring that results be sorted by *title*, but the target source cannot satisfy this results sorting requirement, then the meta-search engine sorts the results from this search engine itself. In the following, when discussing the query translation problem, we do not consider these CSCs and RDCs. So the above-mentioned query expression can be simplified as:

$$Q_{wrapper} = \bigcup_{i=1}^{n} (\Phi_i(Exp_i(T_{i_1}\theta_{i_1}...\theta_{i_{k-1}}T_{i_k}))) .$$

Because any conjunctive query expression in sequential order can be transformed into a disjunctive expression that fits priority explanation, we assume that query expressions are interpreted in priority order by the mediator. We suppose the query expression of the mediator (i.e. users' query expression) is:

$Q_{mediator}$=**PRI**($Exp(T_1\theta_1 ... \theta_{m-1}T_m)$);
• where $m \geq 1$;
• Field(T_i)$\in FM_i$, $1 \leq i \leq m$;

• Qualifier(T_i) $\in$ TQ_i, $1 \le i \le m$;
• $\theta_i \in \{\wedge, \vee, \neg, \sim\}$, $1 \le i \le (m-1)$;

Note:

FM_i = {<Title>, <Full-Text>, <Review>, <Article Keywords>, <Abstract>, <Author>, <Affiliation>, <Date>, <ISBN>, <ISSN>, <Journal Title>, <Citation>, <Editor>, <Anywhere>};

TQ_i = {<Exactly Like>, <Multiple Words>, <Using Stem Expansion>, <Phrase>, <Expression>, <Sound Like>, <Spelled Like>, <Before>, <After>}.

From the above two query expressions: we can see that $Q_{wrapper}$ can contain multiple query expressions while $Q_{mediator}$ only contains one query expression. In section 3.3, we have outlined that the query input page of the NCSTRL search engine contains two CGI query input forms; therefore it has two query expressions. But ACM-DL and IDEAL have only one query expression each. A user query naturally is one query expression.

How can a meta-search engine automatically translate $Q_{mediator}$ to $Q_{wrapper}$? First, $Q_{mediator}$ has to be transformed into a standard disjunctive expression that contains one (or some) conjunctive sub-expression(s); After that, we map the main part of the query expression: for each of the sub-expressions of the wrapper and each of the sub-expressions of the mediator, the meta-search engine will check if they are compatible, if this holds true then it will map this sub-expression. Otherwise, the meta-search engine will rearrange the order of terms or decompose the sub-expression into several smaller sub-expressions (equivalent or minimal subsuming), then continue to check the compatibility and to construct the translation. Finally, the meta-search engine post-processes the results. This post-processing is not merely the union of all results. It will furthermore consider the former decomposition of the query expression that the search engine did not support.

The algorithm listed in Table 4.2 is for translating a query expression from the mediator to a selected source wrapper. Here we ignore the coordination of conflicts from the operational aspects, such as measuring and naming conventions (e.g., different expressions of date type: 19.11.1999=11/19/1999, different scaling methods: some systems use values between [0, 1] to denote the relevance, while some use values between [0, 100]). Some search engines can return all information about an entry at one time; while some search engines require users to visit their server more than one time, e.g., Cora Search Engine only returns the title, authors and abstract fields and links to a Postscript file and BibTeXEntry. The meta-search engine needs to revisit the Cora search engine to get the information about book title and publishing date by following the link "BibTeXEntry"; and the URL of this "BibTeXEntry" is determined by session number and the id of this entry found on the previous result. The following algorithm supposes that all these kinds of translations and transformations are realized inside each wrapper.

When the system maps a query expression (user input) $Q_{mediator} = Exp(T_1\theta_1...\theta_{m-1}T_m)$) to a query expression supported by a search engine, $Q_{mediator}$ will be first disjunctivized as the following form: (we will in next section provide the definition of "***query disjunctivizing***")

$Q_{mediator} = \bigcup_{i=1}^{N_m^{'}}(\bigcap_{j=S_p(i)}^{E_p(i)} T_j^m)$, where $N_m^{'}$ means the number of conjunctive (linked by $\wedge$ '*AND*', $\neg$ '*NOT*', ~ '*NEAR*') sub-expressions;
• $S_p(i)$ means the start position of the i^{th} conjunctive sub-expression;
• $E_p(i)$ means the end position of the i^{th} conjunctive sub-expression;
• $S_p(1)$= 1; $E_p(N_m^{'})$ =$k^{'}$; this means that all these $N_m^{'}$ sub-expressions together contain $k^{'}$ terms.
• $S_p(i+1) = E_p(i) +1$; This means that all sub-expressions are ordered one by one.
• $(E_p(i)–S_p(i)) \geq 2$. This means that one sub-expression contains at least two terms.

The rewritten form consists of $N_m^{'}$ (≥ 0) conjunctive sub-expressions. In the following, Table 4.2 lists the overall query translation algorithm that is used to map queries from the meta-search engine to a specific search engine. Figure 4.2 illustrates this algorithm.

Table 4.2 Query translation algorithm

Algorithm ***CQM( )*** //Pseudo-code

Input: *a user query expression (M) and the query expressions of a selected target search engine*

Output: *all transformed query sub-expressions (derived from the original user query) that can be supported by the target search engine*

int N_S = the number of the selected search engine(SE)'s sub-expressions; /* N_S = the n in $Q_{wrapper}$ */

SingleTermSet = ϕ; // ϕ denotes empty set. When a query expression is decomposed, one or more single terms (SingleTerm) may be generated.

FOR(i=1; i<= N'_m; i++) /* The initial value of N'_m comes from above disjunctivization of the mediator $Q_{mediator}$, and it may change during this FOR cycle.*/

$M = \bigcap_{l=S_p(i)}^{E_p(i)} T_l^m$; /* One of conjunctive sub-expressions of $Q_{mediator}$ */

BOOLEAN SendTag = FALSE; /* It is used to denote if this sub-expression can be sent to target search engine. */

FOR(j=1; j<= N_S; j++)

$S = \Phi_j(Exp_j^s(T_{j_1}\theta_{j_1}...\theta_{j_{k-1}}T_{j_k}))$;

IF (MatchSubQE(M, S, ***S$_\mathbf{i}$**)) /*This Boolean function will be described in Table 4.3; **S$_\mathbf{i}$** means the reorganized sub-expression*/

FOR(n=1;n<=(NumberOf(S)–NumberOf (S_i)); n++) /* The function NumberOf() returns the number of terms in a query expression*/

IF (NotEmpty(SingleTermSet) AND ($\exists$ a SingleTerm $ST \in$ SingleTermSet and it is fit for the constraint conditions(or after relaxing them) of the slot in this S_i, such as Field(ST) $\in/\subseteq$ Field(T_{j_n}), $\theta_{j_{n-1}}$ can be set as $\vee$, etc))

11. Append *ST* to S_i ($T_{j_n} \leftarrow ST$; Field(T_{j_n})$\leftarrow$Field(*ST*); $\theta_{j_{n-1}} \leftarrow \vee$);

12. Remove *ST* from the SingleTermSet;

13. ELSE Break FOR n;

14. Dispatch this newly generated SE sub-expression S_i;

15. SendTag = TRUE;

16. Break FOR j

17. IF (SendTag == FALSE)

18. $S_i \leftarrow \varnothing$; //$\varnothing$ means empty set

19. Use function Decompose(*M*, *p, **S_i, *q, ***ST*) to decompose *M* into p sub-expressions that meet the limitations of the N_S sub-expressions and q single terms; /*int p >= 0; int q>=0; Figure 4.9 illustrates this function*/

20. Append these p newly generated sub-expressions to the unchecked set of mediator sub-expressions;

21. N'_m += p; /* Therefore, the FOR(i) cycle will be extended. */

22. Put these q single terms into SingleTermSet;

23. IF (NotEmpty(SingleTermSet))

24. Divide these remaining single terms into h (h>=1) groups. Each group will meet the limitations (or after broadening the term's modifiers) of one of the N_S sub-expressions. Some groups may contain several single terms combined by the logical operator $\vee$, while other groups each contains only one single term.

25. Dispatch these h sub-expressions: S_{i+1}, S_{i+2}, …, S_{i+h};

26. Result $\Leftarrow$ Post-processing (ΣS_i);

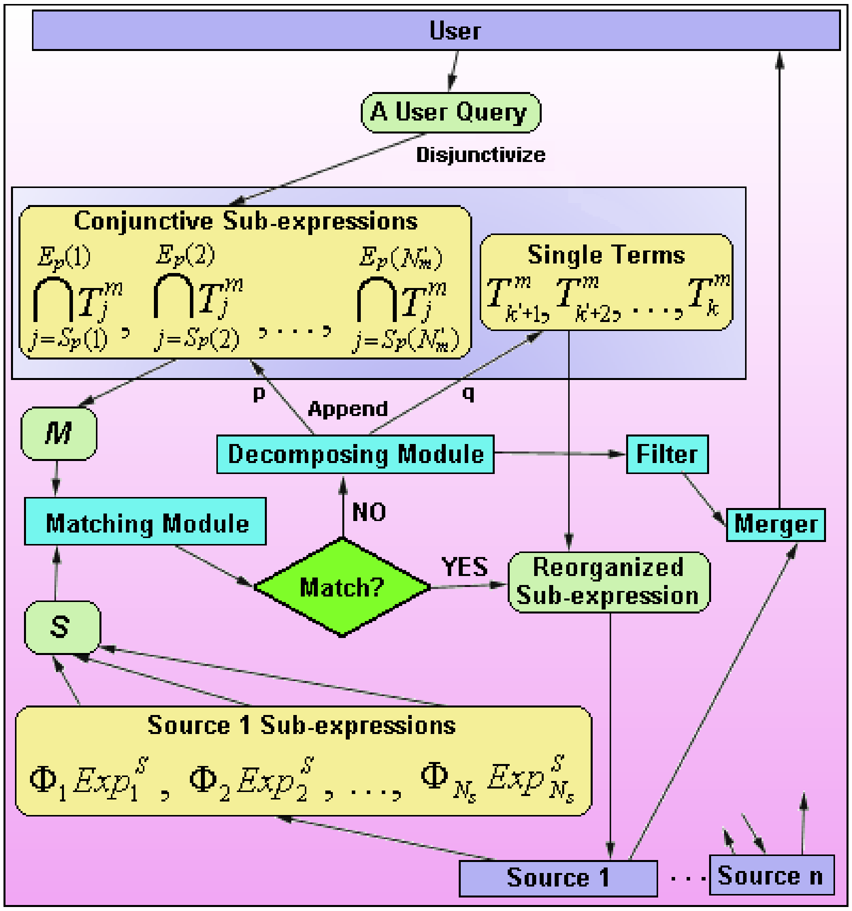


Figure 4.2 Constraint-based query translation

As Figure 4.2 displays, the process (*"Conjunctive Sub-expressions"* ⇒ *"Matching Module"* ⇒ *"Decomposing Module"* ⇒ *"Conjunctive Sub-expressions"*) is a recursive process. This means that a complex original query may be decomposed or relaxed once or several times until all the sub-queries of this query can be supported by target sources. In section 4.2.3, we will discuss the three modules in Figure 4.2 (i.e., "Matching Module", "Decomposing Module", "Filter Module"), the function "MatchSubQE(M, S, *S_i)" (see Line 8 in Table 4.2) and the function "Post-processing (ΣS_i)"(see Line 26 in Table 4.2).

4.2.2 An illustrating example

Before we explain the above-mentioned algorithm and discuss how a query is automatically translated from the meta-search engine to a target source, we will at first briefly outline how the motivating query ***Q*** in section 1.2 (i.e., **((Author is "Charlie Brown")** ***AND*** **((("Information Integration" in All fields)** ***AND*** **(Title contains "query"))** ***OR*** **(("Metadata", "XML") in Abstract))) in the Computer Science category, published during the period of 1995 to 1999, the results to be sorted by date.**) could be manually translated by a human into the formats supported by two sources: NCSTRL (see Figure 1.2) and ACM-Digital Library (see Figure 1.3).

Definition 4.2 Query Disjunctivizing

For a disjunctive user query (it contains the logical operator $\vee$ "OR"), it can be transformed into several conjunctive sub-queries (terms in each sub-query are combined by $\wedge$ "AND", ~ "NEAR", or $\neg$ "NOT" logical operators, in the following the operator $\Diamond$ is used to denote these three operators) or single terms. The following are two basic rules:

1. $(T_1 \vee T_2) \Diamond T_3 \Rightarrow T_1 \Diamond T_3; \quad T_2 \Diamond T_3.$
2. $(T_1 \vee T_2) \Diamond (T_3 \vee T_4) \Rightarrow T_1 \Diamond T_3; \quad T_2 \Diamond T_3; \quad T_1 \Diamond T_4; \quad T_2 \Diamond T_4.$

For example, the query *Q* can be disjunctivized into 2 sub-queries (here we omit "in the Computer Science category, published during the period of 1995 to 1999, the results to be sorted by date"):

Q_1: ((Author is "Charlie Brown") *AND* ("Information Integration" in All fields) *AND* (Title contains "query"));

Q_2: ((Author is "Charlie Brown") *AND* (("Metadata", "XML") in Abstract)).

This process can be denoted as:

(A and ((B and C) or D)) $\Rightarrow$ (A and B and C); (A and D).

In the following, we will discuss the translation of these two sub-queries into target sources respectively. The NCSTRL case will be discussed first.

The first query form of the NCSTRL search engine contains only one input-box. Therefore, all keywords and phrases will be entered into it. The sub-query Q_1 is constructed as in Figure 4.3(a) and the sub-query Q_2 is constructed as in Figure 4.3(b).

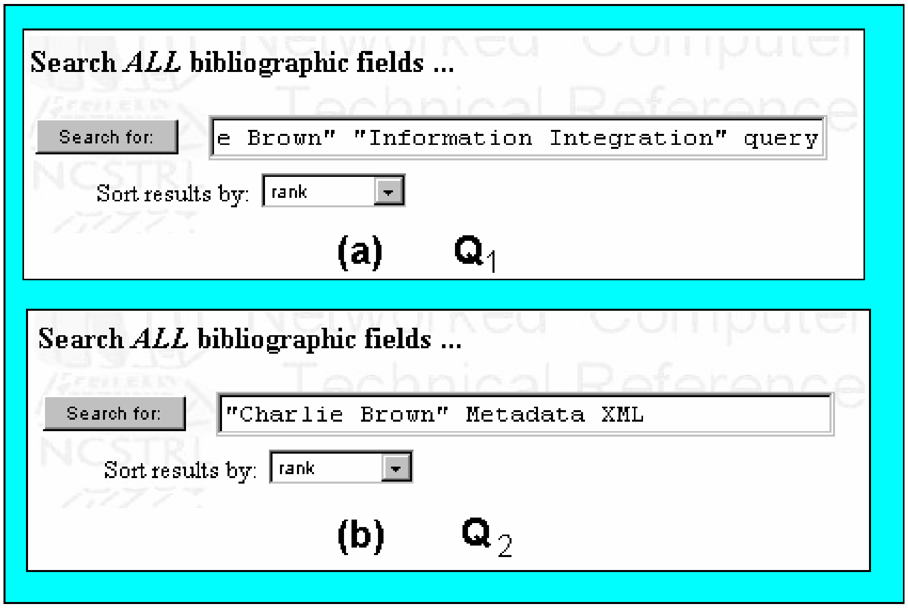


Figure 4.3 The first query form of the NCSTRL

In the following, we will discuss the translation of these two sub-queries into the formats supported by the second query form of the NCSTRL search engine. The term ("Information Integration" in all fields) in Q_1 can only be mapped into two concrete fields in the NCSTRL: 'Abstract' or 'Title'. Therefore, Q_1 must be further decomposed into 2 sub-queries:

$Q_{1,1}$: ((Author is "Charlie Brown") *AND* (Title contains (query, "Information Integration")));

$Q_{1,2}$: ((Author is "Charlie Brown") *AND* ("Information Integration" in Abstract) *AND* (Title contains "query")).

These two sub-queries $Q_{1,1}$ and $Q_{1,2}$ can be constructed in the second form of NCSTRL as in Figure 4.4(a) and in Figure 4.4(b), respectively.

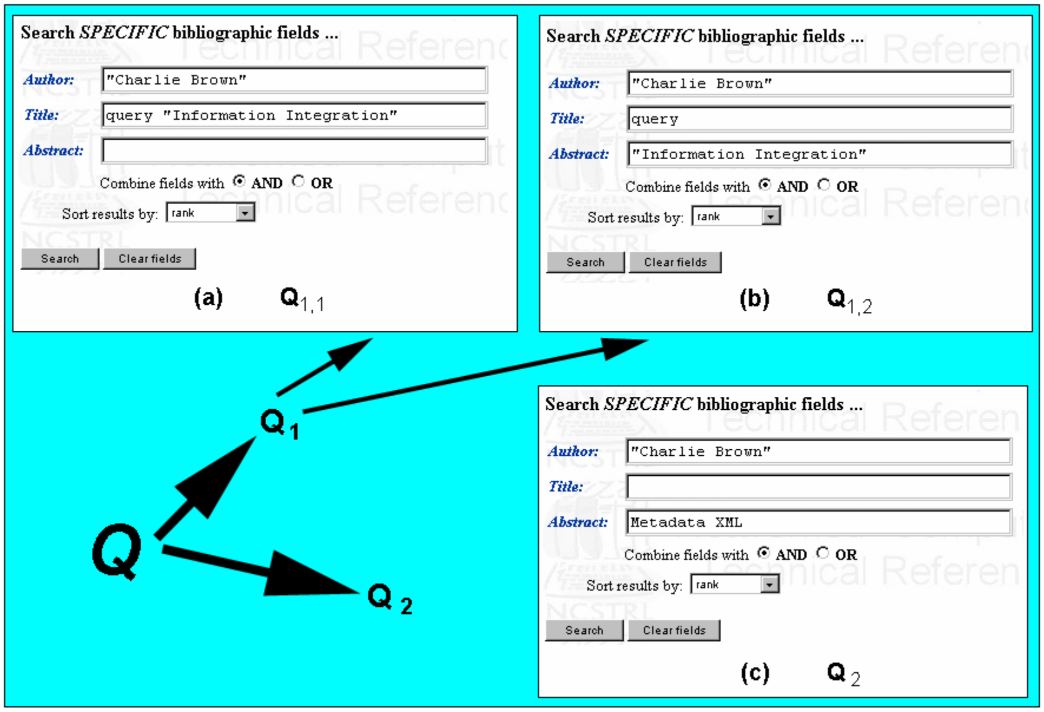


Figure 4.4 the second form of the NCSTRL

Q_2 can be expressed directly in the second query form of NCSTRL (see Figure 4.4(c)). The system dispatches these three sub-expressions to the underlying source. After the results return, the system selects the entries during the period of 01.1999 to 12.1999 and finally merges and sorts all the selected entries by date.

For the query form of ACM-digital Library, the two sub-queries Q_1 and Q_2 can be mapped separately into the query form in Figure 4.5(a) and in Figure 4.5(b). After the results return, the system selects the entries whose titles contain the word “query” (or the words stemming from this word).

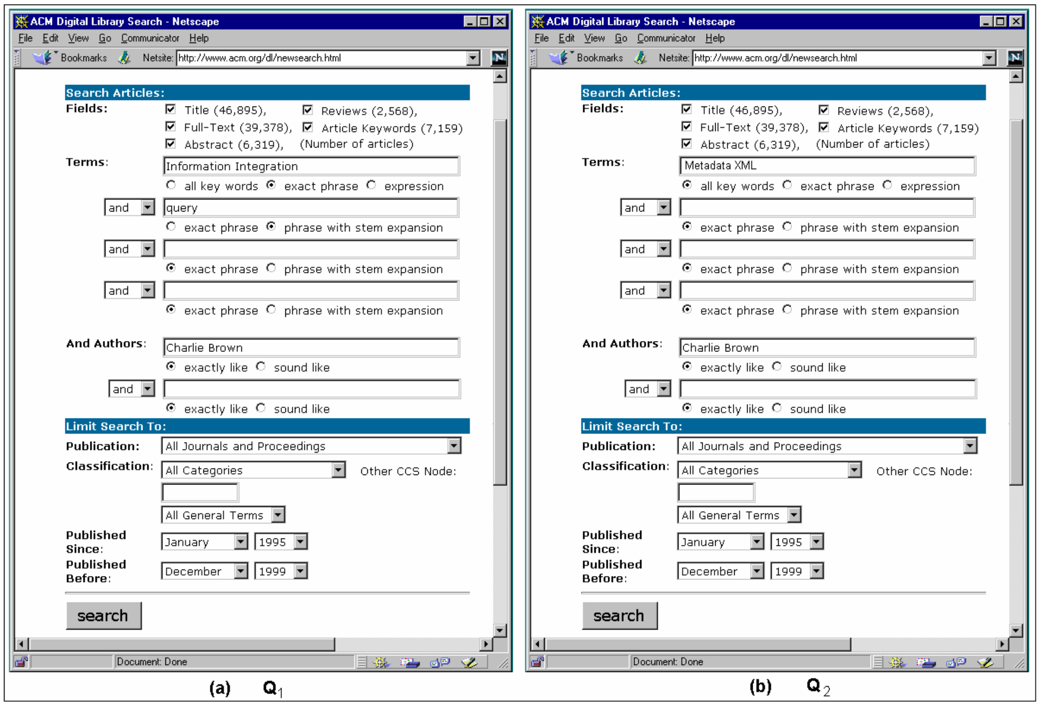


Figure 4.5 the query input on ACM-DL

From the above two examples, it seems that the query translation across heterogeneous information sources is not terribly difficult. However, if we let a computer automatically translate the query from the format understood by one source into the format understood by another source, perfect translation will be very hard. For example, in the case of the term ("Information Integration" in All fields), the computer is not intelligent enough to understand that the phrase "Information Integration" will not be found in the "Date" or "Author" fields. There are also many other constraints for query translation, e.g., a source may not support some term modifiers or some logical operators.

In order to explain our query translation algorithm more clearly, we use figures to express the query capabilities of query forms. For example, query Q_1 can be described as Figure 4.6 and the query capability of the second form (see Figure 4.4) of the NCSTRL search engine can be described as Figure 4.7.

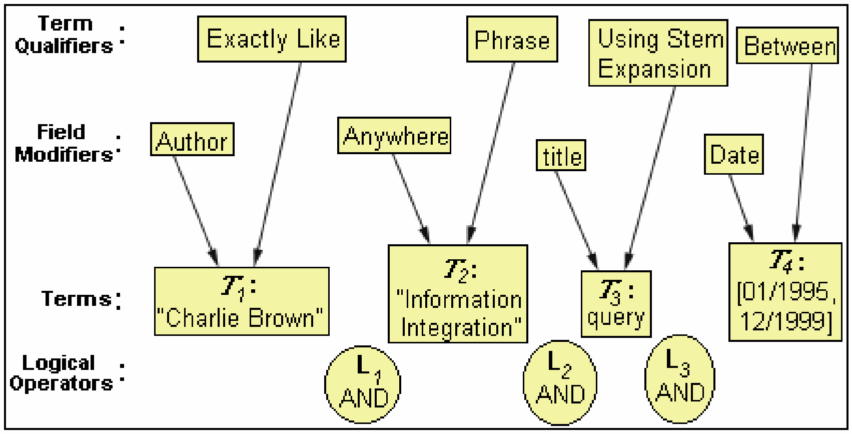


Figure 4.6 Query capability description of the sub-query Q_1

In Figure 4.6, this query expression contains four terms each with its own term qualifier and field modifier:

T_1 : Phrase "Charlie Brown" in 'Author' field;

T_2 : Phrase "Information Integration" can be found anywhere;

T_3 : "query" or its stemming words in the 'Title' field;

T_4 : Date between "01/1995-12/1999";

These four terms are combined by three logical operators L_1, L_2, and L_3.

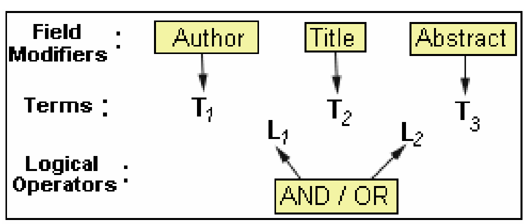


Figure 4.7 Query capability description of the second form of NCSTRL

In Figure 4.7, this query expression contains three terms: (T_1 in 'Author' field), (T_2 in 'Title' field), and (T_3 in 'Abstract' field). These three terms are combined by two logical operators L_1 and L_2. These two logical operators must be equal due to the constraints of the query user interface.

Because each field in Figure 4.7 can only be limited by a specific term modifier, when we translate Q_1 into the format supported by Figure 4.7, the second term T_2 ("Information Integration" in All fields) in Q_1 can only be mapped into three concrete fields in Figure 4.7: <Title>, <Abstract> and <Author>. Therefore, Q_1 can be disjunctivized into three sub-expressions:

$Q_{1,1}$: (T_1) AND (T_2: Title) AND (T_3) AND (T_4);

$Q_{1,2}$: (T_1) AND (T_2: Abstract) AND (T_3) AND (T_4);

$Q_{1,3}$: (T_1) AND (T_2: Author) AND (T_3) AND (T_4).

Apparently, the third sub-query $Q_{1,3}$ contains the term (Authors contain "Information Integration") and this query will retrieve nothing unless someone's name is "Information Integration".

A query and its disjunctivized sub-queries have the same effects, i.e. the retrieved results of these two situations are the same. Sometimes an original disjunctive query can be directly mapped into a target query without being transformed if the target source supports this original query (we will discuss this situation in section 4.2.5: Translation from an arbitrary query into several target query expressions).

4.2.3 Translation of a conjunctive query into a single target query expression

As Table 4.2 and Figure 4.2 display, the query expressions of the mediator and the wrapper contain more than one sub-query expressions. The core of this algorithm is the translation of a conjunctive query in the mediator into a single target query expression in the wrapper of a target source.

In section 4.2.2, we use concrete examples to illustrate the query translation from a meta-search engine to target sources. Now in this section, we will introduce a formal algorithm for the translation of a conjunctive query into a single target query expression and its detailed explanation.

Considering the constraints of field modifiers, term qualifiers and logical operators, the function MatchSubQE(M, S, $*S_i$) (see Line 8 in Table 4.2) judges whether the sub query expression M can match S, or after changing the order of the elements in M, can M match S? If so, return TRUE and $*S_i$ is the reorganized sub-expression; else return FALSE. For the reason of supporting ease of reading, we mention M and S again:

$M = \bigcap_{l=S_p(i)}^{E_p(i)} T_l^m$; /* One of conjunctive sub-expressions of $Q_{mediator}$*/

$S = \Phi_j(Exp_j^s(T_{j_1}\theta_{j_1}...\theta_{j_{k-1}}T_{j_k}))$;

Table 4.3 is the brief description of this function (Figure 4.8 illustrates part of this function excluding the j and k cycles):

Table 4.3 Boolean function MatchSubQE

Algorithm ***Boolean MatchSubQE(M, S,*** *S_i***)**

Input: *a conjunctive query sub-expression (M) of the original user query, a query expression of the selected target search engine (S)*

Output: *it returns TRUE if M matches S (perhaps after reorganization); otherwise it return FALSE. If it returns TRUE,* *S_i *is the point to the reorganized query expression*

01. Boolean tag=TRUE; /*The Boolean variable tag records whether the sub-query mapping succeed */
02. For(int j=1; j<=NumberOf(M); j++) /* The function NumberOf() returns the number of terms in a query expression*/
03. int k=1;
04. While(k<= NumberOf(S) && tag)
05. If ((all modifiers of T_j^m in M) $\in/\subseteq$ (the relevant modifiers of T_k^s in S))
06. If (Empty(*$S_i.T_k$) && (θ_{k-1}^s can be set as θ_{j-1}^m))
07. *$S_i.T_k \leftarrow T_j^m$;

    Modifiers(*$S_i.T_k$)$\leftarrow$Modifiers(T_j^m);

    *$S_i.\theta_{k-1} \leftarrow \theta_{j-1}^m$;
08. Else if ((<Multiple Word>$\in$ Modifier(*$S_i.T_k$)) && (*$S_i.\theta_{k-1}$ == θ_{j-1}^m))
09. Append T_j^m to *$S_i.T_k$;

```
                        Else tag = FALSE;
                Else tag = FALSE;
                k++;
        if(!tag) return FALSE;
return TRUE;
```

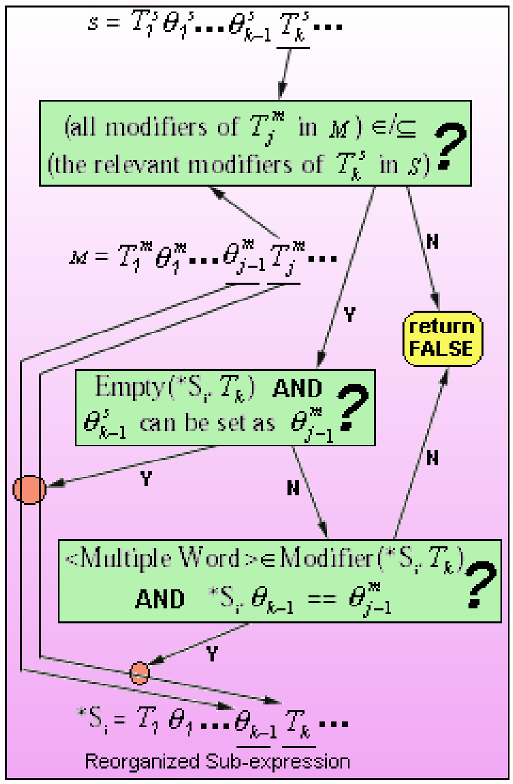


Figure 4.8 Matching query sub-expressions

When we decompose a conjunctive query expression, the split queries inevitably cannot achieve the exact effect of the original query. Therefore, we need to post-process the results according to the previously-employed decomposition information. We use common filters and special filters (their definitions and functions will be discussed later) to record the decomposition strategies and content; later these filters are used to post-process the results. Figure 4.9 illustrates the function Decompose (*M*, *p, **S_i, *q, ** *ST*) (see line 19 in Table 4.2).

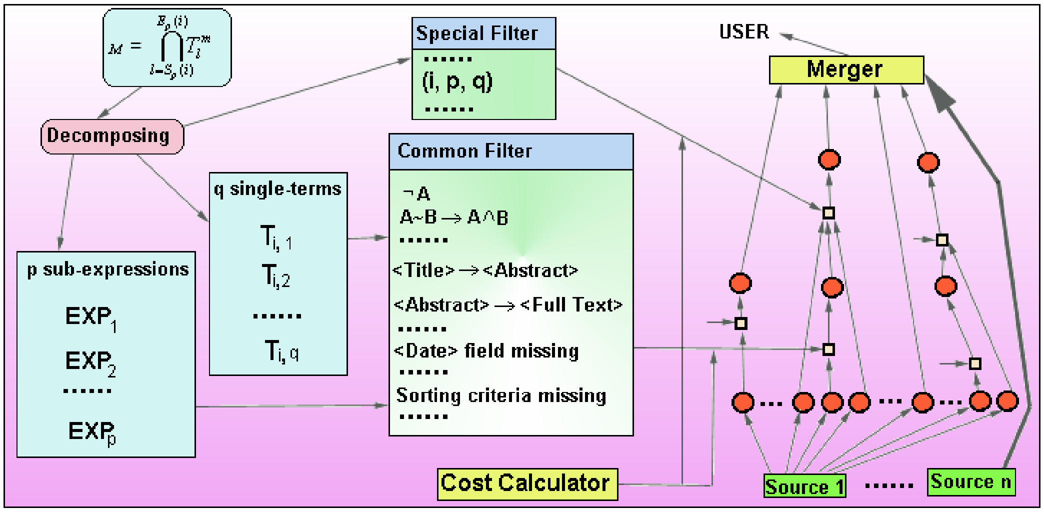


Figure 4.9 Query decomposition and Post-filtering

Some search engines cannot support ¬, ~ or other modifiers; in some cases, we can broaden (relax) the modifiers. Therefore, we insert these logical operators (or modifiers) and the corresponding terms into the CommonFilter_i, and later the system uses this filter to post-process the results.

In Figure 4.9, the conjunctive sub-query expression M of the mediator is decomposed into p (p>=0) sub-expressions (e.g., S_i, S_{i+1}, …, S_{i+p-1}) and q (q>=0) single terms. These p sub-expressions and q single terms can overlap in order to meet the constraints of the selected search engine. We use a SpecialFilter_i (p, q) to record this decomposition information, and later it is used to merge these (p+q) result sets.

Now we discuss the ***MatchSubQE(M, S,*** ***$*S_i$)*** function of the query expressions translation in a more generic way. Suppose that the original query Q^o (see Figure 4.10) is a conjunctive query with m (m>0) terms and (m-1) logical operators (can be 'AND', 'NOT', or 'NEAR') and that the target query Q^t (see Figure 4.11) is a query with n (n>0) terms and (n-1) logical operators. From these two figures, we can see that each term **T** has its own field modifier **FM** and term qualifier **TQ**. Two neighboring terms are combined by a logical operator **L**.

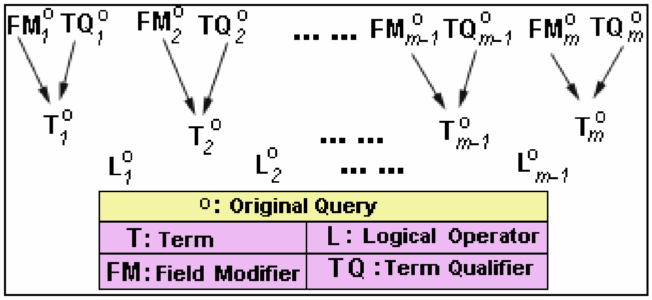


Figure 4.10 the original query Q^o

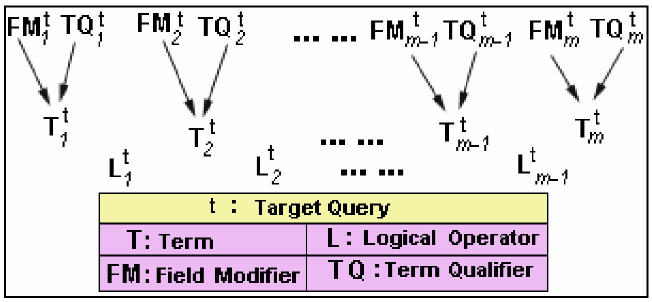


Figure 4.11 the target query Q^t

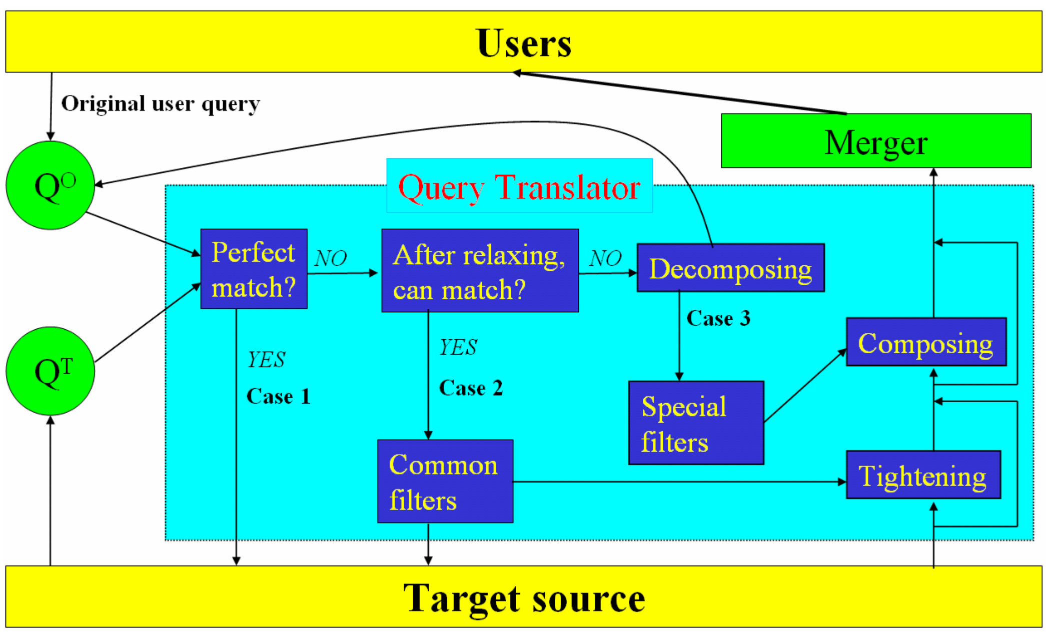


Figure 4.12 Three cases of query translation

When the system translates Q^o into Q^t, one of the following three cases will occur. Figure 4.12 depicts these three cases. In the following we discuss these three cases separately and at the same time introduce how the common filters and the special filters are generated and how they later will be used to post-process the raw results.

Case 1:

In this case, each term in Q^o can be put into a certain term in Q^t, and the field modifier and the term qualifier of this term can also be supported by the corresponding term in Q^t. Furthermore, each logical operator in Q^o can also be supported in Q^t, and the logical value of the new query is equivalent to the original query if some terms exchange their positions. For example, (A AND B AND C NOT D) equals (B AND C AND A NOT D). We call this situation as "Perfect Match" because the results don't need to be post-processed. In the following, we provide a more detailed description of this case.

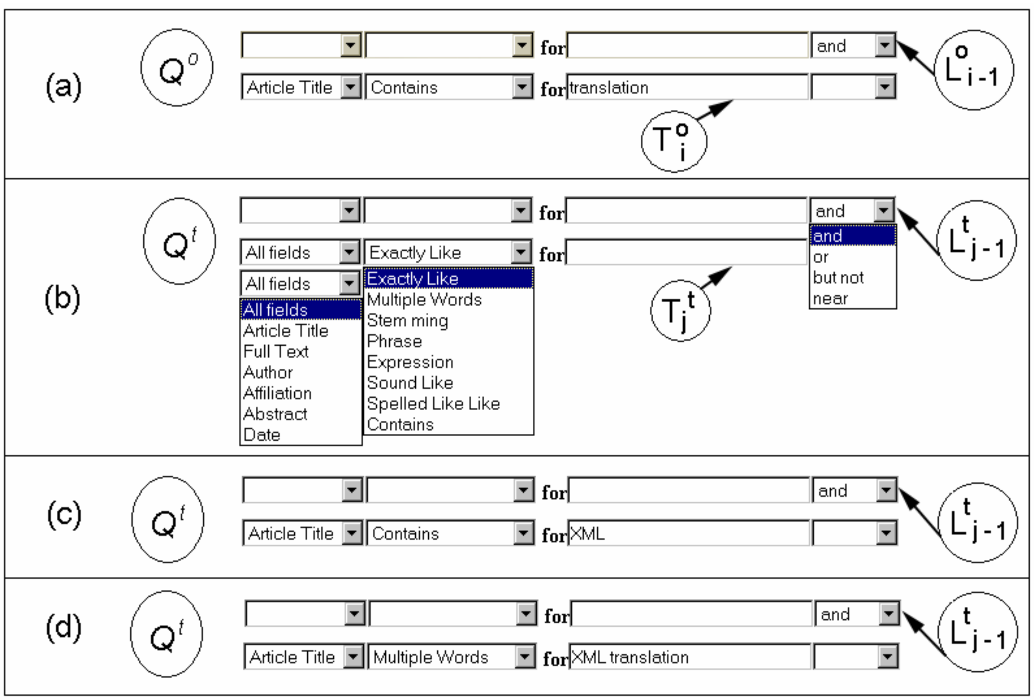


Figure 4.13 Illustration of the "Perfect Match" case

For each term T^o_i in Q^o (see Figure 4.13 (a)), if the field modifier (e.g., in Figure 4.13(a), it is <Article Title>) of this term is one (or a subset) of the field modifiers (e.g., in Figure 4.13(b), they contain <Article Title>) of the corresponding term T^t_j in Q^t (see Figure 4.13(b)), and the term qualifier (e.g., in Figure 4.13(a), it is

<Contains>) of T^o_i is one (or a subset) of the term qualifiers of T^t_j, then we consider the following three situations (otherwise, this case fails):

(a) If T^t_j is empty (see Figure 4.13(b)) and the logical operator L^t_{j-1} supports L^o_{i-1} (e.g., L^t_{j-1} contains L^o_{i-1} <AND>), then the system can put T^o_i into T^t_j, set the field modifier and term qualifier of T^t_j as the corresponding ones of T^o_i, and set L^t_{j-1} as L^o_{i-1}.
(b) Otherwise, T^t_j is not empty (see Figure 4.13(c), this term has already contained "XML" in the <Article Title> field). If the term qualifier of T^t_j supports <Multiple Words> (from Figure 4.13(b), we know that it supports <Multiple Words>) and L^t_{j-1} (in Figure 4.13(c), it is <AND>) equals L^o_{i-1}, then put T^o_i into T^t_j (e.g., in Figure 4.13(d), T^t_j can be two words: XML and translation). This translation is also perfect.
(c) Otherwise, Q^o cannot be translated into Q^t, and this translation fails.

If all terms in Q^o satisfy situation (a) or situation (b), the translation is successful. Otherwise, this query will be transferred to the next stage (Case 2) of the query translation.

If there are still some vacant terms in Q^t, the system can put into each vacant term T^t_k an unused single term T^o_l (we call this process "hitchhiking" the unused single term) if this single term satisfies the following conditions:

(i) the field modifier of T^o_l is one (or a subset) of the field modifiers of T^t_k (or after relaxing it);
(ii) the term qualifier of T^o_l is one (or a subset) of the term qualifiers of T^t_k (or after relaxing it);
(iii) L^t_{k-1} can be 'OR'.

Utilizing these vacant terms in a target query by hitchhiking some single terms will reduce the number of visits to remote sources and therefore improve efficiency.

Case 2:

Some field modifiers, term qualifiers or logical operators in Q^o cannot be supported by Q^t, but after relaxing them (i.e. broadening the scope of the limitation and therefore more results may be retrieved), for example, 'NEAR'→'AND', <Phrase> →

<Multiple Words>, <Title> → <Abstract>, <Keywords>→ <Full Text>, etc., the newly-generated Q^o can be supported by Q^t. In this case, the system dispatches the relaxed query, and when the results come, the system then post-processes the results according to the previous relaxing information. We use <Q> to denote the results by query Q.

Definition 4.3 Common Filters

For the relaxed field modifiers, term qualifiers and logical operators, the system will use some filters to record such information and later use them to refine the results in order to compensate for the relaxing of constraints. We call such filters "common filters" and call the result refining process as "Tightening" (see Figure 4.12).

Now we will discuss some common filters:

(1)'NEAR'→'AND'. Many sources do not support 'NEAR' logical operators. The system can use "A AND B" to replace "A NEAR B" and generates a new "common filter" to record this information. After the results are retrieved, the system uses this filter to select those entries in which term A and term B are near each other (e.g., within 5 words). If term A and term B are in the 'Title' field, this post-processing is easy, but if they are in the 'Abstract' field or even in the 'Full-Text' field, the system will consider the cost of analyzing the content of the source file and the users' patience levels, and then decide whether to post-process it.

(2) 'Phrase'→'Multiple words'. For this case, the system chooses those results that contain the exact phrase. This process is like the relaxing of the "NEAR" to "AND".

(3) 'NOT A'. Some sources do not support the 'NOT' logical operator. When translating the original query, the system discards the 'NOT' operator and its term and generates a new common filter to record this information. After the results come, the system uses this filter to remove the results containing the terms that the original query "NOTted".

Case 3:

In this case, even after relaxing some modifiers or logical operators, Q^o cannot be supported by Q^t.

Definition 4.4 Special Filters

The system will break Q^o into several sub-queries, then translate and dispatch each sub-query separately. We use special filters to record such decomposition information (see Figure 4.12). When the corresponding results come, these "special filters" are employed to compose the results.

However, in most cases, either because we cannot obtain relevant information from target sources or because post-processing will cost unreasonable CPU-time, it is impossible to post-process broken-down conjunctive expressions. For example, suppose that a four-term query is (A *AND* B *AND* C *AND* D) and the target query only supports two terms. Now we decompose the original expression into two sub-expressions (A *AND* B) and (C *AND* D). If the four terms are limited to the "Abstract" field or the "Full-Text" field of the publications, we cannot intersect the two result sets <A *AND* B> and <C *AND* D> because we cannot check whether a term is in such fields (it is not easy to find whether a keyword is in a long PDF/PS document, furthermore, many websites need account/password verification for downloading documents). Even if we can get such information (e.g., by analyzing the PS, HTML, or PDF source file), such strenuous work is unnecessary. If the four terms are in the "title" field of the publications, it is possible to check if each entry from the two result sets contains these four terms. If the post-processing costs a lot of time, the meta-search engine had to directly display the raw results to users because users themselves are the best filters.

4.2.4 Some examples of query translation and post-processing

In the above section, we have described an algorithm for translating a conjunctive query into a single target query expression. Now we use two concrete examples to illustrate this query translation process and show how common filters and special filters are generated and later employed to post-process the results.

Example 1:

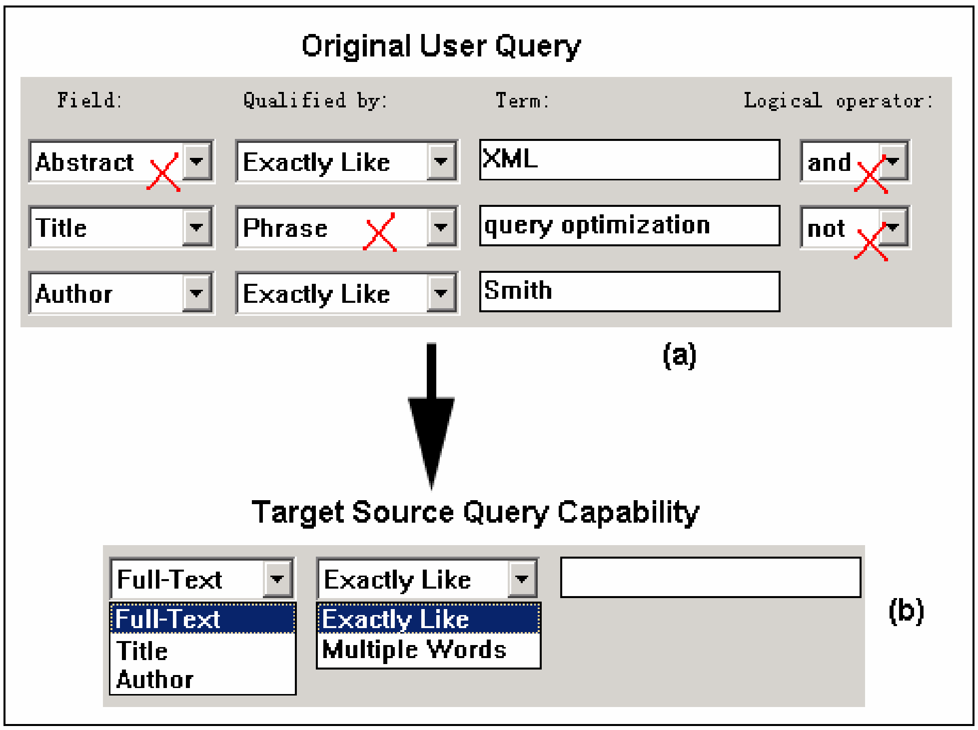


Figure 4.14 First example of query decomposing and relaxing

Suppose the original user query is ((‘XML’ in Abstract field) AND (Title contains phrase “query optimization”) NOT (Author is “Smith”)) (see Figure 4.14(a)). The target source can only support one input-box with three field modifiers (i.e., Full-Text, Title, Author) and two term qualifiers (i.e., Exactly like, Multiple words) (see Figure 4.14 (b)). The target source cannot support the “Abstract” field modifier, the “phrase” term qualifier, and the “AND” and “NOT” Boolean operators. In the following, we discuss how to translate this original user query into the target source.

In Figure 4.15, we use "A" to denote ('XML' in Abstract field), "B" to denote (Title contains phrase "query optimization"), and "C" to denote (Author is "Smith").

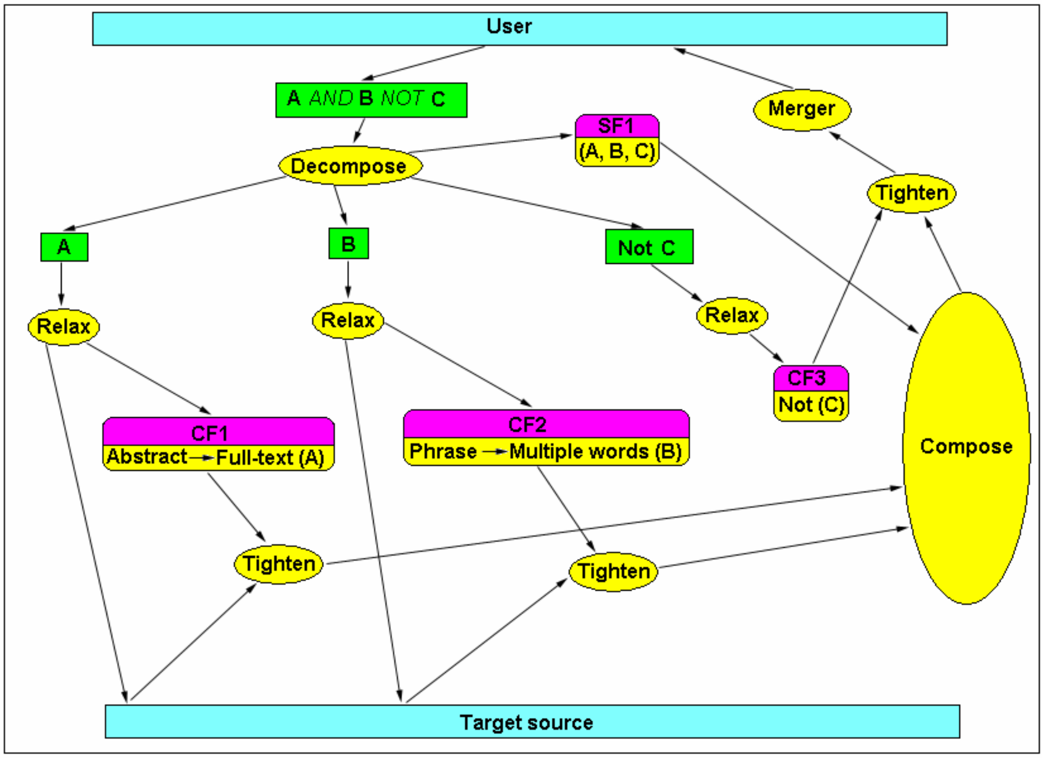


Figure 4.15 First example of query translation

Because the target source can only support one input-box, the original user query will be decomposed (see case 3 in Figure 4.12) into three sub-queries: (A), (B), and (NOT C). A new special filter SF1: "(A, B, C)" will be generated. Later this special filter will be employed to compose the results of these sub-queries.

For the first sub-query (A), because the target source cannot support the "Abstract" field modifier, a new common filter CF1: (Abstract→Full-text (A)) is generated. Then the relaxed sub-query is sent to the target source. When the results of this sub-query return, the system will use the common filter CF1 to refine them, i.e. choosing those documents in which term A (i.e., XML) can be found in the "Abstract" section.

For the second sub-query (B), because the target source cannot support the "Phrase" term qualifier, a new common filter CF2: (Phrase→Multiple words (B)) is generated. Then the relaxed sub-query is sent to the target source. When the results of this sub-query return, the system will use the common filter CF2 to refine them, i.e. choosing those documents in which phrase B (i.e., "query optimization") can be found in the "Title" section.

For the third sub-query (NOT C), because the target source cannot support the 'NOT' logical operator, so this sub-query cannot be sent to the target source. A new common filter CF3: (NOT(C)) is generated and later this filter will be used to post-process the results.

When the results of sub-queries (A) and (B) return, the system will use the special filter SF1 to compose these two result sets by intersecting them. Finally the system will use the common filter CF3 to remove the documents that contain the term C (i.e., "Smith") in the 'Author' section.

Example 2:

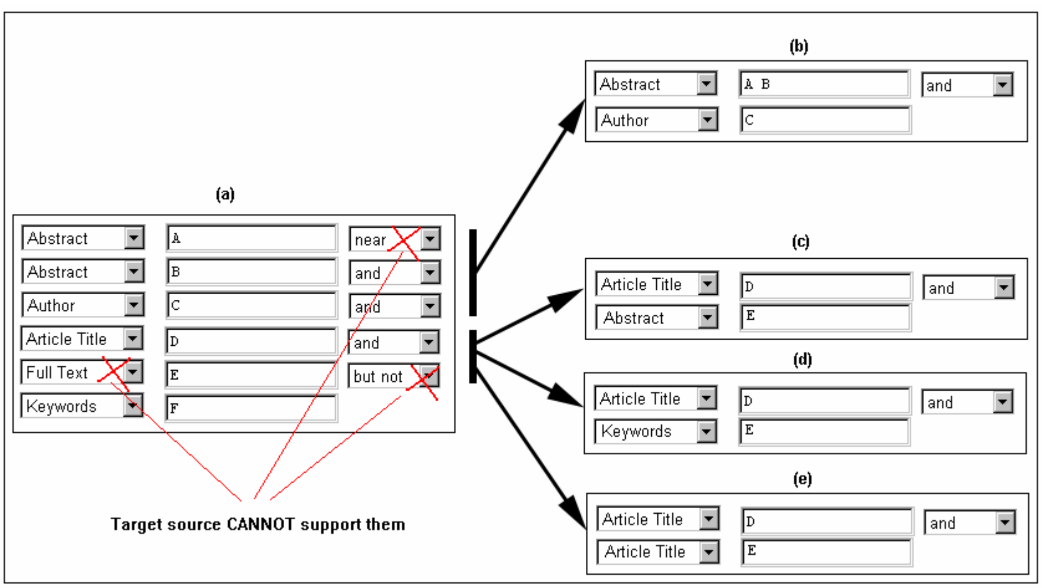


Figure 4.16 Second example of query decomposing and relaxing

Suppose there is a six-term user query Q^o: (A NEAR B AND C AND D AND E NOT F), term A and term B belong to the 'Abstract' field, term C belongs to the 'Author' field, term D belongs to the 'Title' field, term E belongs to the 'Full-Text' field and term F belongs to the 'Keywords' field (see Figure 4.16(a)).

The target query Q^t can only support two terms and each term can only be limited by the 'Author', 'Title', 'Abstract', and 'Keywords' field modifiers, and the 'AND' and 'OR' logical operators (see Figure 4.16(b, c, d, e)).

From Figure 4.16, we can see that the target source cannot support "NEAR" and "NOT" logical operators and the "Full-Text" field modifier.

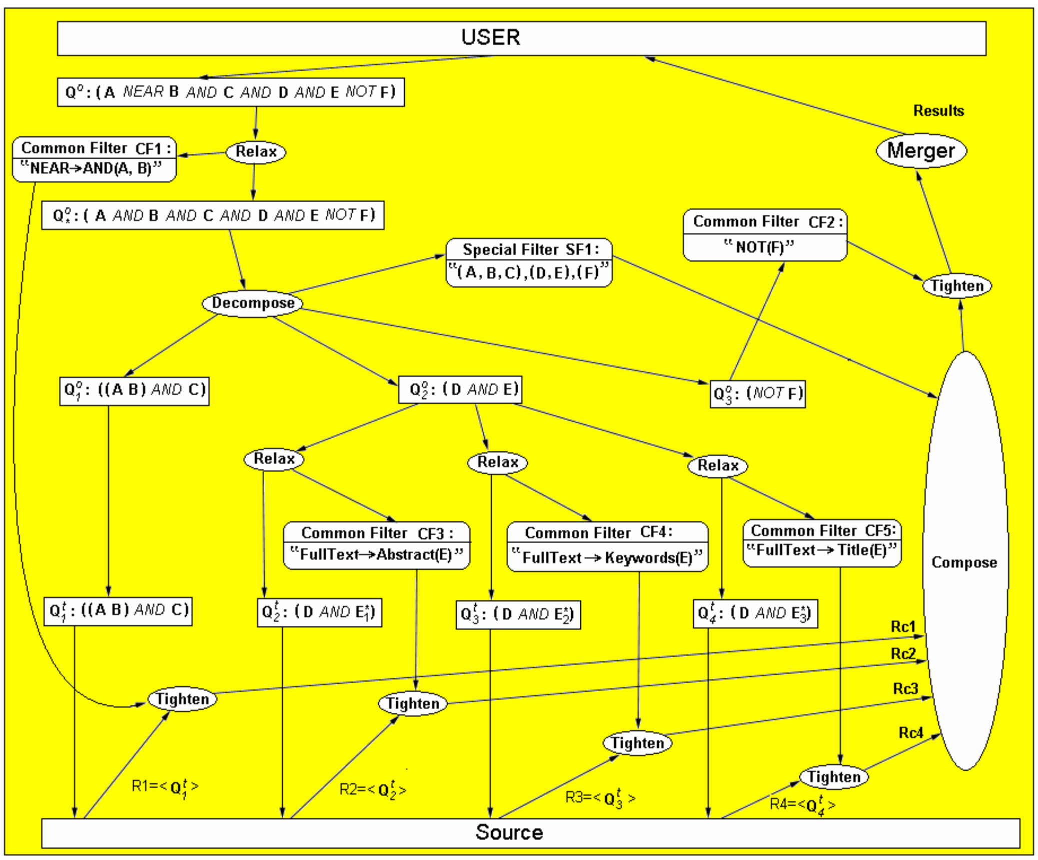


Figure 4.17 Second example of query translation

In the following, we discuss how to translate the original query Q^o to the target form Q^t (see Figure 4.17).

Because the target source (Q^t) cannot support the 'NEAR' logical operator, the system generates a new common filter *CF1: "NEAR→AND(A, B)"* and Q^o becomes (A AND B AND C AND D AND E NOT F).

Because the term A and term B both belong to "Abstract" field, in this query expression, the first two terms "A AND B" can be put together into an input-box of the target source and become "A B". For example, the term A is "XML" and the term B is "RDF", then the new term in the input-box can be regarded as two words: XML RDF (not a phrase "XML RDF".) and the second term is "C".

Because the target source can only support two terms, when the system translates Q^o into Q^t, Q^o will be decomposed (see case 3 in Figure 4.12) into three sub-queries: $Q^o{}_1$: ((A B) AND C), $Q^o{}_2$: (D AND E) and $Q^o{}_3$: (NOT F) and a new special filter *SF1: "(A, B, C), (D, E), (F)"* will be generated. Later this special filter will be employed to compose the three result sets of these three sub-queries.

Because the target source cannot support 'NOT' operator, so the sub-query $Q^o{}_3$: (NOT F) cannot be sent to the target source. Then a new common filter *CF2: "NOT(F)"* is generated and later this filter will be used to post-process the results. In the following, we will discuss how the system translates $Q^o{}_1$ and $Q^o{}_2$ into Q^t separately.

The sub-query $Q^o{}_1$ becomes $Q^t{}_1$ that is in the format of the target source. In this $Q^t{}_1$, the first term is "A B" and the second term is "C".

When the system translates $Q^o{}_2$ into Q^t, because Q^t cannot support the 'Full-Text' field modifier, three new common filters *CF3: "FullText→Abstract(E)"* , *CF4: "FullText→Keywords(E)"* , and *CF5: "FullText→Title(E)"* are generated and $Q^o{}_2$ is transformed into $Q^t{}_2$: (D AND $E^*{}_1$), $Q^t{}_3$: (D AND $E^*{}_2$), and $Q^t{}_4$: (D AND $E^*{}_3$) respectively. In $Q^t{}_2$, the term $E^*{}_1$ belongs to the 'Abstract' field; In $Q^t{}_3$, the term $E^*{}_2$ belongs to the 'Keywords' field; And in $Q^t{}_4$, the term $E^*{}_3$ belongs to the 'Keywords' field.

After that, the system dispatches Q^t_1 , Q^t_2 , Q^t_3, and Q^t_4.

When the results of the query Q^t_1 return (i.e., R1 = < Q^t_1>), the system will use the common filter *CF1* (*"NEAR→AND(A, B)"*) to refine them as Rc1, i.e. choosing those entries in which term A and term B are near each other within a number of (e.g., 3) words. When the results of the query Q^t_2 come (i.e., R2), the system will use the common filter *CF3* (*"FullText→Abstract(E)"*) to refine them as Rc2 (this filter will be skipped because the <Abstract> can almost be regarded as a subset of <FullText>). The common filters *CF4,* and *CF5* are the same as *CF3*. Then the system will use the special filter *SF1* (*"(A, B, C), (D, E),(F)"*) to compose the four result sets Rc1, Rc2, Rc3, and Rc4, i.e. intersecting the four result sets. Finally the system will use the common filter *CF2* (*"NOT(F)"*) to remove the entries that contain the term F in the 'keyword' field.

From these two examples, the two-phase query subsuming mechanism ("Decomposing - Special Filters – Composing" and "Relaxing – Common Filters - Tightening") is further clarified.

4.2.5 Translation from an arbitrary query into several target query expressions

In the above, we have discussed this situation: the source query is a conjunctive query and the target source allows only one query expression. However, some sources provide more than one query form (e.g., in Figure 1.2, NCSTRL supports two forms), thus allowing more than one query expression. Sometimes the target source supports complicated query expressions, so an original query can be directly mapped into the target query without being decomposed (e.g., in Figure 1.3, the ACM-DL source supports disjunctive queries). In the following, we will discuss how an arbitrary query is translated into several queries of a target source.

When an arbitrary query is transformed into several conjunctive sub-queries, the results of these sub-queries and the results of the original query are equivalent. Sometimes an original query can be directly mapped into the target query without being disjunctivized; for example, the user query Q can be expressed in the query forms of Figure 1.6 (the advanced query page of AltaVista search engine supports complex query input), Figure 4.18(a) (query expressions are interpreted sequentially), and Figure 4.18(b) (it supports bracket mechanism). Decomposing a query will increase the number of visits to remote sources and therefore will reduce efficiency. But for most information sources, query decomposition is necessary.

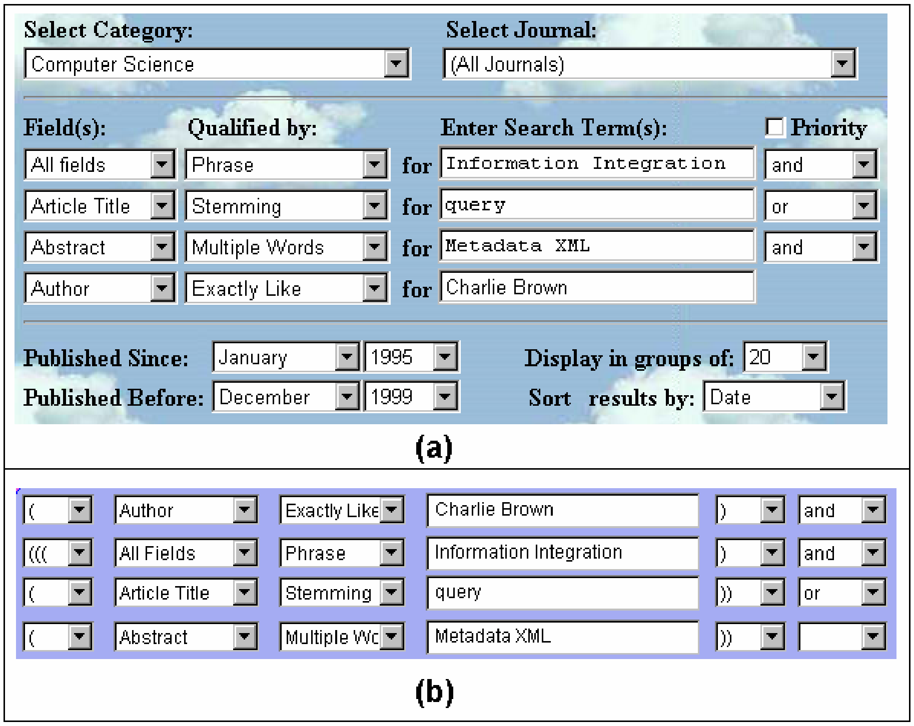


Figure 4.18 Query pages supporting disjunctive query expression

When an original query cannot be directly mapped into the target query, it needs to be disjunctivized. Three situations may occur. (1) The original query supports brackets. (2) The query is interpreted sequentially. (3) The query is interpreted by priority. In the third situation, the query itself is a disjunctive query expression. For the first and second situations, the system needs to reorganize the original query.

4.2.6 Summary of the constraint-based query translation algorithm

Figure 4.19 displays a rough architecture of this query translation processing. When translating the original query into the target query, three steps (i.e., disjunctivizing, decomposing, and relaxing) need to be accomplished, in which depending on the actual situation one or more of these steps may be skipped. When transferring the results from a source to users, three corresponding steps (i.e. tightening, composing, and merging) will be done. The common filters record the relaxing information and later will be used to tighten the results. The special filters record the decomposing information and later will be used to compose the results. The merger will (1) sort and group all results according to certain criteria; (2) revisit search engines (some search engines need to be accessed more than one time to get complete information); (3) dynamically reorganize the displayed results when the results come from some slow-responding search engines; etc.

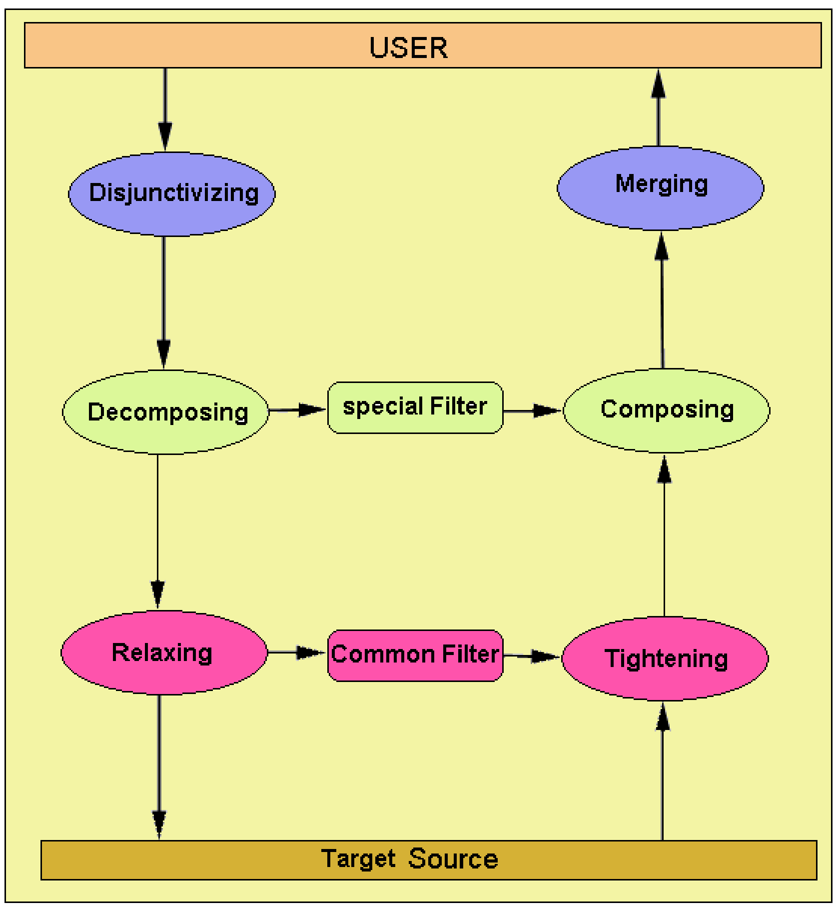
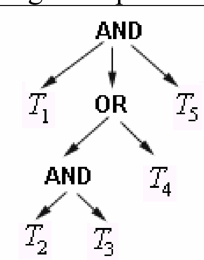


Figure 4.19 Architecture of the query translation

4.3 An example and some problems of query translation

When translating a certain number of disjunctivized sub-queries into several query expressions supported by a specific source, the system maps each sub-query into one of the target query expression that can best support the sub-query. If none of the target query expressions can support a sub-query even after relaxing the term modifiers and logical operators of some terms (see the case 2 in section 4.2), then this sub-query will be decomposed into several smaller sub-queries (see the case 3 in section 4.2). Now we apply this query translation method to the motivating example in section 1.2. The original user query can be denoted as table 4.4.

Table 4.4 The original query *Q*

Term	Value	Field	Qualifier	Logical Operators
T_1	"Charlie Brown"	<Author>	<Exactly Like>	
T_2	"Information Integration"	<Anywhere>	<Phrase>	
T_3	Query	<Title>	<Using Stem Expansion>	
T_4	Metadata XML	<Abstract>	<Multiple Words>	
T_5	[01/1995, 12/1999]	<Date>	<Between>	
Sorting RDC=<Date>; Grouping Size RDC=<20>				

Then, the system rewrites this query as the disjunctive form: $(T_1\wedge T_2\wedge T_3\wedge T_5)\vee(T_1\wedge T_4\wedge T_5)$. Finally, the query translation method is employed to map the original query into the query expressions of the NCSTRL and the ACM-DL. Table 4.5 displays three newly-generated queries for the NCSTRL source and table 4.6 displays two newly-generated queries for the ACM-DL source. These two (or three) queries are not equivalent to the original query and the system needs to post-process the results by means of filters.

Table 4.5 Three newly-generated queries understood by the NCSTRL source

<table>
<tr><td></td><td>Term</td><td>Value</td><td>Field</td><td>LOs</td><td colspan="2">Filter</td></tr>
<tr><td rowspan="3">1</td><td>T_2</td><td>“Charlie Brown”</td><td><Author></td><td>$L_1=\wedge$</td><td rowspan="7">Select hits that are between 01.1995 and 12.1999.</td><td rowspan="7">Merging (1, 2, 3) sorted by date</td></tr>
<tr><td>T_3</td><td>Query</td><td><Title></td><td>$L_2=\wedge$</td></tr>
<tr><td>T_4</td><td>“Information Integration”</td><td><Abstract></td><td></td></tr>
<tr><td rowspan="2">2</td><td>T_2</td><td>“Charlie Brown”</td><td><Author></td><td>$L_1=\wedge$</td></tr>
<tr><td>T_3</td><td>Query “Information Integration”</td><td><Title></td><td></td></tr>
<tr><td rowspan="2">3</td><td>T_2</td><td>“Charlie Brown”</td><td><Author></td><td>$L_1=\wedge$</td></tr>
<tr><td>T_4</td><td>Metadata XML</td><td><Abstract></td><td></td></tr>
</table>

Table 4.6 Two newly-generated queries understood by the ACM-DL source

<table>
<tr><td></td><td>Ter m</td><td>Value</td><td>Field</td><td>Qualifier</td><td>LOs</td><td colspan="2">Filter</td></tr>
<tr><td rowspan="4">1</td><td>$T_{1,1}$</td><td>“Information Integration”</td><td rowspan="2">{<Title>, <Full Text>, <Abstract>, <Keywords> }</td><td><Exact Phrase></td><td>$L_{1,1}=\wedge$</td><td rowspan="4">Select the hits that each title field contains “query” or the words stemmed from “query”.</td><td rowspan="7">Merging (1, 2) sorted by date</td></tr>
<tr><td>$T_{1,2}$</td><td>Query</td><td><Stem expansion></td><td></td></tr>
<tr><td>$T_{2,1}$</td><td>“Charlie Brown”</td><td><Author></td><td><Exactly Like></td><td></td></tr>
<tr><td>T_3</td><td>[01/1995, 12/1999]</td><td><Date></td><td></td><td></td></tr>
<tr><td rowspan="3">2</td><td>$T_{1,1}$</td><td>Metadata XML</td><td><Abstract></td><td><All Keywords></td><td></td><td rowspan="3"></td></tr>
<tr><td>$T_{2,1}$</td><td>“Charlie Brown”</td><td><Author></td><td><Exactly Like></td><td></td></tr>
<tr><td>T_3</td><td>[01/1995, 12/1999]</td><td><Date></td><td></td><td></td></tr>
</table>

In the above, we use the constraint-based query translation algorithm to translate an example query into the query formats supported by two real target search engines: NCSTRL and ACM-DL. On the Web, there are all kinds of domain-specific search engines with various query interfaces and query capabilities. In the following, we will discuss some special situations.

Figure 4.20 Cora search engine

Some sources provide simple user interfaces (only one input-box) and other sources provide complex user interfaces. Therefore, when translating a query from one source to another, one of the following four situations may occur:

(1) Simple → Simple. For most such cases, no translation is needed. However, if translating a query from a simple user interface like Figure 4.20 (it supports modifiers like 'author', 'title', '+', etc.) into a simple user interface that does not support any modifiers, the modifiers of the original query will be removed and therefore some filters will be generated.

(2) Simple → Complex. In most cases, a complex user interface can directly support a simple query. However, if the original query is like the one in Figure 4.20, the query translator may rearrange the order of the query according to the target user interface.

(3) Complex → Simple. If the target simple user interface supports no modifiers, translating a complex query into such a user interface is not so difficult. The system just removes all modifiers from the original query and put terms to the input-box. But more filters will be generated in order to compensate for the loss. If the target source supports modifiers (like Figure 4.20), the modifiers of the original query can be kept. "AND" can be translated into "+" and "NOT" can be translated into "–". If the target source does not support these functions, more filters will be generated in order to compensate for the loss.

(4) Complex → Complex. The algorithm introduced in this chapter is mainly focusing on this case. If the user interfaces of two sources have more similarities, the query translation will be easier and better, i.e. the number of generated sub-queries and filters will be less. Otherwise, the query translation will be complex and the effects will be not good, i.e. the number of generated sub-queries and filters will be more.

In section 7.4, we will introduce a group of experiments that have been carried out to check how many sub-queries and post-filters are generated for three scientific publication-oriented Web search engines. From the experimental results, we can see that: the simpler the user interface, the less sub-queries and more filters may be generated; the richer the user interface, the more sub-queries and fewer filters may be generated. The number of generated sub-queries and filters also depends on both the original query itself and the query capability of the target source.

We can divide the generated filters into three groups.

(1) Some filters can be applied to results post-processing easily and quickly.

(2) Some filters can be applied to results post-processing, but with certain difficulty and it may cost a lot of CPU time. Whether to use these filters or not can depend on users' patience degrees. Therefore, in the user interface of an information integration system, there should be a control for users to input maximal endurable waiting time.

(3) Other filters cannot be applied to results post-processing because it is impossible or the processing cost is very expensive. These filters will be skipped and so the results will inevitably have differences with the original query.

Chapter 5

Adaptive User Interface Generation

In chapter 3, we have discussed the ADMIRE data model that can be used to describe the user interfaces and query capabilities of heterogeneous search engines on the Web. In chapter 4, we have introduced an algorithm for translating queries constructed on the user interface of a meta-search engine into the formats that can be accepted by the various query input user interfaces of heterogeneous target search engines. In this chapter, we will discuss how to construct the user interface of a meta-search engine.

Internet meta-search engines, online catalogues, multi-databases and other kinds of information integration systems have attracted a lot of attention since the advent of the network. The issue of providing a common user interface for distributed networked services can trace back to 70s and 80s, such as [Neg79], [Tol82], [Mor82], [Wil86], [Mar82], [ZSB86], etc. However, this problem remains unresolved.

Most current meta-search engines and information integration systems only use a simple user interface that usually contains only one input-box for users to input keywords and phrases. It may also contain some controls for users to express their domain information needs and the needs for controlling the formats of displaying results. Apparently, it is difficult for users to express the constraint information (such as field modifiers, term qualifiers, and logical operators, etc.) in only one input-box.

Some other systems only list all user interfaces of different sources separately on a page or several hierarchically organized pages. They cannot be regarded as meta-search engines because there is no query translation, parallel multiple targets searching and result merging. Every time, users can only search one target search engine by inputting a query to the copied user interface to this search engine. Therefore, this kind of systems cannot do the automated source selection for users. They let users select one target search engine to search.

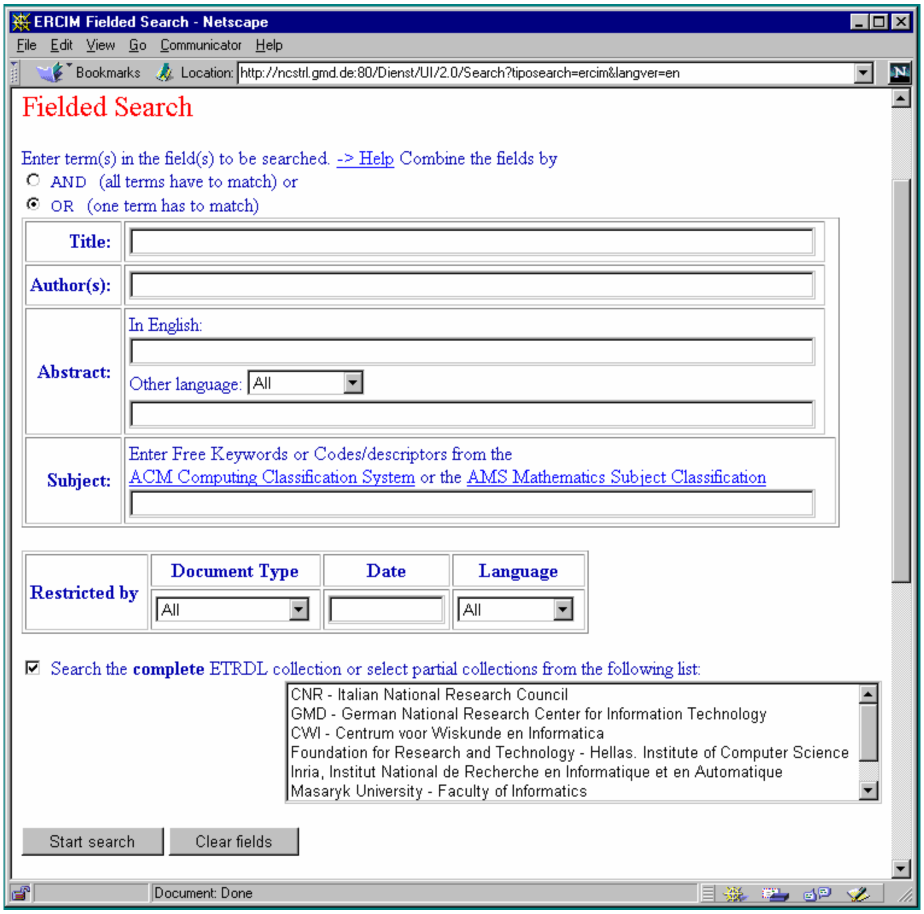


Figure 5.1 the query page of ETRDL

Figures 1.2-1.6 in chapter 1 display that some search engines (especially special-purpose search engines) provide sophisticated user interfaces (several input-boxes, field modifiers, term qualifiers, logical operators, etc.) rather than a simple input-box. For example, the ERTDL (ERCIM Technical Reference Digital Library, See Figure 5.1) has many controls in its user interface. A meta-search engine with only one input-box cannot utilize the rich functionalities of ERTDL.

In [Par99], Park used experiments to suggest "it is important to allow for more user controls in various ways in the distributed environment and to characterize different databases to support user choice for integration". The results in [CGJM02, CC03] also support Park's findings, especially those involving user preference for greater control in database selection and interaction. Apparently, on the user interface of a meta-search engine, the more characteristics of a target search engine a meta-search engine has, the more the meta-search engine can utilize the query capabilities of the search engine.

In order to avoid losing important functions of target search engines, both their generality and particularity should be considered when constructing the user interface of a meta-search engine. Of course, putting all controls of all target search engines into a query input page is impractical; it will make query translation difficult and increase the cognitive loads of users. Therefore, there will be a tradeoff between the utilization of the query capabilities of target search engines and the usability and efficiency of the user interface of the meta-search engine.

There is some research work on providing sophisticated user interfaces for information integration, such as [BW97], [CPW+97], [BKP+98], [MI89], [RRM93], [GR94], etc. However, these systems do not consider the coordination of various constraints among the controls on the user interfaces of heterogeneous information sources.

Considering the great diversity in schematic, semantic, interface and domain aspects, building an efficient user interface for integration purposes is quite difficult. In this chapter, we propose a mechanism of dynamically generating user interfaces based on the adaptive data model introduced in chapter 3. This method can achieve the following advantages that the traditional information integration systems do not have: (1) It will support a progressively self-refining construction of users' information needs; (2) Conflicts among heterogeneous sources can be coordinated efficiently; (3) User queries will match the queries supported by target sources as much as possible.

The remainder of this chapter is organized as follows. First, in section 5.1, we introduce and compare three kinds of user interfaces for meta-search engines. Then, in section 5.2, we define the concept of control constraint rules, which can be used to construct adaptive, dynamically generated user interface of meta-search engines. In section 5.3, we discuss the mechanism of constructing adaptive user interfaces for meta-search engines. Finally, in section 5.4, we introduce the user profiling for meta-search engines.

5.1 What kind of user interface?

Almost all integrated information retrieval systems accessing more than one data source employ uniform user interfaces in order to mask the diversity among heterogeneous sources. Some systems adopt simple user interfaces; all target search engines can easily understand the queries constructed on such user interfaces. Some systems use static, partially mixed user interfaces that have more controls than the simple user interfaces. In contrast with these two kinds of user interfaces, we have built an adaptive, dynamically generated user interface for our meta-search engine. In the following, we will discuss these three kinds of user interfaces separately.

5.1.1 Simple user interface

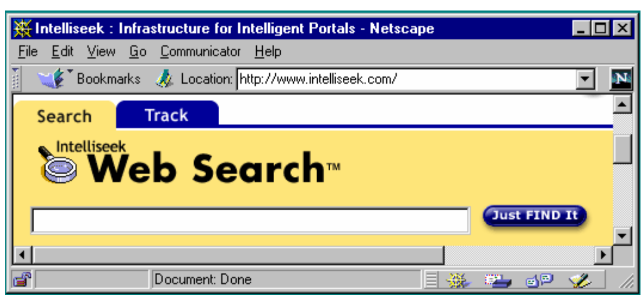


Figure 5.2 an example of simple user interface

Most of current meta-search engines (e.g., SavvySearch, AskJeeves, Dogpile, Highway 61, etc.) provide simple user interfaces. From the query page of the Intelliseek meta-search engine (see Figure 5.2) and those of above-mentioned meta-search engines, we can see that simple user interfaces usually support only one input-box. Therefore, queries constructed from this kind of user interfaces can be easily supported by all target search engines.

We can extend the method of using simple user interfaces to a more general situation – we call it "Greatest Common Divisor" method ("GCD" for short). The concept of "GCD" comes from arithmetic. The greatest common divisor of several integers is the largest integer *g* that all these integers can be divided by *g*. For example, GCD(12, 15, 21) = 3, GCD(6, 12) = 6. A meta-search engine using the "GCD" user interface has *all* the common controls from all target search engines. A simple user interface may not contain *all* the common controls. Figure 5.3 figuratively illustrates the construction of the simple and "GCD" user interfaces for meta-search engines.

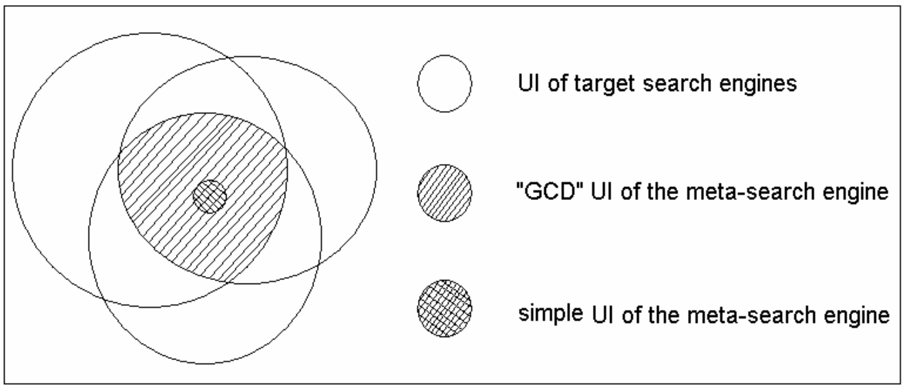


Figure 5.3 Figurative illustration of the "GCD" and simple user interface

The advantage of the simple or "GCD" method is its simplicity both for users when expressing information needs, and for query translation (it is easy for target sources to understand the submitted queries). However, this method will inevitably discard the rich functionality provided by specific information sources, and it is difficult for users to input complicated queries and retrieve more specific information. This weakness is especially obvious when users want to, for example, pinpoint certain scientific publications or specialized information. In section 1.3, we have already pointed out the shortcomings of this kind of user interfaces.

5.1.2 Static, partially-mixed user interface

In order to make full use of the query capabilities of target information sources and improve the precision of retrieved information, some systems adopt static, partially-mixed user interfaces that have more controls than simple or "GCD" user interfaces.

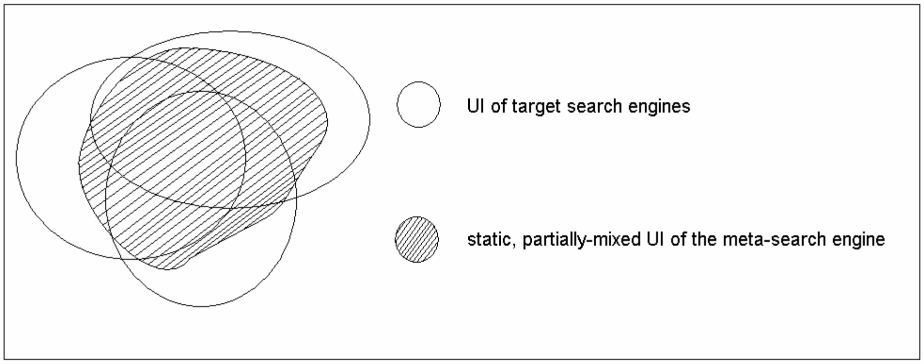


Figure 5.4 Figurative illustration of the static, partially-mixed user interface

Figure 5.4 figuratively illustrates that the static, partially-mixed user interface of a meta-search engine can be constructed based on the major controls (not necessary all the common controls) on the user interfaces of all target search engines. The controls on the user interface of such a meta-search engine may not be contained by each target search engine. Such user interface contains the major controls of target search engines. Figure 5.5 displays an example of such a static, partially-mixed user interface for meta-search engines.

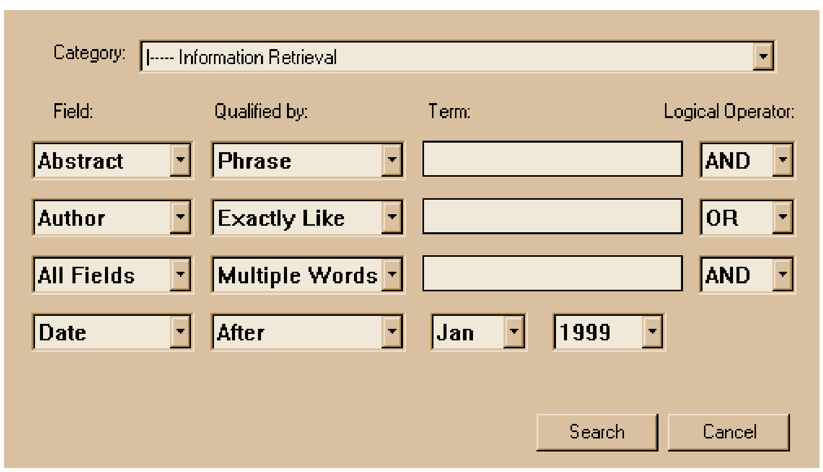


Figure 5.5 an example of static, partially-mixed user interface

Nevertheless, some obstacles need to be overcome by such a static, partially-mixed user interface. The constraints between user interfaces of heterogeneous sources may cause a user query to be inconsistent with a source and make the query translation difficult. In addition, the static user interface lacks flexibility and the interactive nature of IR. For example, some people only need to search a keyword or a phrase, so an input-box will satisfy them. However, some people need to search for very specific information (e.g., the query example in chapter 1) that must be constrained by field modifiers, term qualifiers and logical operators, so they need more controls to express their information needs.

5.1.3 Adaptive, dynamically-generated user interface

In the above two sections, we have discussed two kinds of user interfaces for meta-search engines. Now in this section, we will discuss the adaptive, dynamically-generated user interface that has the advantages and avoids the disadvantages of both the simple (or "GCD") and the static, partially-mixed user interfaces. First we provide a simple example.

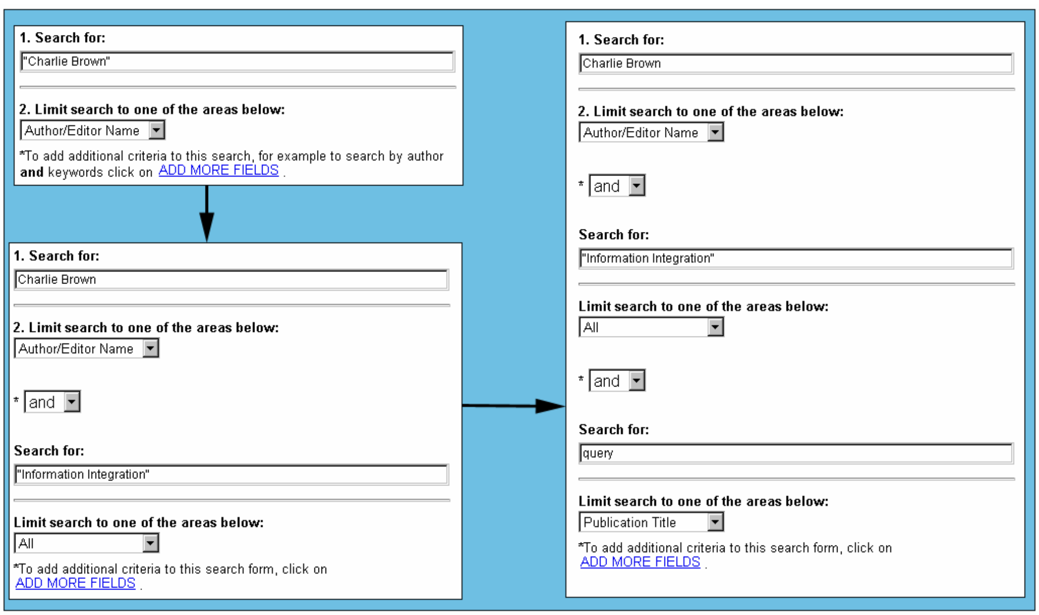


Figure 5.6 the progressive user interface of Elsevier's search service

The search service of Elsevier Science provides a progressive HTML query. Figure 5.6 displays three cropped screenshots of its user interface. The start query page consists of only one input box with a pull-down menu of field modifiers. (There are also Category CSC, Publication CSC, and Grouping Size RDC on the user interface; but these CSCs and RDCs are omitted in this figure.) Besides there is a link "ADD

MORE FIELDS". When users want to add additional criteria to the search, they can click this link, and then the second page appears. This page has two input boxes, two field modifiers and one logical operator. If users need to add more additional criteria to the search, they can click this link many times until they have finished the query construction.

For a progressive query interface, the starting page usually consists of some common controls just as the simple or "GCD" interface supported by most search engines. During the users' query process, the query pages change according to users' needs until the query is finished. Therefore, this kind of query interface has the advantages of both a simple/"GCD" and static, partially-mixed query interface. Sometimes users only need a simple interface to input keyword(s) without any extra controls. But sometimes users want to input complicated queries to search for more specific information.

There is great diversity among heterogeneous search engines, such as:

- With different index and query models (e.g., Boolean models with differing syntax, vector-space/probabilistic model, natural language support, etc.),
- For different domains or subjects (some only index a local repository; some index a certain kind of information on the WWW),
- With different query processing capabilities (some provide rich controls for users to construct complicated queries; some can only deal with simple queries),
- and so on.

Therefore, it is difficult or even impossible for a meta-search engine with a static uniform user interface to coordinate such heterogeneity and to utilize the query functionality of all search engines fully. Therefore, in the following, we provide the concept of progressive Boolean expression expansion that can be used to construct adaptive, dynamically-generated user interfaces.

Definition 5.1 Progressive Boolean Expression Expansion

At the beginning of a query construction dialogue, the system only provides users with a group of query input controls (i.e. a term, its modifiers and a logical operator). If users need to add more additional criteria to the search, they can ask the system to display more controls progressively until they finish the query construction. We call this **Progressive Boolean Expression Expansion**.

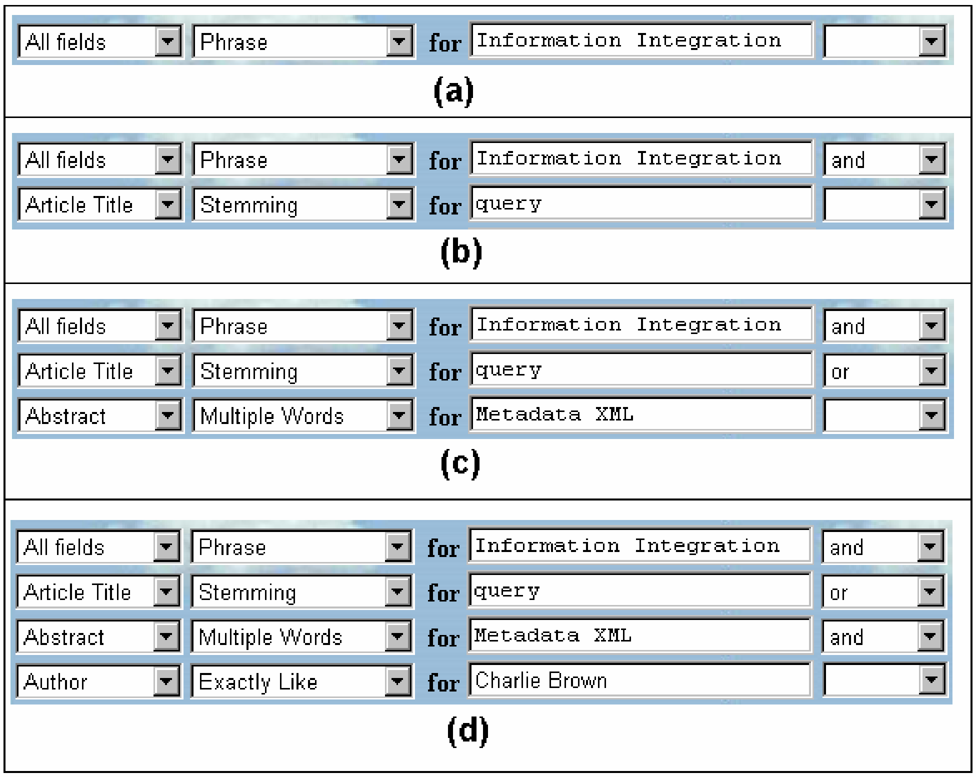


Figure 5.7 progressive query input

For example, Figure 5.7 displays four cropped snapshots of the user interface of a meta-search engine in which an example query has been input progressively. In Figure 5.7(a), the first query term (Phrase “Information Integration” in any fields) has been input. In Figure 5.7(b) and Figure 5.7(c), the second query term and the third query term have been input respectively. Finally, in Figure 5.7(d), this example query has been constructed.

In chapter 7, we will use a group of experiments to test the efficiency of these three kinds of user interfaces. The experimental results show that the adaptive, dynamically-generated user interface can achieve higher precision than the other two interfaces (Adaptive user interface: 58.33%; Static, partially-mixed user interface: 12.28%; Simple user interface: 2.31%).

In the remainder of this chapter, we will first introduce the concept of “control constraint rules” and provide some examples of applying them to the construction of meta-search engines’ user interfaces. Then we will introduce the mechanism of constructing adaptive user interfaces, by which a meta-search engine can make full use of the functionality of various target search engines. Based on this way, the meta-search engine provides users with a dynamically generated user interface that can adapt itself to the concrete interfaces of relevant search engines during the interaction between users and the system.

5.2 Control constraint rules

Due to the great heterogeneity among various information sources on the Web, it is crucial for a meta-search engine to coordinate all kinds of conflicts among sources. In this section, we discuss how to cope with this problem. First, in section 5.2.1, we provide some examples of constraints in the user interfaces of heterogeneous sources. Then, in section 5.2.2, we provide the definition of control constraint rules. Finally, in section 5.2.3, we discuss the applying of control constraint rules to construct the user interfaces of meta-search engines.

5.2.1 Constraints in the user interfaces

There are various constraints among the controls on the user interface of a search engine or user interfaces of different search engines. In the following, we only list some exemplary cases:

(1) **Invalid modifiers for a term.** For example, in Figure 5.8, a term with wildcard cannot use the <Exactly Like> qualifier. If users use the <Exactly Like> qualifier to limit the term "comput*", usually nothing will return.

Figure 5.8 Invalid modifiers for a term

(2) **Incompatible modifiers.** For example, in Figure 5.9, <Date> field cannot be combined with <Sound Like> qualifier. Users can input "Date before January 2000", but what does "Date sounds like January 2000" mean?

Figure 5.9 Incompatible modifiers

(3) **Incompatible classification selection controls.** For example, in Figure 5.10, the <Computer Sciences> category together with the selection of the <Zoological Journal of the Linnean Society> journal will retrieve nothing.

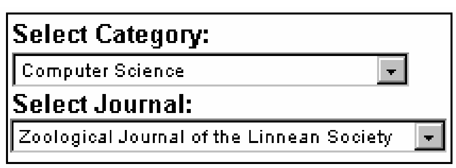


Figure 5.10 Incompatible CSCs

(4) **An item and its descendants are selected at the same time.** For example, in Figure 5.11, in the Category CSC, <Computer Science> and <Hardware> are selected. Since <Computer Science> contains <Hardware>, a search engine or meta-search engine cannot decide for which category the user wants to search. Users can select several items in a classification selection control only if any item is not the descendant

or the ancestor of another item. For example, users can select <Biomedical Sciences>, <Material Sciences> and <Software> at the same time.

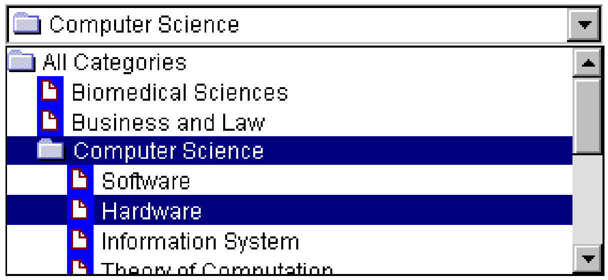


Figure 5.11 an item and its descendants are selected

(5) **A term must be limited by one certain kind modifier or some modifiers.** For example, in Figure 5.12, the second term must be in <Author> field; while the first term can be in <Author>, <Title> or <Abstract> field.

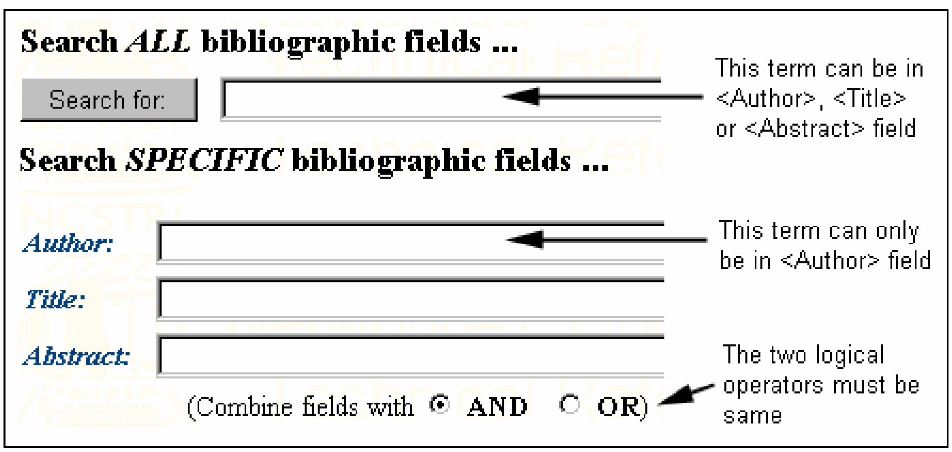


Figure 5.12 Modifiers and logical operators constraints

(6) **Restrictions on logical operators.** For example, in Figure 5.12, the two logical operators connecting the three terms in the second form must have the same value and can only be <AND> or <OR> (There is no explicit logical operators in this query form, but the three fields are logically combined). This means that if the first logical operator is set to be <AND>, the second logical operator will be automatically set to be <AND>.

In the above, we have listed some concrete constraints existing in the user interfaces of search engines. How to describe and coordinate these kinds of constraints? In the next section, we will provide the definition of "Control Constraint Rules".

5.2.2 Definition of control constraint rules

Definition 5.2 Control Constraint Rules

Suppose there are n ($=N_{CSCs}+N_{QICs}+N_{RDCs}$) controls in a search tool (a search engine or a meta-search engine), each control $C_i \in$ CTRLSET, $1 \le i \le n$; and C_i has n_i items (that can be CSCItems, RDCItems, FieldItems, TermQualifiers, or LigicalOperators): $I_{i,1}$, $I_{i,2}$, …, I_{i,n_i}.

If users select p ($p \ge 1$) items from these n controls, there are other q ($q \ge 1$) items from these n controls that must be disabled or must be enabled, i.e. $\boldsymbol{ENABLE}(I_{f(1),\, s(f(1))}, I_{f(2),\, s(f(2))}, \ldots, I_{f(p),\, s(f(p))}) \rightarrow \Psi_1(I_{g(1),\, s(g(1))}), \Psi_2(I_{g(2),\, s(g(2))}), \ldots, \Psi_q(I_{g(q),\, s(g(q))})$; where $\Psi_i \in \{\boldsymbol{ENABLE}, \boldsymbol{DISABLE}\}$; f, g are two single-valued (one-to-one) mapping functions: (f[1, p] $\rightarrow$ [1, n]; g[1, q] $\rightarrow$ [1, n]). s is also a mapping function: $s(k)$ denotes the enabled or disabled item of the kth control. $\Psi_i(I_{k,\, s(k)})$ can also be denoted as $\text{Control}_k.\Psi_i(I_{k,\, s(k)})$. We call it a **control constraint rule**.

Now we explain the meaning of this definition in an informal way. Suppose there are *n* **controls** in the user interface to a search tool, and each control has several **items**. Due to the constraints among the controls of one search tool or among the controls of several different search tools, if users select *p* **items** from these n **controls**, there are other *q* **items** from these n **controls** that must be disabled (users cannot select these q items unless they change their previous selection) or must be enabled (these q items are selected automatically). Take as an example: if the first control (Control_1)'s fifth item ($I_{1,5}$) and the second control (Control_2)'s fourth item ($I_{2,4}$) are selected, then the third control (Control_3)'s first and third items ($I_{3,1}$, $I_{3,3}$) will be disabled and the fourth control (Control_4)'s second item ($I_{4,2}$) will be enabled. We can denote this rule as: {Control_1.**ENABLE**($I_{1,5}$), Control_2.**ENABLE**($I_{2,4}$)} $\rightarrow$ {Control_3.**DISABLE**($I_{3,1}$, $I_{3,3}$), Control$_4$.**ENABLE**($I_{4,2}$)}. We call it a control constraint rule. For example, some control constraint rules are listed in Table 5.1.

Table 5.1 Some control constraint rules

1. Field$_i$.**ENABLE**(<Abstract>)→Qualifier$_i$.**DISABLE**(<Sound Like>, <Spelled Like>, <, >, ≠, ≤, ≥)
2. CategoryCSC.**ENABLE**(<Computer Sciences>)→JournalCSC.**DISABLE**(<Zoological Journal of the Linnean Society>, <Waste Management&Research>, … …)
3. JournalCSC.**ENABLE**(<ACM Transactions on Information Systems>)→Search-EnginesCSC.**ENABLE**(<ACM-DL>)
4. {Search-EnginesCSC.**ENABLE**(<NCSTRL>),Logical-Operator-Control$_1$. **ENABLE**(<AND>)}→Logical-Operator-Control$_2$.**ENABLE**(<AND>)

Disabling some items of a control does not mean the other items of this control will be enabled; it means that users can select one or more of these remaining items of the control or select nothing. Enabling an item means that this item has been selected and will be sent to the target search engine. For example, in the above-mentioned first rule "Field$_i$.**ENABLE**(<Abstract>)→ Qualifier$_i$.**DISABLE**(<Sound Like>, <Spelled Like>, <, >, ≠, ≤, ≥)", if users select the <Abstract> item in the ith field modifier, according to the rule, seven items (e.g., <Sound like>, <Spelled like>, etc) in the corresponding ith term qualifier are disabled. But users can select other items in this term qualifier, such as <Multiple Words>, <Phrase>, <Exactly like>, and so on.

In the second rule, when users select the item <Computer Science> from the CategoryCSC, some items (these journals are not related to computer science) in the JournalCSC will be disabled. In the third rule, when users want to search publications from the journal < ACM Transactions on Information Systems >, the meta-search engine will automatically enable the <ACM-DL> item in the Search-EnginesCSC. In the fourth rule, if users select the NCSTRL search engine in the Search-EnginesCSC and set the first logical operator control to be <AND>, depending on the query capability of the NCSTRL search engine, the second logical operator control will be automatically set to be <AND> too.

5.2.3 Applying control constraint rules

Control constraint rules can be employed to dynamically construct the adaptive query user interface of a meta-search engine. Figure 5.13 displays three cropped screenshots of the meta-search engine using constraints rules to construct its user interface.

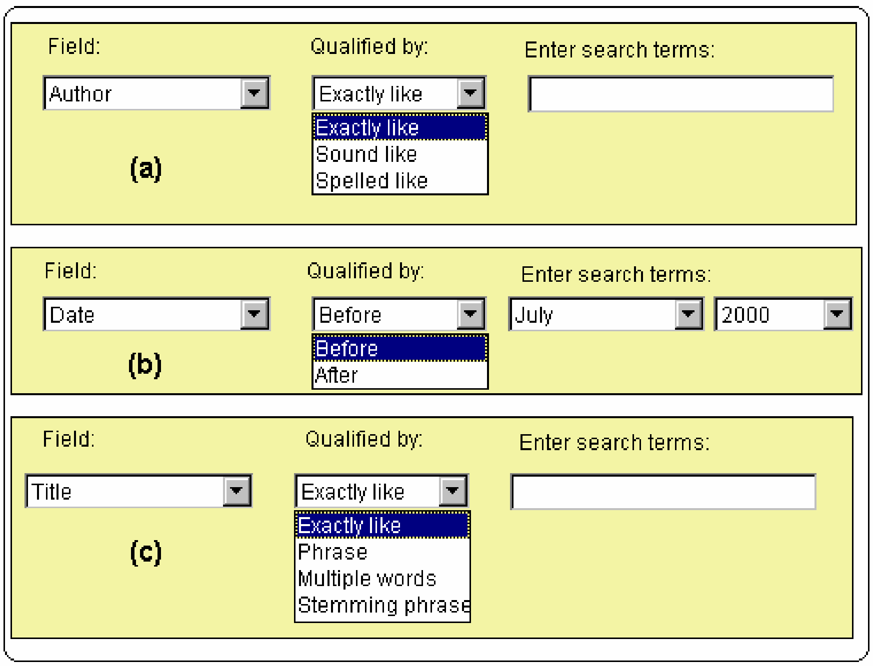


Figure 5.13 Three examples of applying control constraint rules to user interface construction

In Figure 5.13(a), when users select the 'Author' item from a field modifier control, the corresponding term qualifier control will be dynamically changed and it will only contain three qualifiers: 'Exactly like', 'Sound like' and 'Spelled like' qualifiers. That is to mean that the 'author' field can only be modified by 'Exactly like', 'Sound like' and 'Spelled like' qualifiers. While in Figure 5.13(b), the 'date' field can only be modified by 'Before' and 'After' qualifiers and the search terms are two choice controls (one for month and another for year) instead of an input-box. In Figure 5.13(c), the 'title' field can only be modified by 'Exactly like', 'Phrase', 'Multiple words' and 'Stemming phrase'. When queries are typed by users through such a constraints-coordinated query user interface, these queries will be consistent with the query capabilities of search engines. In next section, we will continue to discuss the use of control constraint rules and their applications.

The control constraint rules can be used not only on the above-mentioned QICs (such field modifiers, term qualifiers, and so on) but also on CSCs. Now we provide another example, for the third rule in Table 5.1 "JournalCSC.**ENABLE**(<ACM Transactions on Information Systems>)→Search-EnginesCSC.**ENABLE**(<ACM-DL>)", when users want to search publications in the journal "ACM Transactions on Information Systems" by selecting the corresponding item in the Journal CSC, then the <ACM-DL> item in the Search-EngineCSC will be enabled and the meta-search engine will only search the ACM-DL.

5.3 Adaptive user interface construction for meta-search engines

Different search engines have different user interfaces and query capabilities. Some only provide simple query input interfaces; while others provide complex, advanced query input interfaces. The differences are especially big from search engines on different domains. The user interfaces of search engines on different domains are even completely different. Later on in this section, we will give some concrete examples of information sources on the "car selling" and "weather forecasting" domains. Therefore, it is impossible for a meta-search engine to access hundreds or even more target search engines by using only a uniform, static user interface. A static query input interface lacks the expertise in interpreting and understanding a wide variety of human information needs and using this to formulate and reformulate suitable queries for the retrieval system. When users need very specific (highly specialized) information in the domain, simple user interfaces will be very ineffective. An effective solution of tackling this problem is to construct adaptive query input user interfaces for meta-search engines. That is based on the following two phenomena: (1)

a query is focusing on a domain or subject; (2) the query input user interfaces of search engines on the same domain have many similarities.

When users express their information needs by clicking the controls on the query input user interface of a meta-search engine, the meta-search engine will dynamically construct its user interface by using the control constraint rules introduced in the last section. Before discussing how to adaptively construct the user interface, we first introduce the concept and usage of the "Dynamic Status Transition Table". A dynamic status transition table is used to record the information on user manipulations and user interface status. When a user finishes an action of clicking an item in a control, the system will check the status of all controls by consulting such a table and change the user interface according to the control constraint rules. Without a table to record and manage the information on user interface status and user manipulations, the dynamic interface may be inconsistent with some control constraint rules or even in disorder when the user interface has been changed a lot. This dynamic status transition table can also be used to help users to move back to a former status (just like the "Undo" / "Redo" commands).

When users gradually express their information needs by manipulating the controls (esp. classification selection controls) in the user interface to a meta-search engine, the number of search engines that can satisfy the information needs of users may decrease (according to the source mapping table (see Definition 3.8 in chapter 3), the searching range is limited to one certain domain or subject.

Suppose that only some of the integrated search engines may be relevant. When dynamically constructing the next-stage query page, the system need not consider the irrelevant controls and items that cannot be supported by these search engines. For example, in Figure 5.14, if the first z sources are relevant, the meta-search engine will synthesize the characteristics of these selected sources when generating the query page. In the following, we give a formal description in Definition 5.3.

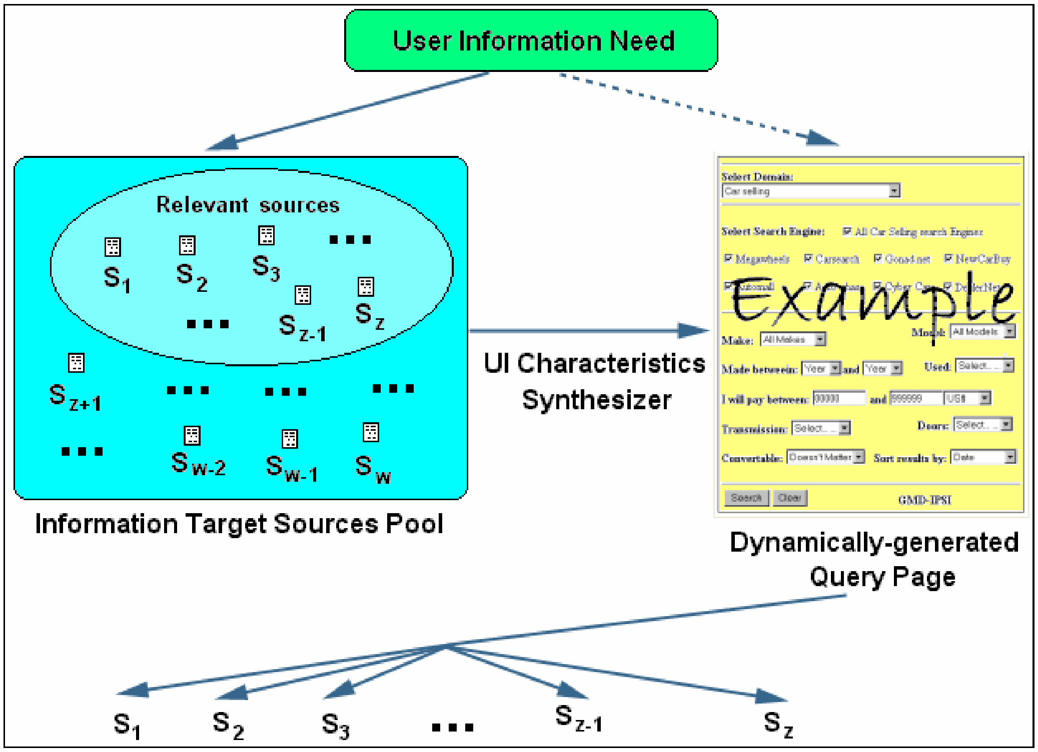


Figure 5.14 Adaptive query user interface construction

Definition 5.3 Adaptive User Interface Generation

Suppose that z search engines of all the w ones ($z \leq w$) may provide results relevant to a user query, when dynamically constructing the next query page, the system need not consider the irrelevant controls $(\bigcup_{i=1}^{w} CTRLSET(S_i) - \bigcup_{j=1}^{z} CTRLSET(S_{f(j)}))$, where the single-valued (one-to-one) mapping function f:[1, z]→[1, w]. The organization of the query input controls will be in line with these z search engines' query expressions: $Q_{f(1)}$, $Q_{f(2)}$, …, $Q_{f(z)}$.

In the following we provide an example. If users select the NCSTRL search engine in the search-engines CSC of the meta-search engine, the user interface of the meta-search engine will be generated like that of the NCSTRL search engine (see Figure 5.15).

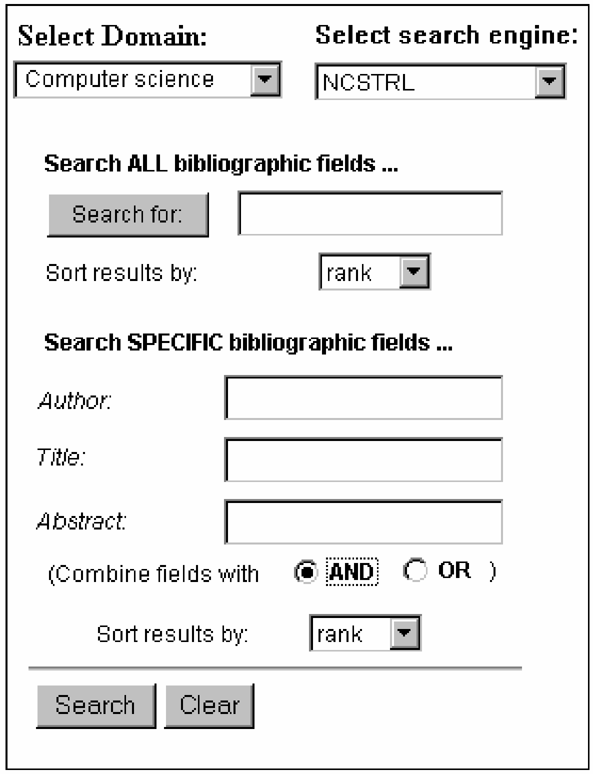
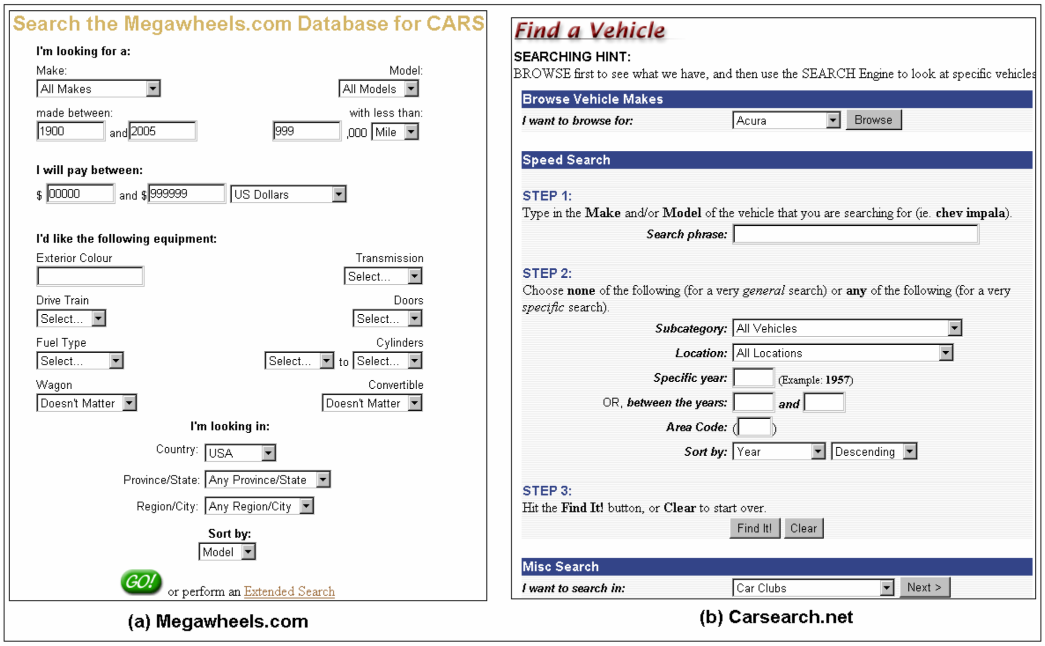


Figure 5.15 Adapting user interface to NCSTRL

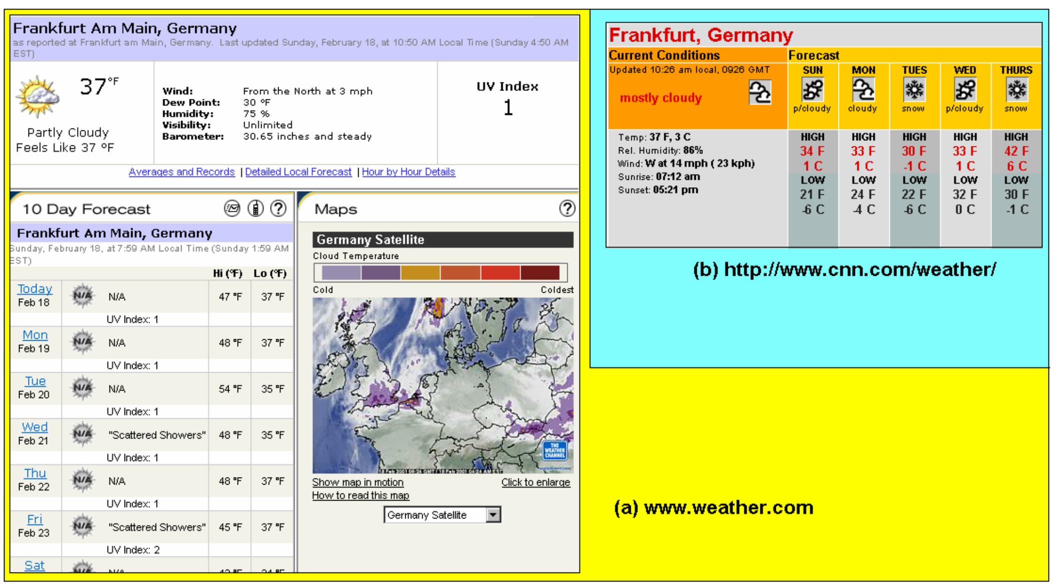
Synthesizing an integrated interface will coordinate the conflicts arising from heterogeneous sources with differing query syntax. There are many differences between the user interfaces and query models of search engines for different domains. For example, it is difficult for a meta-search engine to provide a uniform interface that can be efficiently used by users searching for information on movies, news and architectural engineering.

Figure 5.16 shows the user interfaces of two Web sources for finding vehicles. Their interfaces are quite different. Figure 5.16(a) displays a rich-function query interface in which users can set many specific parameters for searching. Figure 5.16(b) displays three forms (for browsing, fast searching and advanced searching, respectively). Figure 5.17 display two information sources for weather forecasting. These kinds of sources do not provide CGI-based query forms, meaning that users can only browse pages for information. However, these semi-structured pages can easily be queried with the help of wrappers (see section 6.2.3).



(a) Megawheels.com (b) Carsearch.net

Figure 5.16 Two websites providing information for finding vehicles



(b) http://www.cnn.com/weather/

(a) www.weather.com

Figure 5.17 Two websites providing information for weather forecasting

Each time when users execute a query, their information needs are directed towards a certain domain or subject. In addition, search engines for similar domains have many similarities in their user interfaces, for example, Figures 1.2, 1.3 and 1.4 (they share fields such as ‘Author’, ‘Title’, ‘Abstract’, etc.), Figure 5.16 (fields such as ‘Makers’, ‘Models’, ‘Cylinders’, etc.), Figure 5.17 (fields such as ‘Temperature’, ‘Relative humidity', 'Wind', etc.). Therefore, when users select a certain domain or category, the user interface of the adaptive meta-search engine will be constructed according to the characteristics of this domain or category.

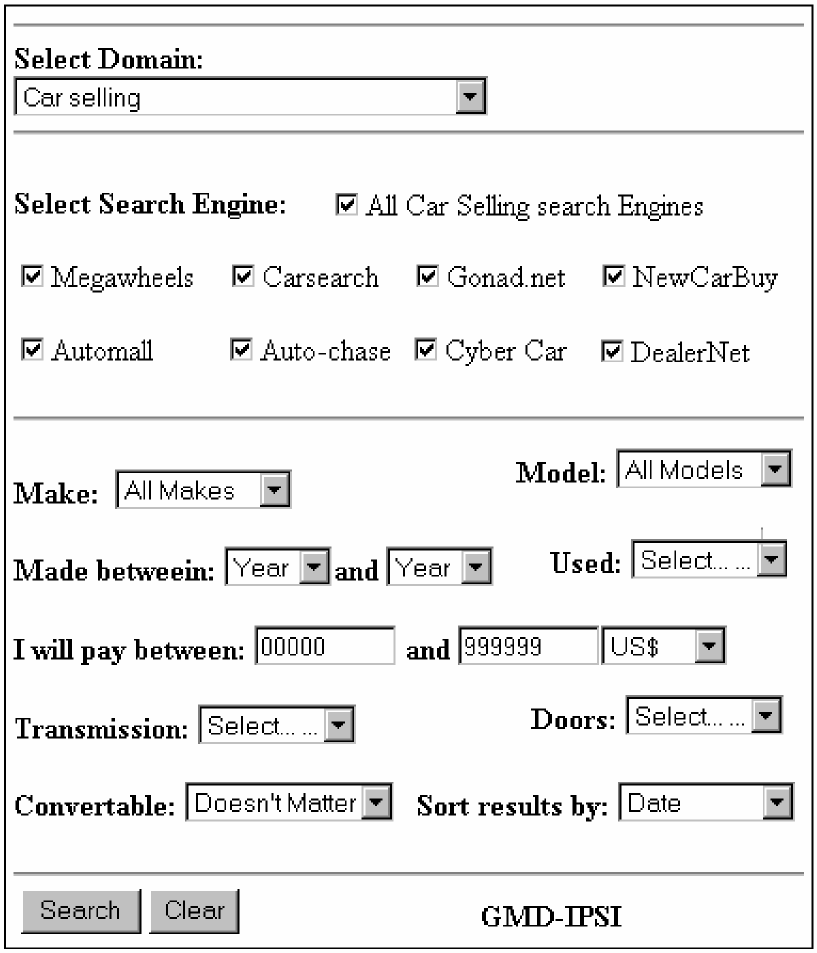


Figure 5.18 Adapting user interface to Car-selling domain

For example, when users select "Car selling" domain, the user interface of the adaptive meta-search engine will be displayed as Figure 5.18. This user interface is mainly constructed by extracting common characteristics (e.g., 'Maker', 'Model', 'Made year') from the search engines of "Car selling" domain (e.g., the two examples in Figure 5.16) and then synthesizing them. Actually, when constructing the concrete user interface of meta-search engine, it is not just a collection of all common controls of search engines in the same domain. For example, if there are 8 sources for "Car selling", seven of them have complex query interfaces and the other one only provides one input box. In this case, the integrated query interface should not contain only an input box. Using simple or "GCD" user interface can satisfy the minority but sacrifices the functionalities of the majority. Therefore, there should be some tradeoffs between the functionalities of different selected search engines. Usually, the benefits of minority will be sacrificed to some extent.

Therefore, based on the adaptive query model discussed before, a meta-search engine can facilitate the expression of both the query capabilities of information sources, and the information needs of users. Such an adaptive information system will have higher flexibility and better scalability than traditional ones.

5.4 User Profiling

Query expressions can be used to describe the domain and schema information of Web search engines and meta-search engines. Since the purpose of a meta-search engine is to facilitate users' information retrieval on the Web, user-centric is one of the main elements for designing the user interfaces of meta-search engines. Now we discuss how to design user profiles and query interfaces from the user point of view.

Because the users of a meta-search engine come from all kinds of application areas, it is favorable for users to be able to define their own personalized view and construct their user profile. For example, some people have interests only in the searching of publications on computer science. In this case, the meta-search engine has to provide users with functionality to customize the query interface, such as selecting some relevant special-purpose search engines and category (domain information) items of some general-purpose search engines. Users can also set up other parameters like sorting criteria, grouping size, layout of result page, file formats of retrieved documents, etc. Customizability is important because it is unrealistic to expect a single generic system to be able to handle all application domains and information seeking tasks as competently as a tailored system.

When a user customizes the user interface of a meta-search engine to a certain extent by restricting domain information needs, the resulting interface will be limited to some specific search engines. Users can also personalize the layout of the controls, the format of results, their patience levels, etc. User profiles are employed to record the configuration set by users and to describe the characteristics of individual users.

An important characteristic of a user profile is the user's knowledge of the subject domain. The information need of a user that is expressed by a query usually relates to a certain application, information, or knowledge domain. Typically, users' information needs are limited to a certain domain over a very short period. On the other hand, the design of the query interface (i.e., controls and layout) is decided by the subject domain information. For example, when users select "Car selling" domain or category in a meta-search engine, the user interface and the query schema knowledge of the meta-search engine will adapt to the characteristics of the domain. There will be no field modifiers for scientific publications such as 'Author', 'Abstract', and so on. The field modifiers will contain 'Maker', 'Model', 'Cylinder', etc. Therefore, the user model and the schema (query) model are closely related to the domain model. Furthermore, a user profile can also be enriched by analyzing the queries of the user.

Chapter 6

Implementation

In chapters 3, 4, and 5, we have discussed the ADMIRE data model, constraint-based query translation algorithm, and adaptive user interface construction for meta-search engines. In this chapter, we will discuss some technical issues about the implementation of our adaptive meta-search engine prototype - SPOMSE. This chapter is structured as follows. First, section 6.1 introduces the overall architecture of our system. Then in section 6.2, we discuss the wrapper generation problem and introduce some existing document extracting tools. In section 6.3, some issues on result merging are discussed. Finally, in section 6.4, we discuss the user interface design of our system.

6.1 Architecture of our adaptive meta-search engine prototype

Based on the ADMIRE data model, constraint-based query translation algorithm, and adaptive user interface construction discussed in the previous chapters, we have built a meta-search engine that is for users to search for scientific publications on the Web. This meta-search engine employs a "Mediator-Wrapper" architecture [Wie92] that has been used by many information integration systems. The mediator manages the meta-data information on all wrappers and provides users with integrated access to multiple heterogeneous data sources, while each wrapper represents access to a specific data source. "Mediator" is sometimes referred to as "broker" or "agent". "Wrappers" are also called "proxies", "adapters", "translators", or "converters". Users formulate queries in line with the mediator's global view that is combined schemas [PHW+99] of all sources. Mediators deliver user queries to some relevant wrappers. Each wrapper records the features (such as input, output, domain, average response time, etc.) of one integrated search engine. Because the information on the Web is always changing, this module will periodically check if the user interface of a search engine has been changed and timely modify the information that describes the query capability and user interface of the search engine. Each selected wrapper translates user queries into source specific queries, accesses the data source, and translates the results of the data source into the format that can be understood by the mediator. The mediator then merges all results and displays them to users.

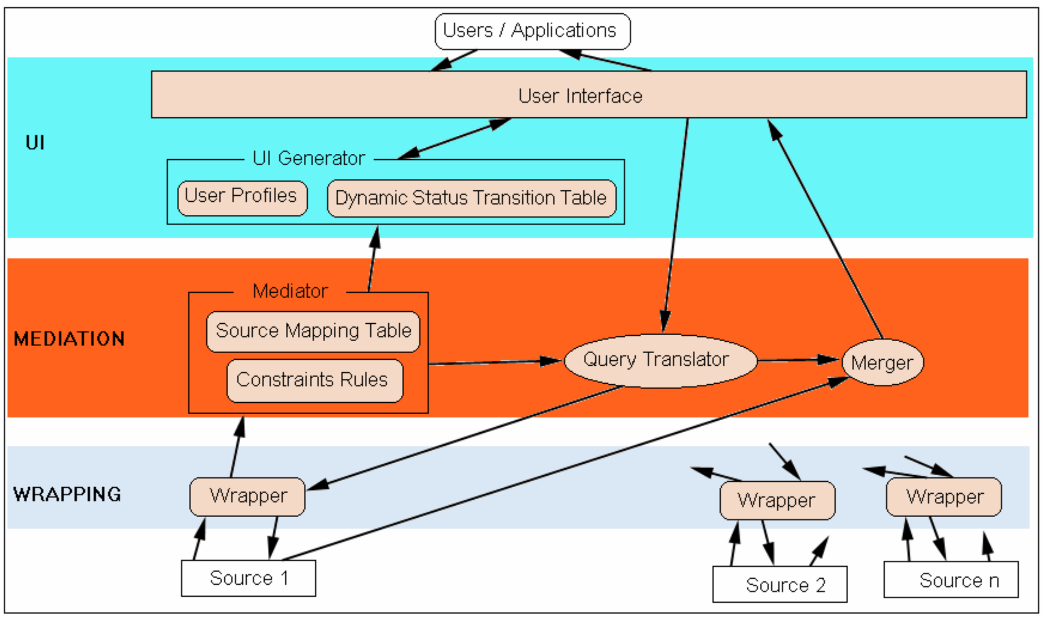


Figure 6.1 Architecture of our adaptive meta-search engine

Figure 6.1 illustrates the architecture of our adaptive meta-search engine prototype. It consists of three layers. The first one is the wrapping layer. Each wrapper is responsible for the communication between the meta-search engine and an information source. The second one is the mediation layer, which acts as an agent between users and the wrappers. The third one is the UI (user interface) layer that dynamically constructs the query form for users to input information needs. Some modules in this figure have been discussed in previous chapters, for example, "source mapping table" (chapter 3), "query translator" (chapter 4), "constraints rules", "dynamic status transition table", "adaptive UI generating", and "user profile" (chapter 5). In chapter 3, we also applied the ADMIRE data model to describe three wrappers for concrete information sources and a mediator for our meta-search engine prototype. We will elaborate the "wrapper" module and discuss some wrapper generation issues in section 6.2. In section 6.3, we will discuss the "merger" module.

The first version of our meta-search engine prototype puts almost all functions (e.g., source selection, query translation, result post-processing, etc.) on the server-site. However, it is very resource consuming (e.g., CPU, Memory, Network bandwidth, etc.) for the meta-search engine server to cope with all these functions. This is especially worse when many users simultaneously submit queries to the server because the query translation and result merging are very CPU time-consuming. Therefore, the response to users' queries will inevitably be slow. In order to lessen the burden of the meta-search engine server, we developed a client-site application.

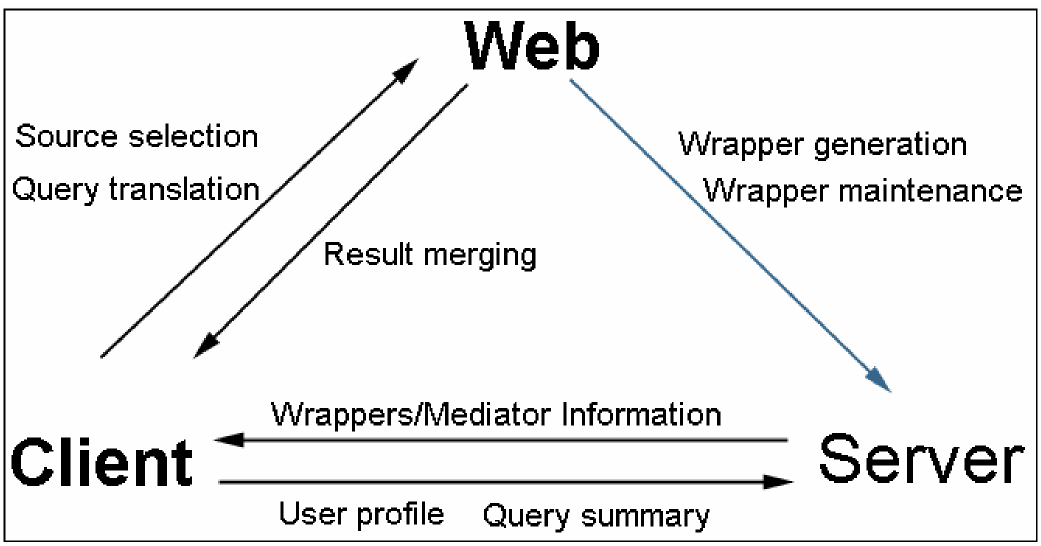


Figure 6.2 Architecture of the client-based meta-search engine

Figure 6.2 displays the architecture of the client-based meta-search engine. The functionalities in Figure 6.1 are spread to the server and many client machines. The server gathers and generates wrappers for Web information sources. It also maintains the wrapper and mediator information. The client gets wrapper / mediator information from the server and the user profile and query summary information can be sent from the client to the server. Each client machine will do the work of selecting source, translating query and merging result. Therefore, this architecture can facilitate both the server and the users.

6.2 Wrapper generation

If there is close cooperation from the remote information sources, building a meta-search engine will be very easy. Actually, there are too many information sources on the Internet and new sources emerge everyday, so it is impossible for all autonomous sources to cooperate with a meta-search engine. On the contrary, most information sources do not like to provide detailed indexing information to outsiders. For our meta-search engine, we extract metadata from the user interfaces and result pages of remote non-cooperative search engines. In chapter 3, we have introduced the ADMIRE data model and used it to formally describe the query capabilities and the user interfaces of heterogeneous information sources. Now in this section, we discuss how we can technically develop wrappers for information sources. A wrapper can do two things: wrapping query input pages and wrapping query result pages. These two things are very different. In the following, we will discuss them separately. First, in section 6.2.1, we introduce query result page wrapping. Then in section 6.2.2, we discuss some issues on query input page wrapping. In section 6.2.3, we discuss the semi-structured, queryable source wrapping problem. Finally, in section 6.2.4, we introduce some existing document extraction methods.

6.2.1 Result page wrapping

In the following, we provide an example for a result page wrapping. Figure 6.3 is a cropped screenshot that was captured when the query: *the phrase "Query Translation" in the title of articles* was input to the query page of ACM-Digital Library.

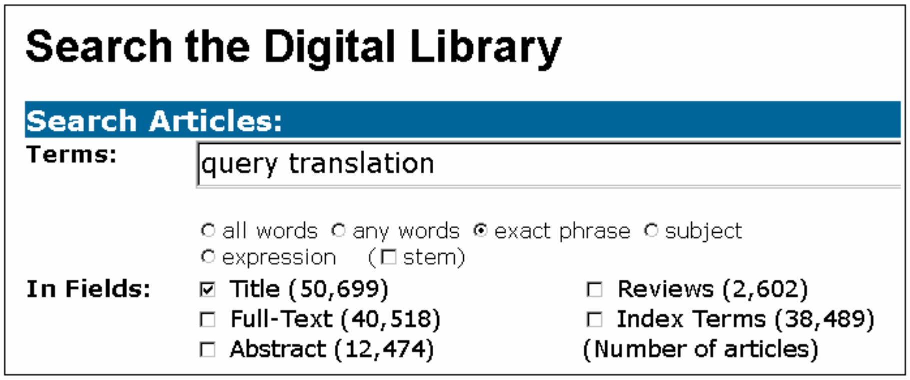


Figure 6.3 the example of query input in ACM-Digital Library

When clicking the "Search" button, the ACM-DL returned two hits (entries) (see Figure 6.4). Usually, the result page of a search engine has some regular structure like:

(head, (entry)*, tail). //* denotes *arbitrarily repeatable*

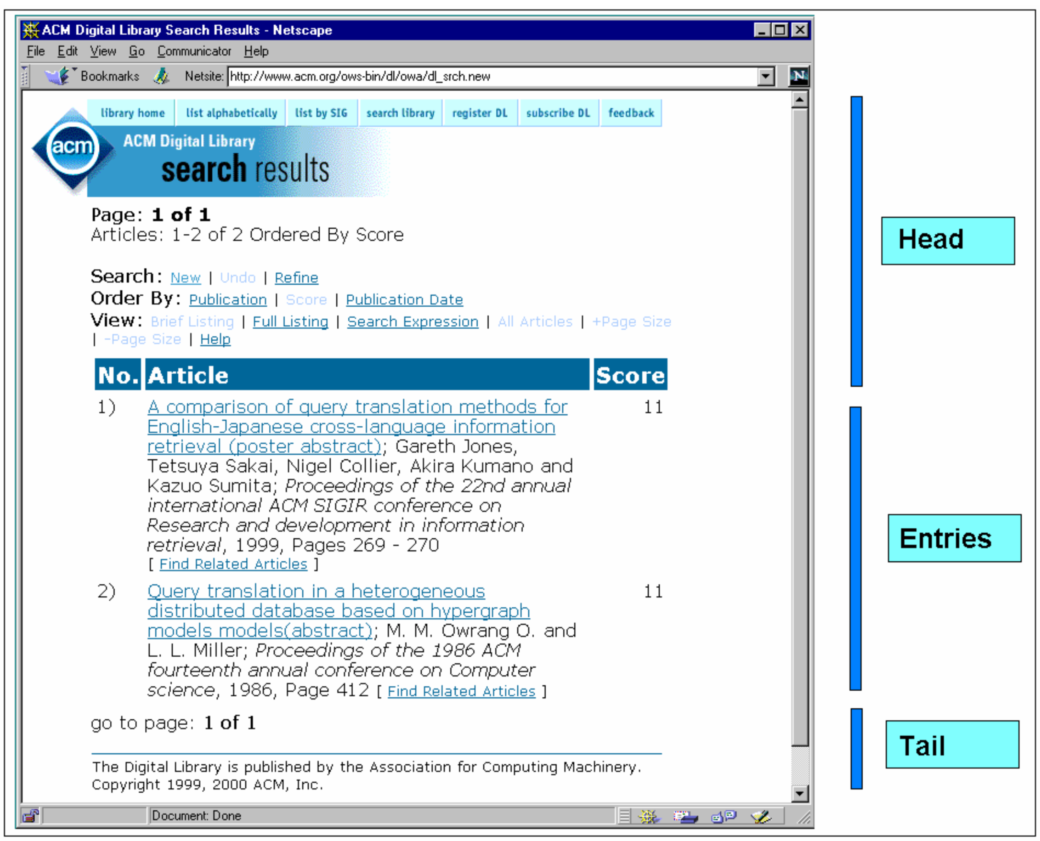


Figure 6.4 the result page of the ACM Digital Library

As Figure 6.4 displays, the meta-search engine needs to extract the information such as “Title”, “Authors”, “Conference/Journal title”, “Date”, “Pages”, “Ranking Score”, etc. In the following, we will discuss how we extract information from such result pages.

The extracting method for such kind of documents adopted in our meta-search engine consists of three steps:

1. Manually extract some landmarks such as HTML tags (<B>, <FONT, etc), labels and keywords (Author, Publisher, etc), and punctuations;
2. Manually generate rules for later parsing;
3. Use the rules to automatically extract information from the same sources.

In the following, we discuss these three steps in detail.

Step 1: Manually extracting some landmarks

In Figure 6.5, an entry from the result page of ACM-DL is analyzed. The original appearance of this entry is displayed in Figure 6.5(a). The html source code of this entry is displayed in Figure 6.5(b). From Figure 6.5(b), we can see that there is a lot of landmark information we can use to extract the information of each field. For example, the publication title of this entry is embedded between {**<A HREF="/pubs/citations/proceedings/ir/312624/p269-jones/">**} and {**;**}. The source file of this entry can be found in {**HREF="…"**>}. We can also find the landmark information for authors, date, and other information.

But for the Cora search engine, each entry does not contain information such as date, book_title (journal/conference proceeding), etc. (see Figure 6.6). This information can be found in the “BibTex Entry” link of each entry (see Figure 6.7).

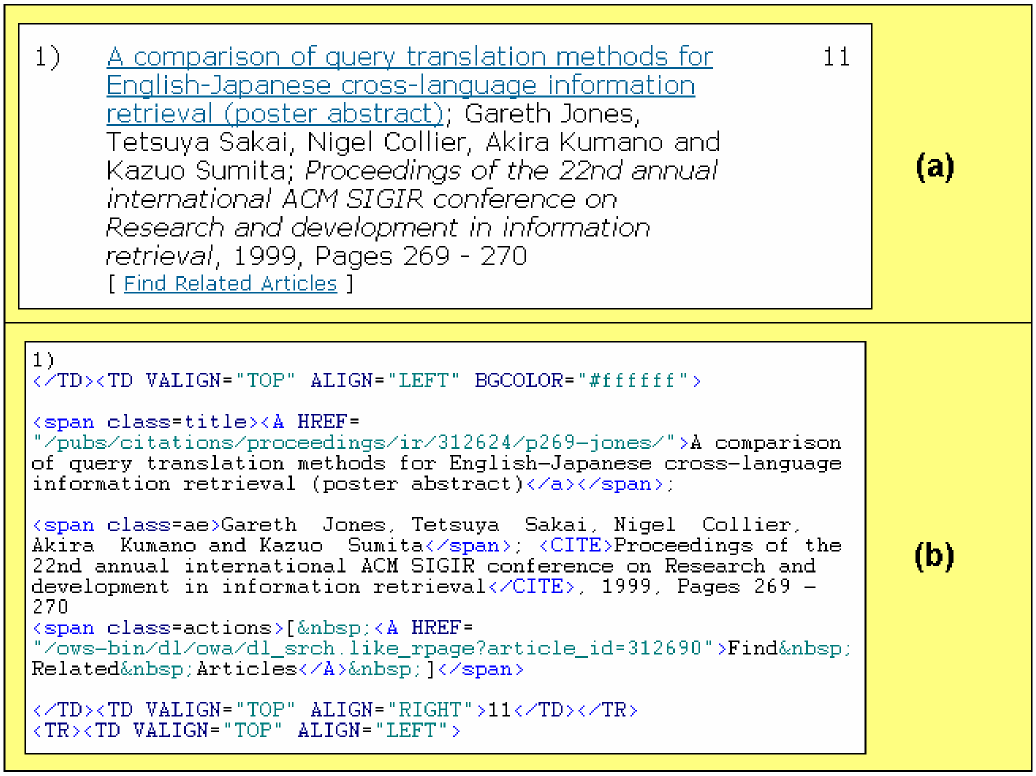


Figure 6.5 analysis of an entry in the result page

Figure 6.6 the result page of Cora search engine

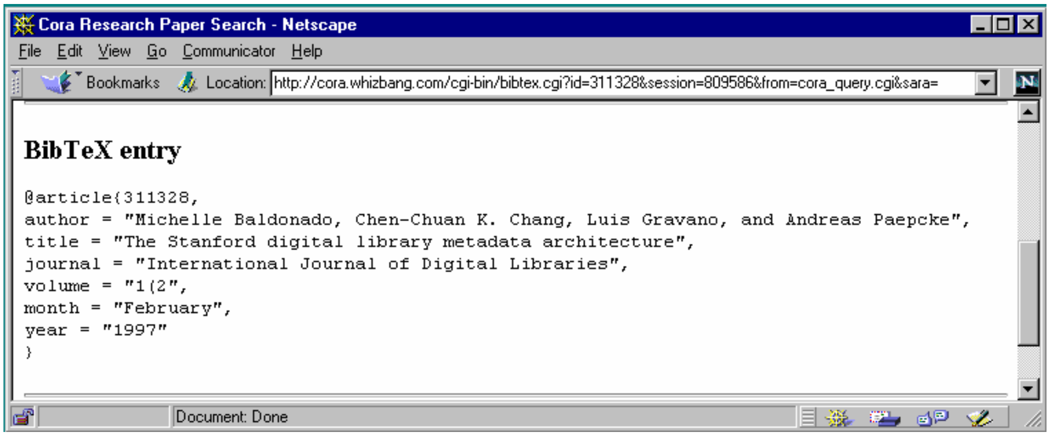


Figure 6.7 the "BibTex Entry" page of a publication in Cora search engine

Step 2: Manually generate some rules for parsing

After identifying the landmark information for each field, we can generate parsing rules for the result page of a search engine. For example, in Table 6.1, we can design the following rules for parsing an entry in the the ACM-DL result page (see Figure 6.5):

Table 6.1 Part of parsing rules for ACM-DL

.*

^'</TD><TD VALIGN="TOP" ALIGN="LEFT" BGCOLOR="#ffffff">

^'<A HREF="' **Source_file** = .* '">' Title = .* '; ' **Authors**= .* '; <CITE>' **Book_title**= .* '</CITE>,' **Date** = .* ', Pages ' **Pages** = .*

^'[<A HREF="' **Related_articles**= .* '">Find Related Articles</A>]'

^'</TD><TD VALIGN="TOP" ALIGN="RIGHT">' **Ranking** = .* '</TD></TR>'

^'<TR><TD VALIGN="TOP" ALIGN="LEFT">'

Note:

(1) We use the word with **bold font** to mean the variable that records field value.

(2) "**Source_file**" records the source file URL. In the ACM-DL, we can only get part of the URL information, such as "/pubs/citations/proceedings/ir/312624/p269-jones/". Therefore, after the rule, we will let the variable "Source_file" be a valid URL by inserting the missing

URL information. Source_file = "http://www.acm.org/ows-bin/dl/owa/dl_srch.new" + Source_file.

(3) ".*" means an arbitrary string.

(4) "^" means the beginning of a new line.

(5) The string between two single quotation marks will be strictly matched.

```
<TD> <FONT SIZE=-1><A HREF=
"bibtex.cgi?id=796538&session=444090&from=cora_query.cgi&sara=">BibTeX Entry
</A></FONT>
```

Figure 6.8 the html source code for "id" and "session" information of an entry

Now, we provide a more complicated example for generating parsing rules for the Cora search engine. From Figure 6.6, we can easily design the rules for extracting the information of four fields: title, authors, abstract, and score. For each entry, we can follow the "BibTex Entry" link to get other information, such as book_title, month, and year (see Figure 6.7). How can a meta-search engine follow this link? It must get the "id" and "session" from the html source code. From Figure 6.8, we can extract the "id" (here is "796538") and "session" (here is "444090") for each entry by using parsing rules. Here, we omit the description of these rules that are similar to the rules in Table 6.1.

Step 3: Use the rules to automatically extract information from the same sources.

After generating the parsing rules for the result page of a search engine, the meta-search engine can use them automatically to extract information from the same source. In the following, we introduce how the meta-search engine uses the generated rules to extract information from the Cora search engine (see Figures 6.6 and 6.7 for the example result pages). Table 6.2 gives a brief description of this procedure.

Table 6.2 using rules to extract information from Cora search engine

```
concept cora_metadata is
attributes id, session, book_title, month, year;
end
```

```
function get_Cora_result
p_cora =Get_Result_page( query expression known by Cora search engine);
all_cora_raw_records =  cora_raw_entries.parse(p_cora);
temp_cora  = cora_metadata();

forall x in all_cora_raw_records do
p_cora = PrefetchStream(URL("http://cora.whizbang.com/cgi-bin/
/bibtex.cgi?id="+x.id+"&session="+x.session+"&from=cora_query.cgi"));

temp_cora = cora_BibTex_entry.parse(p_cora);
x.book_title = temp_cora.booktitle;
x.month = temp_cora.month;
x.year = temp_cora.year;
end
```

6.2.2 Query input page wrapping

For the query input page, the above-mentioned methods seem to be useless because each query input page has its own content and layout. Although there are some similarities among some query input pages such as using the same identifiers (“Author”, “Title”, “Abstract”, etc) and arranging controls in the same order, defining a rule to extract all kinds of query input pages on the WWW is impossible. For example, from Figures 1.2-1.6, we know that there are many differences in various aspects, such as functionality, layout, syntax, semantics, etc. For the query input pages, we manually generate wrappers for each source.

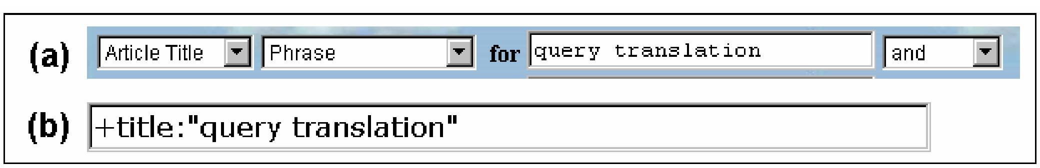


Figure 6.9 the implicit constrains

Some search engines do not provide term modifiers in the user interfaces. But users can find some limiting methods in help files, such as using “+” to mean “must contain”, using “–” to mean “must not contain”, using “title:” before a term to mean that this term will be in the Title field, and so on. Such implicit information cannot be extracted by rules. Administrators will design the converting functions in the wrapper. For example, in Figure 6.9, when users input the query in the user interface of a meta-search engine (see Figure 6.9(a)), the meta-search engine will convert this query to the form in Figure 6.9(b) that can be accepted by the target source.

In the query input page of ACM-DL, there is no result sorting controls. (see Figure 1.3). However, users can sort the results in the result page by clicking the links in "Order By: Publication | Score | Publication Date" (see Figure 6.4). As a meta-search engine, it only presents users with uniform results instead of letting users seeing the raw results of search engines. Nevertheless, meta-search engines can use this information to sort the results of target search engines. The realization of this method (i.e., there is no result sorting control on the query page but on the result page) is quite different from the normal situation (i.e., there are result sorting controls on the query page).

For example, the query for NCSTRL with results sorting intend can be explicitly denoted as the following URL form:

http://cs-tr.cs.cornell.edu/Dienst/UI/2.0/Search/QueryNF?sort=date&keywords=query translation.

In this URL, there is a sub-string "sort=date".

While in the case of ACM-DL, in order to sort the results of this query by date, the meta-search engine must visit the ACM-DL twice. The first submitted URL form of the query is:

http://www.acm.org/ows-bin/dl/owa/dl_srch.new? fields_expr=query expression …

After the result page comes, the meta-search engine will find the URL information in it. Then the meta-search engine will send the refined URL form as:

http://www.acm.org/ows- bin/dl/owa/dl_srch.rpage?search_conid=B100926979&hit_count=2 &page_size=24&page=1&format=c&oby=date

In this URL, we can see that there is a sub-string "oby=date".

In the following, we briefly introduce the data structure for storing information of each wrapper.

```
WrapperInfoOfSearchEngine {
        NameOfSearchEngine      String;
        WebSite                 URL;
        Description             Text;
        NumberOfForm            Integer;
        pFormInfo               **FormInfo;
        pResultInfo             *ResultInfo;
}
```

This data structure describes the basic information of one search engine. This information includes: the name, URL, functional description of the search engine. It also records the number of query input forms, a point array to query forms, and a point to the result page description.

```
FormInfo {
        Action                  String;
        pCSC                    *ClassificationSelectionControls;
        pQIC                    *QueryInputControls;
        pRDC                    *ResultDisplaysControls;
}
```

In the “FormInfo” data structure, the “Action” string records the URL configuring information (such as parameters and formats) for the query action. The “ClassificationSelectionControls” data structure records controls on the query input page for domain, subject, and category selection. The “QueryInputControls” data structure records the terms, the field modifiers, term qualifiers, and logical operators and the constraint information. The “ResultDisplayControls” data structure records controls on the query input page for users to control the formats, sizes, and sorting methods of the query results. In the “ResultInfo” data structure records the formats of result pages (see section 6.2.1).

6.2.3 Semi-structured queryable page wrapping

Due to the heterogeneity of Internet information sources, different kinds of wrappers should be employed for different kinds of sources, such as web search engines (e.g., Altavista), web databases (e.g., Lexis Nexis), online repositories (e.g., digital libraries), other meta-search engines (e.g., Ask Jeeves), semi-structured web documents (e.g., product lists, BibTex), non-structured documents (e.g., Deja UseNet, e-mail installations), and so on. Each wrapper records the features (user interfaces, query capabilities) of an integrated source. In this section, we briefly introduce how a meta-search engine can query semi-structured documents information sources that do not provide query input forms. These semi-structured documents have implicit structure, but not as rigid, static, or regular as standard database systems.

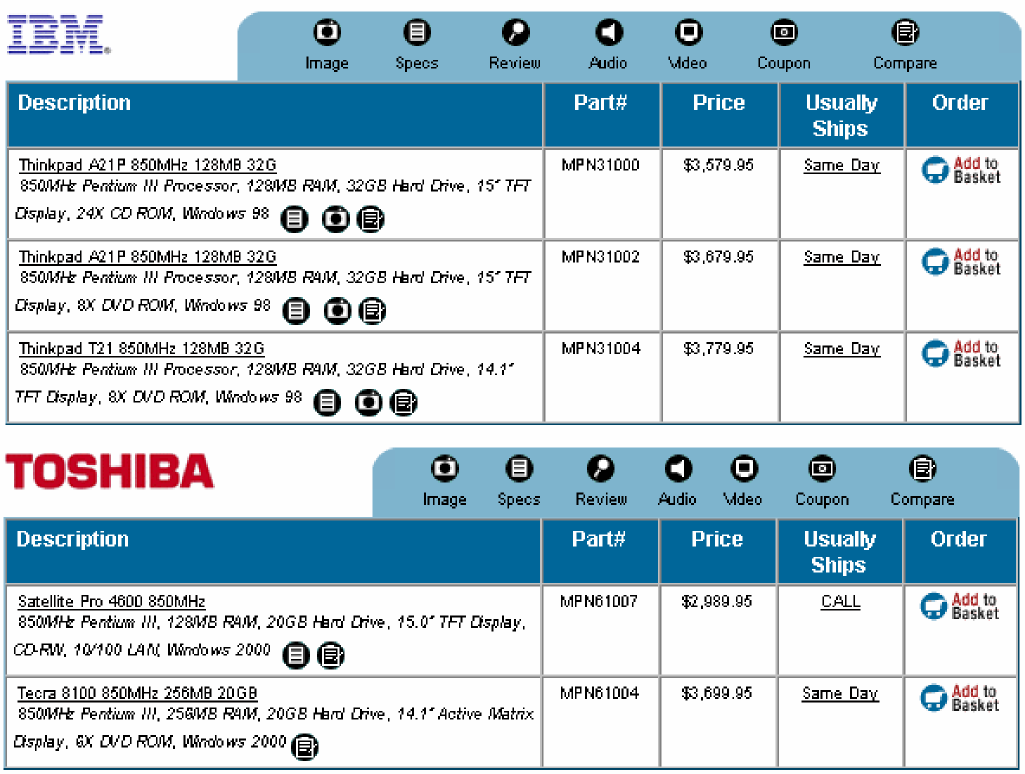


Figure 6.10 An example Web page that can be queried by wrappers

In order to query these semi-structured sources in a database-like fashion on the basis of their underlying structure, it is required to wrap them. In Figure 6.10, a cropped screenshot of a Web page for selling laptop computers is displayed. From this page, we can extract the information of producers (IBM, TOSHIBA, HP, COMPAQ, etc.), descriptions (CPU, Memory and Storage, Display and Graphics, Multimedia, Communications, etc.), Part#, Prices, etc; and then use wrappers to record such information. The meta-search engine can use an SQL-like query language (e.g., "SELECT * FROM 'http://www.companyURL.com/…' where producer = 'IBM' and price < '$2000' …") to express the information needs of users and extract the relevant information from the pages.

6.2.4 Existing document extraction tools

From previous three sub-sections, one can know that meta-search engines need to analyze and extract information from the query input pages and results pages of Web search engines (see Section 6.2.1 and Section 6.2.2) and semi-structured Web documents (see Section 6.2.3) as well. Document analysis and extraction technologies are outside the scope of this thesis. In this sub-section, we just briefly introduce some existing document extraction tools that can be employed by meta-search engines and information integration systems to analyze and extract information from Web documents.

There are many tools that can be employed to extract information from semi-structured documents, such as JEDI [HFAN98]. The wrapper tool of JEDI has a fault tolerant parser. Using attributed, nested rules that describe the source structure of documents, the parser segments them to any desired level, and collates the parsed data into a network of objects. Unlike parsers for formal languages, JEDI's parser can cope with incomplete and ambiguous source specifications. This is accomplished by a parsing technique that chooses always the most specific rule among several applicable rules. When finding no applicable rule for some document portion, it skips as little as necessary to continue with an applicable rule.

In [GRVB98], the wrapper specification toolkit provides graphical interfaces to (1) specify the interface for sources; (2) specify the wrapper capability table; (3) specify the URL constructor; and (4) specify simple and complex extractors. Each of these specifications, e.g., an extractor specification, can be directly tested using the toolkit.

The toolkit will also generate executables, e.g., a wrapper server that can answer queries, or a client applet to accept queries, according to the specified capability of the source. Klein and Fankhauser [KF97] presented an approach to extract the logical structure of text documents. The extracted structure is explicated by means of SGML. It distinguishes three main kinds of structure: logical, syntactic and layout structure. Bergamaschi and Sartori [BS97] presented Description Logics approach for optimizing information extraction and for implementing mediators. The Classifier/Extractor in TSIMMIS [AAB+95] automatically classifies unstructured information and extracts key attributes. The information collected by the Classifier/Extractor can then be exported via a translator to the rest of the TSIMMIS system, together with the raw data. Raghavan and Garcia-Molina [RGM00] proposed a task-specific crawler called the Hidden Web Exposer (HiWE) for extracting content and semantic information from the hidden Web by exploiting visual cues (such as layout, labels, etc.) from the query input pages and response pages.

Freitag [Frei98] presented the SRV, a top-down relational algorithm for information extraction, makes no assumptions about document structure and the kinds of information available for use in learning extraction patterns. It must induce rules to identify text fragments that are instances of some fields. In [DEW97], Doorenbos et al. developed a Web comparison-shopping agent that can automatically build wrappers for Web sites. They make much stronger assumptions about the type of information (i.e., pages containing items for sale) they are looking for and use that information to hypothesize the underlying structure. Their wrapper language is not very expressive and the system is quite limited in terms of the types of pages for which it can generate wrappers. Hammer et al. [HGMN+97] introduced a template-based approach to generating wrappers for Web sources and other types of legacy systems. This approach provides a way of rapidly generating wrappers by example, but it could require a large number of examples to specify a single source.

6.3 Result merging

Because a meta-search engine visits many remote information sources and different search engines have different formats of results, there will be inevitably some problems when merging all results from heterogeneous sources into uniform format. (Result Merging, also known as the Collection Fusion problem). In section 6.2, we have discussed the way we use to extract the information of each field from result pages. We have seen several result pages from different information sources (see Figures 6.4, 6.6). Now, in Figure 6.11, we will compare the entries of four sources: ACM-DL (Figure 6.11(a)), Cora (Figure 6.11(b)), NCSTRL (Figure 6.11(c)), and IDEAL (Figure 6.11(d)). In the following, we will discuss some issues about result merging.

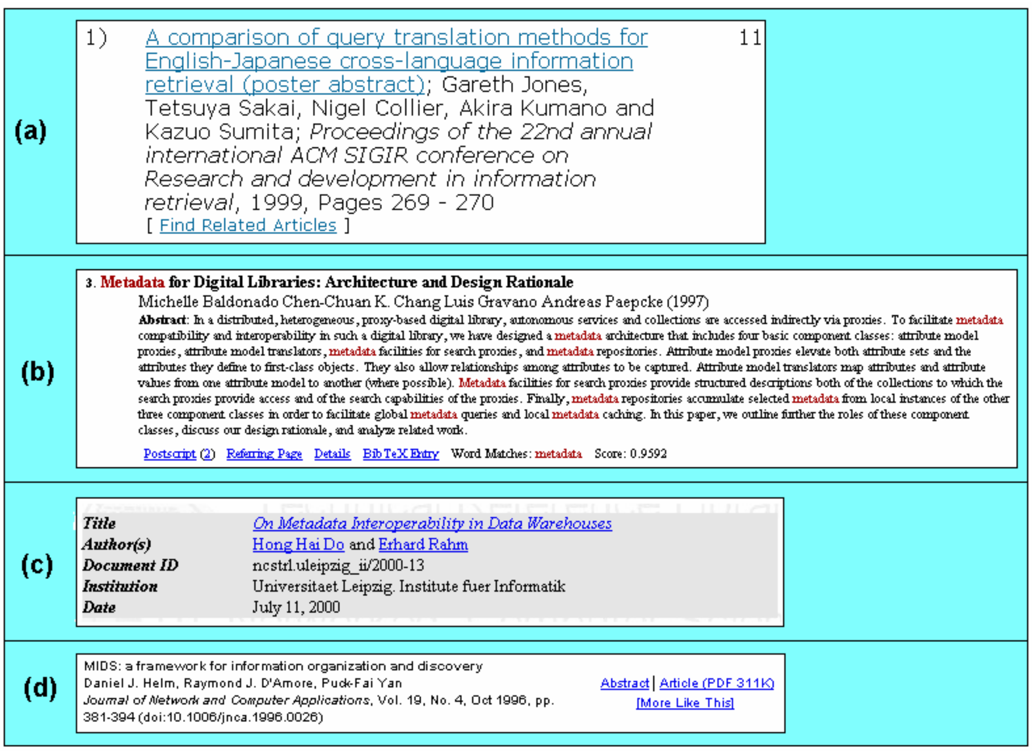


Figure 6.11 Result page comparison

6.3.1 Result sorting

An information source that supports the vector-space retrieval model ranks its documents according to how "similar" the documents and a given query are. Unfortunately, there are many ways to compute these similarities. To make matters even more complicated, the ranking algorithms are usually proprietary to the information source vendors, and their details are not publicly available.

Merging results from sources that use different and unknown ranking algorithm is hard. From Figure 6.11, we can see that ACM-DL assigns an entry the score 11 (see Figure 6.11(a)) and while Cora search engine assigns an entry the score 0.9592 (see Figure 6.11(b)). If we want to merge the results from ACM-DL and Cora into a single document rank, should we rank 11 of ACM-DL higher than 0.9592 of Cora, or rank 0.9592 of Cora higher than 11 of ACM-DL?

The problem of merging document ranks from multiple sources has been studied in the information retrieval field, where it is often referred to as the collection fusion problem. How to compare the scores from different sources? An approach to addressing it is to learn from the results of training queries. Given a new query, the nearest training queries are used to determine how many documents to extract from each available collection, and how to interleave them into a single document rank [VGJL94, VGJL95, and VT97]. Another approach is to calibrate the document scores from each collection using statistic about the word distribution in the collections [CLC95]. In [GGM97], the authors want to guarantee that meta-searchers extract the top target objects from the sources and return these objects ordered according to their target scores.

However, for scientific publication-oriented search engines, it is particularly hard to compare their scores by using the above methods, because each source has its own domain, information coverage, etc. For example, the Elsevier service only provides the publications from Elsevier publisher and ACM-DL only provides the publications of ACM journal and proceedings. Therefore, even though they use the same ranking method, it is difficult to compare the scores from two completely different sources. In this case, if users of our meta-search engine demand the result sorting by relevance, we only display the result sets of different sources separately, with each result set sorted by relevance.

Compared with the results sorting by relevance, sorting results by date or author is easier. For the date sorting, different sources may use different date formats. For example:

(1) ACM-DL only provides year information (1999, see Figure 6.11(a)).
(2) In the results page of Cora search engine (see Figure 6.11(b)), we cannot find date information. But the meta-search engine can follow the "BibTex Entry" to get the date information for a hit. For example, from the linked page in Figure 6.7, the meta-search engine can get the information: Month = "February", Year = "1997".
(3) From Figure 6.11(c), we can see that NCSTRL provides detailed date information for a hit, e.g., Date: July 11, 2000.
(4) From Figure 6.11(d), IDEAL provides date information in the format like: Oct 1996.

In addition, there are other date formats, such as "09/08/00", "08/09/00", or "08092000", etc. The meta-search engine can convert each format into a uniform format and then sort results by date. Some sources only provide year information (e.g., ACM-DL) and some provide month and year information (e.g., Cora, IDEAL). For these two cases, the meta-search engine can assign a medium value to this hit, for example, "1999" = "June 30, 1999", "Oct 1996" = "October 15, 1996", etc.

For the author sorting, some sources provide full-name information; some only provide an initial for first name with last-name information.

6.3.2 Duplicate removing

Hylton provided a good overview of prior work on duplicate detection [Hyl96]. In his discussion of strategies used in the library world for identifying duplicates, he focuses on the OCLC [44] rules, which stipulate that items are duplicates if and only if they have matching-values for various values, including author, title, publisher, etc. The rules specify that some attribute values must match exactly, while others need only match partially.

When the meta-search engine detects that a hit of source A is the same document as a hit of source B, one of these two hits can be removed. However, for the same document, source A may provides some information that source B doesn't have, and vice versa. For example, some sources provide source files, such as PS, PDF file downloading; some only provide abstract information of a document; some provide "Similar papers" information. Therefore, the meta-search engine can integrate all these useful information together. This is an advantage of meta-search engines over search engines.

6.3.3 Dynamical result display

Because the differing response time of search engines, some search engines can return results for a user query within a second, while some search engines can answer a user query after 30 seconds or more. Sometimes due to the network jamming, a search engine may respond slower than usual. If the server is down, a query from the meta-search engine will never be answered by the search engine.

When users issue a query, they want to see the results as soon as possible. Therefore, when the results of one search engine come, the meta-search engine will display them immediately to users. When the results of slow-responding search engines come, the meta-search engine will reorganize the results page according to the sorting criteria users set. The meta-search engine can also provide a patience level selection control for users to set their maximum waiting time.

6.3.4 Visiting source several times

For a user query, one search engine may return thousands or even millions of hits. It is not necessary for a meta-search engine to compute the results from all sources one time. Moreover, users usually only see some hits of all results. Therefore, the meta-search engine can visit a source more times. Every time it only gets a number of results. Fortunately, most search engines provide this results grouping function. For some search engines (Cora), the meta-search engine needs to visit the source more than one time to get some other information (see section 6.2.1).

6.4 User interface design

In the following, we briefly introduce the user interface design of our SPOMSE meta-search engine prototype. Figure 6.12 displays a snapshot of the main window of our prototype. Besides the menu-bar and the toolbar, there are three split-views in this window. Here, the term *view* comes from the Microsoft Visual Studio, meaning a kind of window that is corresponding to a document. The view on the left part is the "Categories" view. The one on the top-right part is the "Queries" View and the one on the bottom-right part is the "Results" view. In the following, we will introduce these three views respectively.

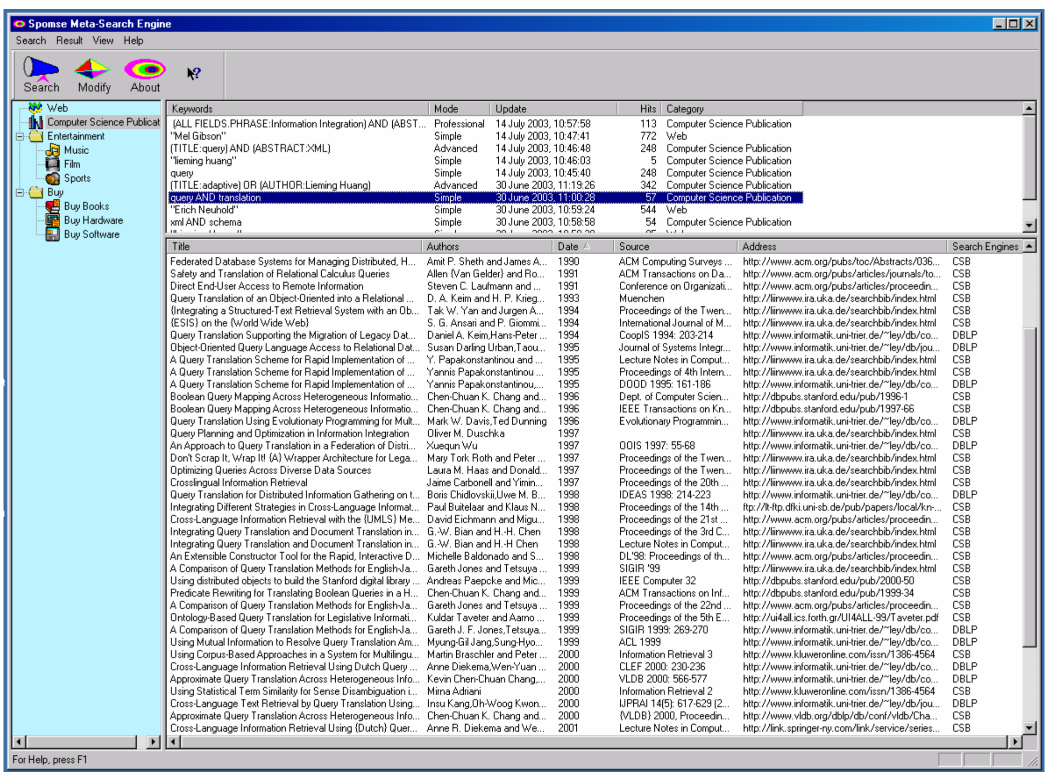


Figure 6.12 The main window of the SPOMSE meta-search engine

The "Categories" view contains a tree control that is organized as a "Category CSC" that has been introduced in Chapter 3. Users can also customize this tree according to their favorite categories.

This "Queries" view contains the list of queries. In SPOMSE meta-search engine, there are three modes of queries: *simple*, *advanced*, and *professional*. Figure 6.14 displays three examples of different modes of queries. The "List of queries" section displays the general information associated with each query. The elements for each query are displayed in these columns:

- **Keywords**: the keywords to search. More specifically, for a simple query, the displayed text string is just the input keywords or phrases; for an advanced or professional query, the displayed text string is a Boolean expression like *(Title:XML) and (Abstract: query) and (Date after 01.1999)*.
- **Mode**: the search mode, it can be simple, advanced, or professional.
- **Updated**: the date and time of the query when it was created or its last update.
- **Hits**: the number of matches obtained during the search
- **Category**: the selected category

It is possible to sort by column in ascending or descending order (click any column header), to move the columns (use drag and drop) and to adjust their width. A triangle appears on the sort column and the direction in which it is pointed indicates the ascending or descending order of the queries.

This "Results" view contains the list of results for the current selected or input query. The "List of results" section displays the document list for the selected query. Queries on different categories have different meta-information fields for the retrieved documents. For example, the retrieved documents on "Computer Science Publications" category have 6 meta-data fields: Title, Authors, Date, Source, Address, and Search engines (see Figure 6.12). The retrieved documents on "Web" category have 3 meta-data fields: Title, Address, and Search engines (see Figure 6.13). The main elements for each document are displayed in these four columns:

- **Title**: the title of the publication or Web page
- **Authors**: the author(s) of this publication

- **Date**: the date of this publication to be published
- **Source**: the source of this publication, such as the name of a journal issue (e.g., ACM Computing Surveys 22), the title of conference proceedings (e.g., ACM SIGIR 2002), or a publication repository of a university or organization.
- **Address**: the URL where this publication appeared
- **Search engines**: the search engines where the document was found.

It is also possible to sort by column in ascending or descending order (click any column header), to move the columns (use drag and drop) and to adjust their width. A triangle appears on the sort column and the direction in which it is pointed indicates the ascending or descending order of the results.

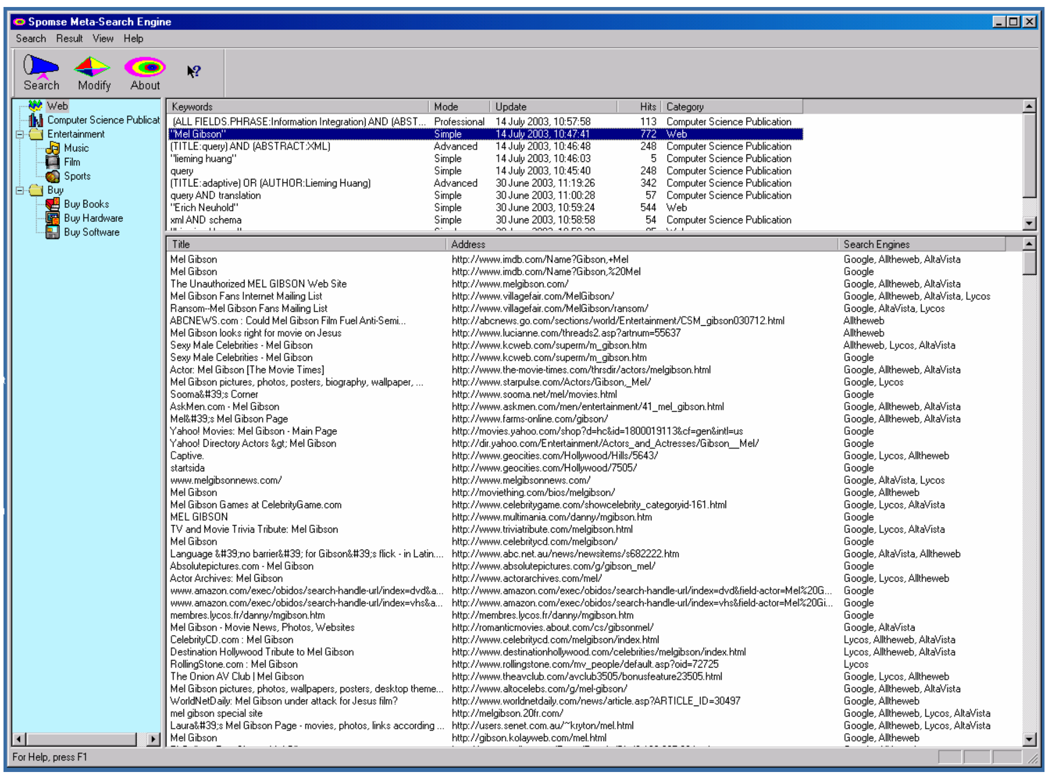


Figure 6.13 Result View of queries on Web Category

Figure 6.14 displays three kinds of query input user interfaces (*simple, advanced, and professional*) and a search status window showing the search progress of each target search engine (see Figure 6.14(d)).

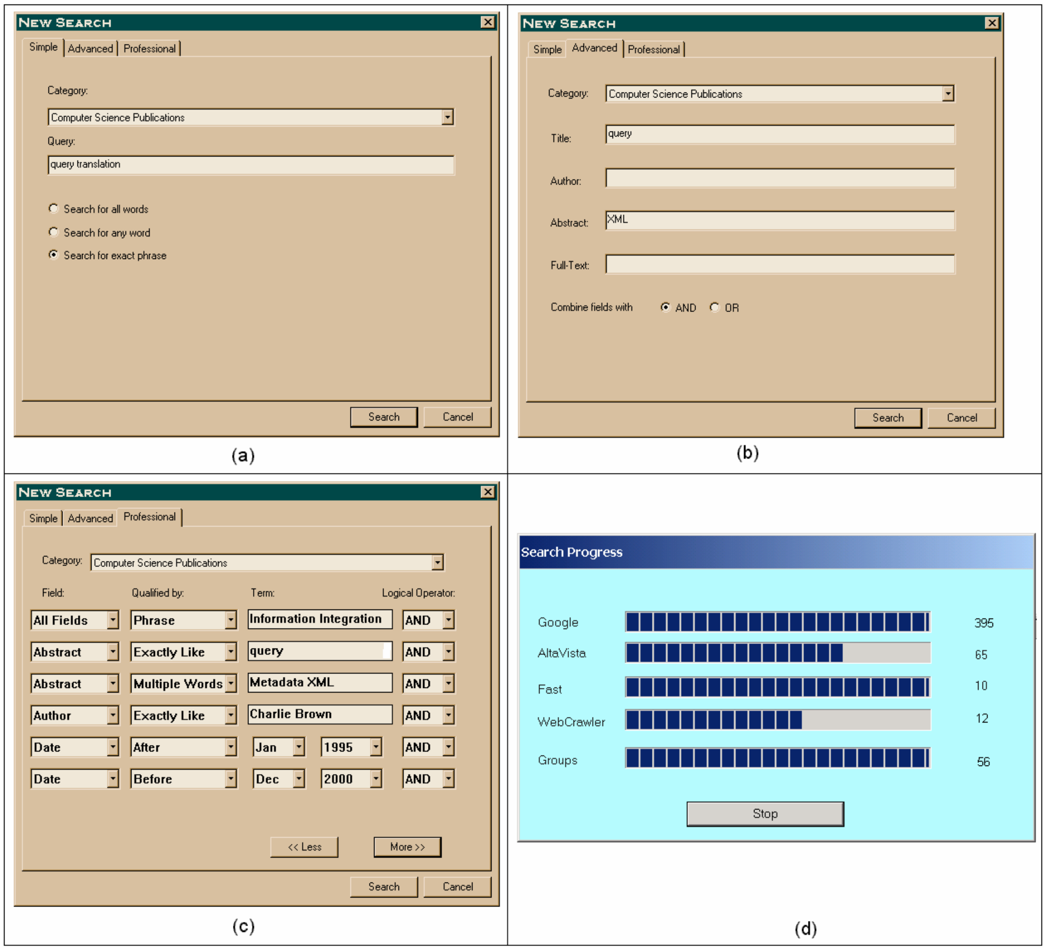


Figure 6.14 query input pages of SPOMSE

When users want to create a new query, they can use a simple query input page like Figure 6.14(a). There is a category list-box for domain selecting and an input-box for keywords inputting. Under this input-box, there are three radio-boxes. “Search for all words” means that results will include every keyword entered (and). “Search for any word” means that results will include at least one keyword (or). “Search for exact phrase” means that results will include the exact phrase entered (“…”).

Users can also use “Advanced” search interface to input queries that can limit terms to certain fields (see Figure 6.14(b)). These terms can be logically combined by either “AND” or “OR”.

Users can also use “Professional” search interface to dynamically construct any kind of complex queries (see Figure 6.14(c)). At the beginning of the query construction, there is only one row of controls. When users click the “More” button, there will be a new row of controls under the last row of controls. By clicking the “Less” button, the last row of controls will disappear. When users want to input a term concerning date information by choosing the “Date” field modifier, the input-box control (for term keyed in) will become two pull-down menus (one for month selection, another for year selection) and the corresponding qualifier control will only provide two items: “Before” and “After”.

Chapter 7

Evaluation

In order to test the effectiveness and the efficiency of the methods proposed in this dissertation, three groups of experiments have been performed. The first group contains three experiments comparing the efficiency of our prototype using three different types of user interfaces (i.e. simple, static mixed, and dynamically-generated). The second group contains two experiments comparing our meta-search engine prototype with four general-purpose meta-search engines (i.e., comparing the effectiveness of the Invisible Web with the Visible Web for the users' specific information needs). Finally, the third group of experiments checks the number of generated sub-queries and post-filters for eight complex query expressions in order to test our query translation algorithm and illustrate the great difference between sources. Before discussing these three experiments, first in section 7.1 we introduce the experimental settings.

7.1 Experimental settings

We now describe the selected data collections (Section 7.1.1), three test data sets (Section 7.1.2), target search engines and meta-search engines (Section 7.1.3), and metrics (Section 7.1.4) of our experimental evaluation.

7.1.1 Selected data collections

We have selected 96 international conference papers and journal articles (see Table 7.1) in the area of information retrieval, database, and digital library. These publications have been read carefully by the author of this dissertation during the PhD research. These conferences and journals include:

- ICDE – International Conference on Data Engineering;
- IR – Information Retrieval (Kluwer Academic Publishers);
- SIGMOD – ACM International Conference on Management of Data;
- CIKM – International Conference on Information and Knowledge Management;
- SIGIR – ACM International Conference on Research and Development in Information Retrieval;
- TODS – ACM Transactions on Database Systems;
- TOIS – ACM Transactions on Information Systems;
- VLDB – International Conference on Very Large Data Bases;
- WWW – International World Wide Web Conference;
- ACM DL – ACM conference on Digital Libraries (now JCDL).

Table 7.1 the 96 selected publications

1	Chen Li, Edward Chang: Query Planning with Limited Source Capabilities. ICDE 2000: 401-412
2	Luc Bouganim, Francise Fabret, C. Mohan, Patrick Valduriez: Dynamic Query Scheduling in Data Integration Systems. ICDE 2000: 425-434
3	Ugur Cetintemel, Michael J. Franklin, C. Lee Giles: Self-Adaptive User Profiles for Large-Scale Data Delivery. ICDE 2000: 622-633
4	Hector Garcia-Molina, Wilburt Labio, Ramana Yerneni: Capability-Sensitive Query Processing on Internet Sources. ICDE 1999: 50-59
5	Yannis Papakonstantinou, Pavel Velikhov: Enhancing Semistructured Data Mediators with Document Type Definitions. ICDE 1999: 136-145
6	Weiyi Meng, King-Lup Liu, Clement T. Yu, Wensheng Wu, Naphtali Rishe: Estimating the Usefulness of Search Engines. ICDE 1999: 146-153
7	Ling Liu: Query Routing in Large-Scale Digital Library Systems. ICDE 1999: 154-163
8	Marian H. Nodine, William Bohrer, Anne H. H. Ngu: Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth. ICDE 1999:358-365
9	Berthold Reinwald, H. Pirahesh, G. Krishnamoorthy, G. Lapis, B. T. Tran, S. Vora: Heterogeneous Query Processing through SQL Table Functions. ICDE 1999: 366-373
10	Jacques Calmet, Sebastian Jekutsch, Joachim Schü A Generic Query-Translation Framework for a Mediator Architecture. ICDE 1997: 434-443
11	Xiaolei Qian: Query Folding. ICDE 1996: 48-55
12	Sudarshan S. Chawathe, Hector Garcia-Molina, Jennifer Widom: A Toolkit for Constraint Management in Heterogeneous Information Systems. ICDE 1996: 56-65
13	Yannis Papakonstantinou, Hector Garcia-Molina, Jeffrey D. Ullman: MedMaker: A Mediation System Based on Declarative Specifications. ICDE 1996: 132-141
14	Budi Yuwono, Dik Lun Lee: Search and Ranking Algorithms for Locating Resources on the World Wide Web. ICDE 1996: 164-171
15	Sibel Adali, Ross Emery: A Uniform Framework for Integrating Knowledge in Heterogeneous Knowledge Systems. ICDE 1995: 513-520
16	Christoph Baumgarten: Retrieving Information from a Distributed Heterogeneous Document Collection. Information Retrieval 3(3): 253-271 (2000)
17	Ian Ruthven: Incorporating Aspects of Information Use into Relevance Feedback. Information Retrieval 2(1): 79-84 (2000)
18	Eero Sormunen: A novel method for the evaluation of Boolean query effectiveness across a wide operational range. SIGIR 2000: 25-32
19	Allison L. Powell, James C. French, James P. Callan, Margaret Connell, Charles L. Viles: The impact of database selection on distributed searching. SIGIR 2000: 232-239
20	Brian D. Davison: Topical locality in the Web. SIGIR 2000: 272-279
21	Peter Bruza, Robert McArthur, Simon Dennis: Interactive Internet search: keyword, directory and query reformulation mechanisms compared. SIGIR 2000: 280-287
22	Xiaolan Zhu, Susan Gauch: Incorporating quality metrics in centralized/distributed information retrieval on the World Wide Web. SIGIR 2000: 288-295
23	J. C. French, A. L. Powell, J. P. Callan, C. L. Viles, T. Emmitt, K. J. Prey, Y. Mou: Comparing the Performance of Database Selection Algorithms. SIGIR 1999: 238-245
24	Christoph Baumgarten: A Probabilitstic Solution to the Selection and Fusion Problem in Distributed Information Retrieval. SIGIR 1999: 246-253
25	Krishna Bharat, Monika Rauch Henzinger: Improved Algorithms for Topic Distillation in a Hyperlinked Environment. SIGIR 1998: 104-111
26	Jinxi Xu, James P. Callan: Effective Retrieval with Distributed Collections. SIGIR 1998: 112-120
27	James C. French, Allison L. Powell, C. L. Viles, T. Emmitt, K. J. Prey: Evaluating Database Selection Techniques: A Testbed and Experiment. SIGIR 1998: 121-129
28	Bienvenido Velez, Ron Weiss, Mark A. Sheldon, David K. Gifford: Fast and Effective Query Refinement. SIGIR 1997: 6-15
29	M. Hearst, C. Karadi: Cat-a-Cone: An Interactive Interface for Specifying Searches and Viewing Retrieval Results using a Large Category Hierarchy. SIGIR 1997:246-255
30	Mark Magennis, C. J. van Rijsbergen: The Potential and Actual Effectiveness of Interactive Query Expansion. SIGIR 1997: 324-332
31	Jinxi Xu, W. Bruce Croft: Query Expansion Using Local and Global Document Analysis. SIGIR 1996: 4-11
32	Brendon Cahoon, Kathryn S. McKinley: Performance Evaluation of a Distributed Architecture for Information Retrieval. SIGIR 1996: 110-118
33	Anil S. Chakravarthy, Kenneth B. Haase: Netserf: Using Semantic Knowledge to Find Internet Information Archives. SIGIR 1995: 4-11
34	James P. Callan, Zhihong Lu, W. Bruce Croft: Searching Distributed Collections with Inference Networks. SIGIR 1995: 21-28
35	Mirja Iivonen: Searchers and Searchers: Differences between the Most and Least Consistent Searchers. SIGIR 1995: 149-157
36	Ellen M. Voorhees, Narendra Kumar Gupta, Ben Johnson-Laird: Learning Collection Fusion Strategies. SIGIR 1995: 172-179
37	P. Ingwersen: Polyrepresentation of Information Needs and Semantic Entities: Elements of a Cognitive Theory for Information Retrieval Interaction. SIGIR 1994: 101-110
38	N. J. Belkin, C. Cool, W. B. Croft, James P. Callan: Effect of Multiple Query Representations on Information Retrieval System Performance. SIGIR 1993: 339-346
39	Vassilis Christophides, Sophie Cluet, Jerome Simeon: On Wrapping Query Languages and Efficient XML Integration. SIGMOD Conference 2000: 141-152
40	M. Rodriguez-Martinez, N. Roussopoulos: MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources.

	SIGMOD Conference 2000: 213-224
41	Roy Goldman, Jennifer Widom: WSQ/DSQ: A Practical Approach for Combined Querying of Databases and the Web. SIGMOD Conference 2000: 285-296
42	Zachary G. Ives, D. Florescu, M. Friedman, Alon Y. Levy, D. S. Weld: An Adaptive Query Execution System for Data Integration. SIGMOD Conference 1999: 299-310
43	Daniela Florescu, Alon Y. Levy, Ioana Manolescu, Dan Suciu: Query Optimization in the Presence of Limited Access Patterns. SIGMOD Conference 1999: 311-322
44	Ramana Yerneni, Chen Li, Hector Garcia-Molina, Jeffrey D. Ullman: Computing Capabilities of Mediators. SIGMOD Conference 1999: 443-454
45	Sophie Cluet, Claude Delobel, Jerome Simeon, Katarzyna Smaga: Your Mediators Need Data Conversion! SIGMOD Conference 1998: 177-188
46	William W. Cohen: Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. SIGMOD Conference 1998: 201-212
47	Roberto J. Bayardo Jr., Bill Bohrer, et. el.: InfoSleuth: Semantic Integration of Information in Open and Dynamic Environments. SIGMOD Conference 1997: 195-206
48	Luis Gravano, K. Chang, H. Garcia-Molina, A. Paepcke: STARTS: Stanford Proposal for Internet Meta-Searching. SIGMOD Conference 1997: 207-218
49	Sibel Adali, K. S. Candan, Y. Papakonstantinou, V. S. Subrahmanian: Query Caching and Optimization in Distributed Mediator Systems. SIGMOD Conf. 1996: 137-148
50	Jason Chaffee, Susan Gauch: Personal Ontologies for Web Navigation. CIKM 2000: 227-234
51	A. Kruger, C. Lee Giles, F. Coetzee, E. J. Glover, G. W. Flake, S. Lawrence, C. W. Omlin: DEADLINER: Building a New Niche Search Engine. CIKM 2000: 272-281
52	Leah S. Larkey, M. Connell, James P. Callan: Collection Selection and Results Merging with Topically Organized U.S. Patents and TREC Data. CIKM 2000: 282-289
53	King-Lup Liu, Weiyi Meng, Clement T. Yu, Naphtali Rishe: Discovery of Similarity Computations of Search Engines. CIKM 2000: 290-297
54	Ruxandra Domenig, Klaus R. Dittrich: A Query based Approach for Integrating Heterogeneous Data Sources. CIKM 2000: 453-460
55	Yong Lin, Jian Xu, Ee-Peng Lim, Wee Keong Ng: ZBroker: A Query Routing Broker for Z39.50 Databases. CIKM 1999: 202-209
56	Eric J. Glover, Steve Lawrence, William P. Birmingham, C. Lee Giles: Architecture of a Metasearch Engine That Supports User Information Needs. CIKM 1999: 210-216
57	Clement T. Yu, Weiyi Meng, King-Lup Liu, W. Wu, N. Rishe: Efficient and Effective Metasearch for a Large Number of Text Databases. CIKM 1999: 217-224
58	M. L. Barja, T. Bratvold, J. Myllymaki, Gabriele Sonnenberger: Informia: A Mediator for Integrated Access to Heterogeneous Information Sources. CIKM 1998: 234-241
59	I-Min A. Chen, Doron Rotem: Integrating Information from Multiple Independently Developed Data Sources. CIKM 1998: 242-250
60	Louiqa Raschid, Yahui Chang, Bonnie J. Dorr: Interoperable Query Processing with Multiple Heterogeneous Knowledge Servers. CIKM 1993: 461-470
61	K. Chang, H. Garcia-Molina, A. Paepcke: Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System. TOIS 17(1): 1-39 (1999)
62	David Hawking, Paul B. Thistlewaite: Methods for Information Server Selection. TOIS 17(1): 40-76 (1999)
63	Norbert Fuhr: A Decision-Theoretic Approach to Database Selection in Networked IR. TOIS 17(3): 229-249 (1999)
64	C. Goh, S. Bressan, S. Madnick, M. Siegel: Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. TOIS 17(3):270-293(1999)
65	Ee-Peng Lim, Ying Lu: Harp: A Distributed Query System for Legacy Public Libraries and Structured Databases. TOIS 17(3): 294-319 (1999)
66	Daniel Dreilinger, Adele E. Howe: Experiences with Selecting Search Engines Using Metasearch. TOIS 15(3): 195-222 (1997)
67	Anthony Tomasic, Luis Gravano, Calvin Lue, Peter M. Schwarz, Laura M. Haas: Data Structures for Efficient Broker Implementation. TOIS 15(3): 223-253 (1997)
68	Ulla Merz, Roger King: Direct: A Query Facility for Multiple Databases. TOIS 12(4): 339-359 (1994)
69	Luis Gravano, Hector Garcia-Molina, Anthony Tomasic: GlOSS: Text-Source Discovery over the Internet. TODS 24(2): 229-264 (1999)
70	Maurizio Panti, Luca Spalazzi, Alberto Giretti: A Case-Based Approach to Information Integration. VLDB 2000: 557-565
71	Felix Naumann, Ulf Leser, Johann Christoph Freytag: Quality-driven Integration of Heterogenous Information Systems. VLDB 1999: 447-458
72	Weiyi Meng, King-Lup Liu, Clement T. Yu, Xiaodong Wang, Yuhsi Chang, Naphtali Rishe: Determining Text Databases to Search in the Internet. VLDB 1998: 14-25
73	Tova Milo, Sagit Zohar: Using Schema Matching to Simplify Heterogeneous Data Translation. VLDB 1998: 122-133
74	Luis Gravano, Hector Garcia-Molina: Merging Ranks from Heterogeneous Internet Sources. VLDB 1997: 196-205
75	Daniela Florescu, Daphne Koller, Alon Y. Levy: Using Probabilistic Information in Data Integration. VLDB 1997: 216-225
76	Vasilis Vassalos, Yannis Papakonstantinou: Describing and Using Query Capabilities of Heterogeneous Sources. VLDB 1997: 256-265
77	Mary Tork Roth, Peter M. Schwarz: Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources. VLDB 1997: 266-275
78	Laura M. Haas, Donald Kossmann, Edward L. Wimmers, Jun Yang: Optimizing Queries Across Diverse Data Sources. VLDB 1997: 276-285
79	Alon Y. Levy, Anand Rajaraman, Joann J. Ordille: Querying Heterogeneous Information Sources Using Source Descriptions. VLDB 1996: 251-262
80	Luis Gravano, Hector Garcia-Molina: Generalizing GlOSS to Vector-Space Databases and Broker Hierarchies. VLDB 1995: 78-89
81	Wen-Syan Li, Chris Clifton: Semantic Integration in Heterogeneous Databases Using Neural Networks. VLDB 1994: 1-12
82	Renee J. Miller, Yannis E. Ioannidis, Raghu Ramakrishnan: The Use of Information Capacity in Schema Integration and Translation. VLDB 1993: 120-133
83	Daniela Florescu, Donald Kossmann, Ioana Manolescu: Integrating keyword search into XML query processing. WWW9 / Computer Networks 33(1-6): 119-135 (2000)
84	Atsushi Sugiura, Oren Etzioni: Query routing for Web search engines: architecture and experiments. WWW9 / Computer Networks 33(1-6): 417-429 (2000)
85	Oren Zamir, Oren Etzioni: Grouper: A Dynamic Clustering Interface to Web Search Results. WWW8 / Computer Networks 31(11-16): 1361-1374 (1999)
86	Steve Lawrence, C. Lee Giles: Inquirus, the NECI Meta Search Engine. WWW7 / Computer Networks 30(1-7): 95-105 (1998)
87	Sergey Brin, Lawrence Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine. WWW7 / Computer Networks 30(1-7): 107-117 (1998)
88	K. Bharat, A. Broder. A technique for Measuring the Relative Size and Overlap of Public Web Search engines. WWW7 / Computer Networks 30(1-7): 379-388 (1998)
89	Massimo Marchiori: The Quest for Correct Information on the Web: Hyper Search Engines. WWW6 / Computer Networks 29(8-13): 1225-1236 (1997)
90	Nick Craswell, Peter Bailey, David Hawking: Server selection on the World Wide Web. ACM DL 2000: 37-4
91	Sergey Melnik, Hector Garcia-Molina, Andreas Paepcke: A mediation infrastructure for digital library services. ACM DL 2000: 123-1326

92	Naomi Dushay, James C. French, Carl Lagoze: Using Query Mediators for Distributed Searching in Federated Digital Libraries. ACM DL 1999: 171-178
93	Soyeon Park: User Preferences When Searching Individual and Integrated Full-Text Databases. ACM DL 1999: 195-203
94	M. Wang Baldonado, Chen-Chuan K. Chang, Luis Gravano, Andreas Paepcke: Metadata for Digital Libraries: Architecture and Design Rationale. ACM DL 1997: 47-56
95	Ellen M. Voorhees, Richard M. Tong: Multiple Search Engines in Database Merging. ACM DL 1997: 93-102
96	Chen-Chuan K. Chang, Hector Garcia-Molina: Evaluating the Cost of Boolean Query Mapping. ACM DL 1997: 103-112

In section 7.1.4, we will discuss that these publications are also used to evaluate the relevance of the results.

From these 96 publications we have selected 30 testing terms. These 30 testing terms contain 15 single keywords (Table 7.2), 12 phrases (Table 7.3), and 3 authors' names (Table 7.4).

Table 7.2 Fifteen selected single keywords

Broker	Categorization	Cluster	Collection	Effectiveness
Intermediary	Mediator	Metadata	Optimization	Precision
RDF	Relevance	Repositories	Wrapper	XML

Table 7.3 Twelve selected phrases

Data fusion	Source selection	Digital library
Distributed search	Heterogeneous information sources	Information integration
Information need	Query formulation	Query translation
Relevance feedback	Resource discovery	Wrapper generation

Table 7.4 Three selected authors' names

Alon Y. Levy	James P. Callan	Luis Gravano

7.1.2 Three test sets

From the above 30 testing terms, three test sets (for single keywords, phrases, and query expressions, respectively) have been constructed:

The first test set (**TS1**) contains 15 original single keywords listed in Table 7.2. When we use these keywords to test the relevance of results from general-purpose meta-search engines (for the Visible Web), we find out that most results are irrelevant because the relevance measure is on certain publications.

The second test set (**TS2**) contains 12 phrases listed in Table 7.3.

The third test set (**TS3**) contains 8 query expressions (see Table 7.5) that are constructed by the original query terms with the modifiers and logical operators according to the concrete publications we have selected. These 8 query expressions (**QE1- QE8**) are respectively constructed according to publications 61, 39, 84, 79, 92, 19, 48, and 10 in Table 7.1. In this test set, we use six field modifiers (i.e. <Title>, <Keywords>, <Abstract>, <Full-Text>, <Author>, and <Date>), six term qualifiers (i.e. <Exactly Like>, <Contain>, <Multiple words>, <Stemming word>, <Before>, and <After>), and three logical operators (i.e. <AND>, <OR>, <NOT>).

Table 7.5 the third test setTS3 (8 query expressions QE1-QE8)

1.	“Mediator”, “query” and “optimization”(or “optimizing”) in title, abstract, or keywords.
2.	Title contains (“wrapper” or “wrapping”, or other stemming words) AND (“XML” and “Integration”) in title, abstract, keywords, or full-text.
3.	“Cluster” (or “clustering”) and “relevance”, “query routing” (or “source selection”) in the paper, not in SIGIR and CIKM conferences.
4.	((Author is “Alon Y. Levy”) AND (((“Information Integration” in All fields) AND (Title contains “query”)) OR ((“Metadata”, “XML”) in Abstract))) in the Computer Science category, published during the period of 1995 to 1999.
5.	“Mediator” in title and (“resource discovery” and “distributed”) in abstract or keywords, in the area of “Digital Library”.
6.	Author MAY include “James P. Callan” AND (Title OR Abstract) contains (“Database selection” or “Source Selection”) and “Distributed search” AND published date is after 1999.
7.	Author is “Luis Gravano” AND the paper MAY include (“Digital Library”, Metadata, “collection or repository”).
8.	Title contains “query translation” AND “mediator” in Title or Abstract, published during the period of 1996 to 1998.

TS1 and TS2 will be tested in the second group of experiments. TS3 will be tested in the first and the third groups of experiments.

7.1.3 Target search engines and meta-search engines

Four general-purpose Web meta-search engines (i.e., Mamma, SavvySearch, Profusion, and MetaCrawler) have been selected to compare with our special-purpose meta-search engine. These general-purpose meta-search engines integrate some general purpose search engines, such as Google, AltaVista, Infoseek, and so forth. These meta-search engines will be tested in the second group of experiments.

We test our meta-search engine by integrating three computer science publication oriented search engines: ACM-DL, CORA, and NCSTRL. ACM-DL is a digital library offering almost all ACM journals and proceedings. CORA is a specialized Web search engine with crawlers scouring cyberspace with the goal of indexing only a small subset of Web documents relevant to the computer society. NCSTRL is an international collection of computer science research reports and papers from a number of participating institutions and archives. These three sources are quite different. ACM-DL is a local repository. CORA is an index base collecting all relevant documents on the Visible Web. NCSTRL is like a distributed, federated database. The contents of these three sources are quite different but with some overlapping. There are also many discrepancies between the user interfaces and the query capabilities of these sources (see Figure 1.2: <NCSTRL>, Figure 1.3 : <ACM-DL>, and Figure 4.20 : <CORA>).

7.1.4 Evaluation metrics

For the evaluation of IR systems, the trade-off between recall and precision is well-known. In the Web searching, users can judge the precision of an IR system by inspecting the retrieved documents. However, they cannot judge recall, which would involve inspecting the whole collection, and that in the case of Web is impossible. Just as the paper [BMD00] points out: “the document collection is the WWW where recall is impossible to measure. Moreover, most searches on the WWW are not concerned with finding all of the relevant material”.

In addition, the meta-search engine does not have local indexing data and it just translates the users' original queries into the formats understood by remote search engines. So the recall completely depends on the indexing schemes and retrieving mechanisms of the target sources.

However, in our first group of experiments, recall can be easily measured because the three target sources can be examined and our 96 selected publications are the criteria for checking the precision and recall of all retrieved results. For example, when testing the sixth query expression QE6 on the ACM-DL, we can examine the whole journal and proceedings collections. Of course, most irrelevant categories can be ignored, i.e., "Computer Graphics", "Computer Security" and "Lisp and Functional Programming" have nothing to do with "distributed search" and "source selection".

Because we only search on three sources, some publications cannot be found from them. For example, the third query expression QE3 is constructed according to a paper (#84) from the 9th International World Wide Web Conference (WWW-2000). It cannot be found in these three sources (see Tables 7.7-7.9). When counting the relevant hits, we do not consider duplicate hits from different search engines.

In the first group of experiments, we test the eight query expressions to compare the retrieval efficiencies on three different kinds of user interfaces (i.e., simple, static, and adaptive).

In the second group of experiments we test the selected single keywords and phrases on the Visible Web (the fourth experiment uses 4 general-purpose meta-search engines that search on the WWW) and the Invisible Web (the fifth experiment uses our SPOMSE meta-search engine prototype that searches on the NCSTRL, ACM-DL, and CORA). These two experiments search on completely different kinds of document collections. The purpose of why we do so is to illustrate the difference of retrieval precision between the Visible Web and the Invisible Web.

We judge that an entry (or a hit) from a search engine is qualified if it is (or it is relevant to) one of the 96 publications in the selected conferences and journals. The author of this thesis have tested all the selected data and judged the relevance of the retrieved results. The author is qualified to this evaluating work because the author has studied all the publications. When testing the selected general-purpose meta-search engines, the retrieval precision is very low. Therefore, we loosen the criteria of their relevance judging. We regard a return hit (such as homepages of research institutes, researchers and project description, and so on.) as relevant if it contains information relevant to the input keyword or phrase. For example, when we input the phrase "Data fusion", the Mamma meta-search engine returns this page: http://www.cs.iastate.edu/~honavar/ailab/projects/fusion.html. Even though in this page there is no expected publication information, we still count it one match. If we strictly measure the retrieval precision of general-purpose search engine by our metrics, the actual precision will be far less than the results in the tables of this thesis.

In our prototype, the purpose of combining the adaptive UI construction and the constraint-based query translation is to enable users to easily input their information needs (queries) and to enable target sources to accurately understand users' queries, thus significantly reducing the number of irrelevant results.

7.2 Efficiencies under different user interfaces

What kind of user interface will improve the efficiency of special-purpose Web meta-searching? Simple? Static? Or Adaptive, dynamically-generated? In this section, we use three experiments to answer this question. First in section 7.2.1, we introduce the experimental setup. Then in section 7.2.2, we list all experimental results and discuss them. Finally, in section 7.2.3, we compare the pros and cons of these three kinds of user interfaces.

7.2.1 Experimental setup

In this group of experiments, we compare the retrieval precision of our meta-search engine prototype with three different kinds of user interfaces. This prototype integrates 3 scientific publication-oriented search engines (i.e., ACM-DL, CORA, and NCSTRL). Here we only test the 8 query expressions listed in Table 7.5 (**TS3**). It is the same to query single keywords (or phrases) through these three kinds of user interfaces; even the simple user interface can support all single keywords and phrases. There are three experiments in this group of experiments, i.e., **Experiment1**, **Experiment2**, and **Experiment3**.

Just as paper [BDMS94] points out: "It is safe to say that at least 99% of the available data is of no interest to at least 99% of the users", the research on pinpointing what users want (or "searching for a needle in a world of haystacks") attracts more and more attention of scientists and common users. In the following three experiments, we want to pinpoint certain publications by using the eight complex queries. Now we introduce the three experiments one by one:

Figure 7.1 User interface of the first experiment (Experiment1)

The first experiment (**Experiment1**) adopts a simple user interface that contains only a simple input box without constraining controls (see Figure 7.1). When inputting a query expression, we discard all field modifiers, term qualifiers and logical operators. In Table 7.6, we list all simplified queries from TS3 in Table 7.5. Therefore, any search engine can support such simple queries and there is no need to translate queries from the meta-search engine to the target search engines.

Table 7.6 Simplified queries of TS3

Mediator query optimization
Wrapper XML Integration
Cluster relevance "query routing"
"Alon Y. Levy" "Information Integration" query Metadata XML
Mediator "resource discovery" distributed "Digital Library"
"James P. Callan" "Database selection" "Distributed search"
"Luis Gravano" "Digital Library" Metadata collection
"query translation" mediator

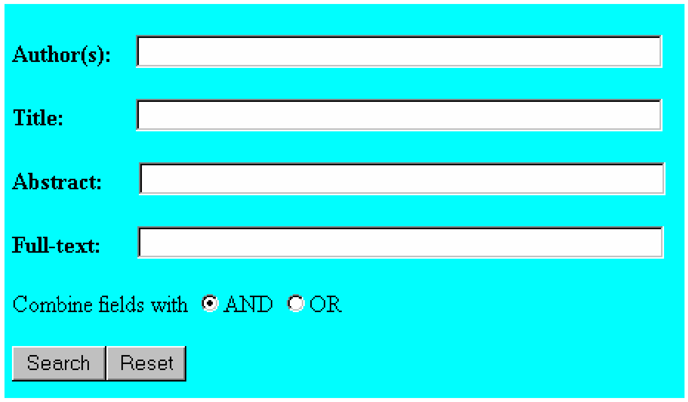


Figure 7.2 User interface of the second experiment (Experiment2)

The second experiment (**Experiment2**) employs a static, partially-mixed HTML user interface (see Figure 7.2) containing major controls. When translating a query from the meta-search engine to a search engine, the system does not post-process the results (e.g., tightening and composing results, See Chapter 4). Because of the limitations of the static user interfaces, an original query cannot be perfectly supported by them. Sometimes, some queries have to be manually decomposed by users.

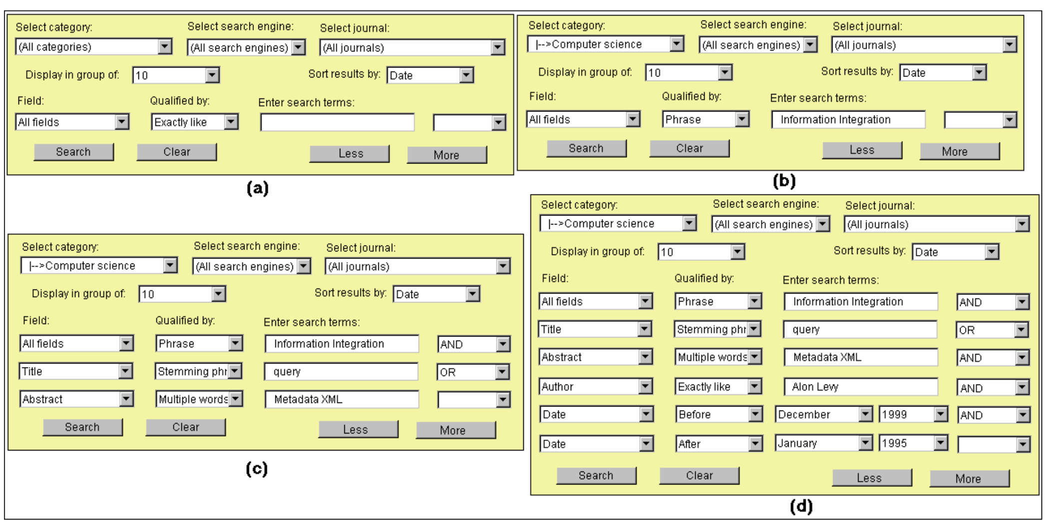


Figure 7.3 User interface of the third experiment (Experiment3)

The third experiment (**Experiment3**) adopts a progressive, dynamically-generated user interface (see Figure 7.3) and uses the constraint-based query translation method proposed in chapter 4. In the user interface of this experiment, almost all conflicts are automatically resolved by using our approach. Figure 7.3 shows four snapshots of user interfaces in which the example query has been input progressively. Figure 7.3(a) is the initial interface and in Figure 7.3(d), the query is completely input. When users press the "More" button, there will be a new row of controls under the last row of controls; and when the "Less" button is pressed, the last row of controls will disappear. Because the flexibility and extensibility of this kind of query interfaces, all the tested query expressions can be supported well.

7.2.2 Experimental results

In this section, we list and discuss the results of the three experiments. First in Tables 7.7-7.9, we list the results of each target search engine (i.e., ACM-DL, CORA, and NCSTRL). Table 7.10 lists the overall results in our meta-search engine prototype, in which all duplicates are removed. We use "QE1"-"QE8" to mean the eight query expressions in Table 7.5. For each query in these tables, the number in the left cell means the number of returned hits and the number in the right cell means the number of relevant hits.

Table 7.7 lists the experimental results of using TS3 on ACM-DL. We can see that the third query expression QE3 and the eighth query expression QE8 retrieve no relevant results because the expected publications (#84, #10) are from WWW-2000 and ICDE-1997 conferences, so ACM-DL has no such publications. In the last row of this table, we can see that the precision of using simple user interface (Experiment1) is 2.29% (6/262). The precision of using static user interface (Experiment2) is 16.22% (6/37). The precision of using dynamically generated user interface (Experiment3) is 75% (6/8).

Table 7.7 The experimental results of TS3 on ACM-DL

	Experiment1		Experiment2		Experiment3	
QE 1	56	1	1	1	1	1
QE 2	29	1	2	1	1	1
QE 3	81	0	3	0	0	0
QE 4	2	1	1	1	1	1
QE 5	19	1	1	1	1	1
QE 6	21	1	15	1	2	1
QE 7	6	1	2	1	1	1
QE 8	48	0	12	0	1	0
∑	262	6	37	6	8	6

Table 7.8 lists the experimental results of using TS3 on Cora search engine. In this table, only three query expressions (QE1, QE6, and QE8) get relevant results. This means that the authors of these publications put their papers on the websites that can be publicly accessed and crawled by the search engine spiders. In the last row of this table, we can see that the precision of using simple user interface (Experiment1) is 6.12% (3/49). The precision of using static user interface (Experiment2) is 15.79% (3/19). The precision of using dynamically generated user interface (Experiment3) is 60% (3/5).

Table 7.8 The experimental results of TS3 on Cora

	Experiment1		Experiment2		Experiment3	
QE 1	6	1	3	1	2	1
QE 2	3	0	0	0	0	0
QE 3	8	0	2	0	0	0
QE 4	3	0	1	0	0	0
QE 5	5	0	0	0	0	0
QE 6	17	1	12	1	2	1
QE 7	1	0	0	0	0	0
QE 8	6	1	1	1	1	1
∑	49	3	19	3	5	3

Table 7.9 lists the experimental results of using TS3 on NCSTRL. Only the fifth query expression (QE5) gets relevant result. This means that the institute where the authors work at is one of the participating institutions of NCSTRL and they put their publications on one of the servers of NCSTRL. In the last row of this table, we can see that the precision of using simple user interface (Experiment1) is 8.33% (1/12). The precision of using static user interface (Experiment2) is 14.29% (1/7). The precision of using dynamically generated user interface (Experiment3) is 25% (1/4).

Table 7.9 The experimental results of TS3 on NCSTRL

	Experiment1		Experiment2		Experiment3	
QE 1	4	0	4	0	3	0
QE 2	0	0	0	0	0	0
QE 3	0	0	0	0	0	0
QE 4	0	0	0	0	0	0
QE 5	1	1	1	1	1	1
QE 6	7	0	2	0	0	0
QE 7	0	0	0	0	0	0
QE 8	0	0	0	0	0	0
∑	12	1	7	1	4	1

In the above three tables, we list the experimental results for each target search engine. In Table 7.10, we list the overall results of our meta-search engine prototype, where duplicates are not counted. In the last row of this table, we can see that the precision of using simple user interface (Experiment1) is 2.31% (7/303). The precision of using static user interface (Experiment2) is 12.28% (7/57). The precision of using dynamically generated user interface (Experiment3) is 58.33% (7/12).

Table 7.10 The experimental results of TS3 on SPOMSE

	Experiment1		Experiment2		Experiment3	
QE 1	58	1	5	1	3	1
QE 2	31	1	2	1	1	1
QE 3	88	0	4	0	0	0
QE 4	5	1	2	1	1	1
QE 5	22	1	1	1	1	1
QE 6	40	1	28	1	3	1
QE 7	8	1	2	1	1	1
QE 8	51	1	13	1	2	1
∑	303	7	57	7	12	7

Table 7.11 Precision comparison

	Experiment 1	Experiment 2	Experiment 3
Mean returned hits per query	37.875	7.125	1.5
Mean relevant hits per query	0.875	0.875	0.875
Precision (%)	2.31	12.28	58.33

Table 7.11 compares retrieval precision of the three experiments. The first experiment does not use all kinds of constraint information, so it retrieves a lot of irrelevant information. The second experiment can use constraint controls, so its retrieval precision (12.28%) is higher than the first experiment (2.31%). From this comparison, we know that using field modifiers and term qualifiers can significantly impact the precision of information retrieval, especially for specific and exact information needs (this is particularly important when considering the tremendous amount of information on the Internet). For example, when searching the keyword “query” on Cora search engine, using the <Title> field modifier gets 268 hits while without any modifiers gets 1588 hits. However, this kind of static query interface is not flexible enough for users to input queries and the results have not been post-processed. In the third experiment, because almost all conflicts between different sources or between

the controls of the same source have been sufficiently coordinated and the constraint-based query translation method has been employed, this experiment achieves higher precision (58.33%) than the other two experiments.

7.2.3 Comparison of different user interfaces

Table 7.12 compares the pros and cons of the three kinds of information integration systems with differing user interfaces.

From the above experiments, we know that a meta-search engine with an adaptive, dynamically generated user interface, coordinating the constraints among the controls of heterogeneous search engines, will greatly improve the effectiveness of information retrieval on the Internet.

Table 7.12 Comparison of the three kinds of user interfaces for Web meta-searching

Simple or "Greatest-Common-Divisor" user interface (Figure 7.1)	
Pros	(1) It can be supported by all integrated sources; (2) It is simple for users to input information needs and for the system to translate queries;
Cons	(1) It will inevitably discard the rich functionality provided by specific information sources; (2) It is difficult for users to input complicated queries and retrieve more specific information; (3) Retrieval precision is very low;
Static, partially-mixed user interface (Figure 7.2)	
Pros	(1) Users can express their information needs more accurately than in simple or "GCD" interface;
Cons	(1) The constraints between the user interfaces of heterogeneous sources may cause a user query to be inconsistent with a source and make the query translation difficult; (2) The static user interface lacks flexibility and makes the interaction between users and system difficult.
Adaptive, dynamically-generated user interface (Figure 7.3)	
Pros	(1) It has the advantages and avoids the disadvantages of both "GCD" and Static, partially-mixed user interfaces; (2) It will support the progressively self-refining construction of users' information needs; (3) Conflicts among heterogeneous sources can be coordinated efficiently; (4) User queries will match the queries supported by target sources as much as possible;
Cons	Its implementation needs more efforts than the other two.

7.3 The Invisible Web *vs*. the Visible Web

In this section, two experiments have been carried out to compare the retrieval effectiveness of using general-purpose meta-search engines and using special-purpose meta-search engines. General-purpose meta-search engines integrate general-purpose Web search engines (e.g., Google, AltaVista, Fast-Search, Lycos) that search the Visible Web. A special-purpose meta-search engine integrates some Web-searchable databases and specialized search engines and its purpose is, therefore, for searching on the Invisible Web. We have introduced some general issues about the Invisible Web in chapter 1 and some examples of Invisible Web catalogues in chapter 2. In the following, we will first introduce the experimental setup in section 7.3.1. Then in section 7.3.2, the results of Experiment4 will be listed. After that, the results of Experiment5 will be listed in section 7.3.3. Finally, in section 7.3.4, we will analyze the results of these two experiments.

7.3.1 Experimental setup

This group of experiments contains two experiments:

The fourth experiment (**Experiment4**) tests 4 general-purpose Web meta-search engines (i.e., Mamma, SavvySearch, ProFusion, and MetaCrawler). In this experiment, we only test single keywords (**TS1**) and phrases (**TS2**). Current Web search engines and meta-search engines do not support query expressions with constraint information.

The fifth experiment (**Experiment5**) tests our meta-search engine prototype that integrates 3 scientific publication-oriented search engines the same as those in the first group of experiments (i.e., ACM-DL, Cora, and NCSTRL). In this experiment, we also test two test sets: **TS1** and **TS2**.

7.3.2 Experimental results of Experiment4

Table 7.13 lists the results of using TS1 on Experiment4. Each of the 15 single keywords in TS1 has been sent to 4 general-purpose meta-search engines.

Table 7.13 Results of using TS1 on Experiment4

	Mamma		Savvysearch		Profusion		MetaCrawler		∑	
Broker	75	0	43	0	26	0	57	0	201	0
Categorization	43	1	31	1	23	3	39	1	136	6
Cluster	58	0	50	0	28	0	46	0	182	0
Collection	86	0	44	0	22	0	54	0	206	0
Effectiveness	55	0	56	0	25	0	30	0	166	0
Intermediary	49	0	55	0	29	0	36	0	169	0
Mediator	48	0	55	0	26	0	38	0	167	0
Metadata	43	6	48	11	25	4	30	4	146	25
Optimization	67	1	54	1	25	0	45	1	191	3
Precision	57	0	53	0	25	0	46	0	181	0
RDF	41	5	36	4	24	3	26	4	127	16
Relevance	42	0	53	0	28	0	15	0	138	0
Repositories	45	1	46	0	25	1	34	0	150	2
Wrapper	48	1	54	0	19	2	38	0	159	3
XML	75	3	52	3	10	2	49	2	186	10
∑	832	18	730	20	360	15	583	12	2505	65

From Table 7.13, we can see that some words can get higher retrieval precision than others, e.g., Metadata (25/146 = 17.12%), RDF (16/127 = 12.60%), XML (10/186 = 5.38%). We can also get some relevant results using some other words (e.g., Categorization, Optimization, Repositories, and Wrapper). The precision of the rest words is zero. In the last row of this table, we can see that the precision of using the Mamma search engine is (18/832 = 2.16%). The precision of using Savvysearch is (20/730 = 2.74%). The precision of using Profusion is (15/360 = 4.17%). The precision of using MetaCrawler is (12/583 = 2.06%).

Table 7.14 lists the results of using TS2 on Experiment4. Each phrase in TS2 has been sent to 4 general-purpose meta-search engines.

Table 7.14 Results of using TS2 on Experiment4

	Mamma		Savvysearch		Profusion		MetaCrawler		∑	
Data fusion	6	0	26	0	36	1	20	0	88	1
Source selection	4	0	26	0	28	0	16	0	74	0
Digital library	17	0	42	1	88	3	29	2	176	6
Distributed search	0	0	22	2	66	2	15	2	103	6
Heterogeneous information sources	10	6	25	8	30	21	9	0	74	35
Information integration	20	0	40	2	75	17	25	1	160	20
Information need	11	0	21	0	61	2	4	0	97	2
Query formulation	10	3	25	8	27	16	9	4	71	31
Query translation	10	5	19	13	39	14	6	2	74	34
Relevance feedback	0	0	28	8	32	11	13	5	73	24
Resource discovery	11	1	23	4	58	4	19	6	111	15
Wrapper generation	10	6	19	15	26	12	12	6	67	39
∑	109	21	316	61	566	103	177	28	1168	213

From Table 7.14, we can see that some phrases can get higher retrieval precision than others, e.g., “Heterogeneous information sources” (35/74 = 47.30%), “Query formulation” (31/71 = 43.66%), “Query translation” (34/74 = 45.95%), “Relevance feedback” (24/73 = 32.88%), and “Wrapper generation” (39/67 = 58.21). We can also get some relevant results using some other phrases (e.g., “Digital library”, “Distributed search”, “Resource discovery”, etc.). In the last row of this table, we can see that the precision of using the Mamma search engine is (21/109 = 19.27%). The precision of using Savvysearch is (61/316 = 19.30%). The precision of using Profusion is (103/566 = 18.20%). The precision of using MetaCrawler is (28/177 = 15.82%).

7.3.3 Experimental results of Experiment5

In this experiment, we test our meta-search engine prototype by using TS1 and TS2. Table 7.15 lists the results of using TS1 on Experiment5.

Table 7.15 Results of using TS1 on Experiment5:

	SPOMSE	
Broker	181	36
Categorization	476	109
Cluster	1406	117

Collection	3041	290
Effectiveness	1839	139
Intermediary	70	17
Mediator	161	56
Metadata	225	58
Optimization	6407	159
Precision	1367	187
RDF	16	9
Relevance	1105	248
Repositories	286	27
Wrapper	148	36
XML	188	63
∑	16916	1551

From Table 7.15, we can see that some words can get higher retrieval precision than others, e.g., RDF (9/16 = 56.25%), Mediator (56/161 = 34.78%), XML (63/188 = 33.51%), and Metadata (58/225 = 25.78%). In Experiment4, from Table 7.13, we can also see that these words (such as RDF, XML, and Metadata) get higher retrieval precision than other words. The reason is that these words are quite new and they have not been commonly used in other domains.

Table 7.16 lists the results of using TS2 on Experiment5.

Table 7.16 All results of using TS2 on Experiment5:

	SPOMSE	
Data fusion	40	7
Source selection	6	4
Digital library	336	92
Distributed search	48	17
Heterogeneous information sources	44	37
Information integration	49	31
Information need	111	48
Query formulation	381	65
Query translation	44	14
Relevance feedback	231	95
Resource discovery	89	38
Wrapper generation	11	6
∑	1390	454

7.3.4 Analysis of the results of Experiment4 and Experiment5

In Experiment4, although we judge the relevance very loosely, the number of relevant results is still very few. The reason is that the tested keywords have different meanings in different domains and contexts. Only from a single keyword, a general-purpose meta-search engine does not know in which domain a user is interested. For example, when we submit the word "RDF" to a general-purpose meta-search engine, we expect something about "Resource Description Framework" defined by W3C. However, many irrelevant results are returned, such as

- RDF Media http://www.rdf.co.uk/ ;
- **RDF Yellowstone Base** ;
- **Rapid Deployment Force** ;
- **Rose Drive Friends Church** ;
- RdF Corporation ;
- **RDF Racing Radio Direction Finding**;
- and so on.

This kind of semantic ambiguity problem occurs also in the special-purpose search systems. For example, in Experiment5, we use domain-specific search engines that are only for searching computer science publications. When we input "Cluster" to these search engines, we expect the retrieved documents to be relevant to Information Retrieval. However, because there is no more specific domain information in this simple query, the search engines return many publications on "Cluster-based network server" or "parallel computing", which are not what we want.

Now we analyze the experimental results in Table 7.17 (Figure 7.4 graphically illustrates the result data):

Table 7.17 The results of the second group of experiments

	Experiment4 Visible Web (*Mean of SavvySearch, Profusion, MetaCrawler*)		**Experiment5** Invisible Web (*SPOMSE*)	
Test set	TS1 (*words*)	TS2 (*phrases*)	TS1	TS2
Mean returned hits per query	41.75	24.33	1127.73	115.83
Mean relevant hits per query	1.08	4.44	103.4	37.83
Precision (%)	**2.59**	**18.24**	**9.17**	**32.66**

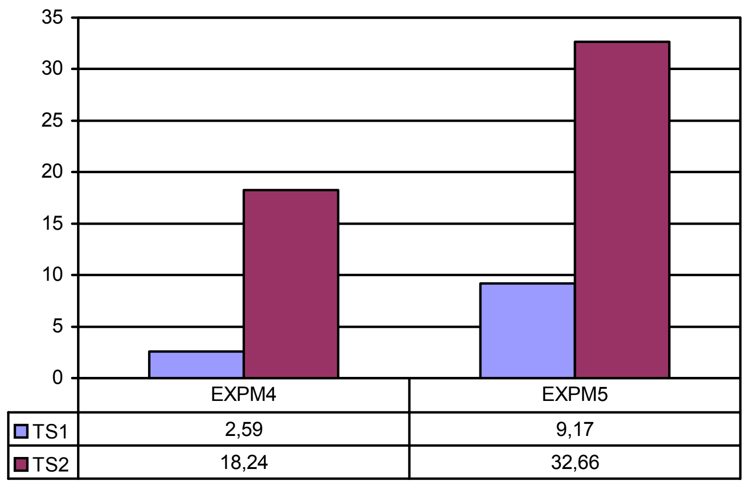


Figure 7.4 Average precision of the experiments' results

In **Experiment4**, because general-purpose Web meta-search engines greatly tailor the results (in [SHMM99], the authors observed the fact that almost 85% of users don't request beyond just the first results screens for their query) from multiple search engines and only return the most probably relevant results to the users, the number of returned hits for each query is not high. Nevertheless, their precision is not high (2.59% for single words and 18.24% for phrases). Because general-purpose Web search engines only index public-accessible pages (not specialized local databases), their retrieval precision is not high for specific information needs. In [Gra00], the author says: "web search technology is far from mature, and there is still plenty of room for improvement. More specifically, users are often overwhelmed with query results that include many irrelevant pages". [SE00] also observes: "general-purpose search engines such as AltaVista and Lycos are notorious for returning irrelevant results in response to user queries". In this experiment, we observed that most relevant hits are found on personal homepages and the Web pages of research institutions and conference organizers.

In **Experiment5**, because our prototype only visits publication-oriented search engines, the retrieval precision is higher than **Experiment4**. Why do domain-specific search engines achieve higher retrieval precision than general-purpose search engines? Paper [SE00] gives three reasons: "Topic-specific search engines often return higher-quality references than broad, general-purpose search engines for several reasons. First, specialized engines are often a front-end to a database of authoritative information that search engine spiders, which index the Web's HTML pages, cannot access. Second, specialized search engines often reflect the efforts of organizations, communities, or individual fanatics that are committed to providing and updating high-quality information. Third, because of their narrow focus and smaller size, word-sense ambiguities and other linguistic obstacles to high-precision search are ameliorated."

From tables 7.13-7.16, we can see that the precision of searching phrases is higher than that of searching single keywords. In [BMD00], some experiments also suggest that the phrase-based query reformulation "can significantly improve the relevance of the documents through which the user must trawl versus standard query-based Internet search". The experimental results in Table 7.17 suggest that a meta-search engine integrating special-purpose search engines can achieve higher precision than general-purpose search engines and meta-search engines for users' specific information needs. In section 7.2, the Experiment3 shows that a scientific publication-oriented meta-search engine, utilizing the constraints of field modifiers, term qualifiers and logical operators, will greatly improve the effectiveness of searching for scientific publications, especially of pinpointing a paper (the precision is 58.33%).

7.4 Generated sub-queries and post-filters

In this section, this group of experiments has been carried out to test the query translation algorithm introduced in Chapter 4. This group of experiments uses the 8 query expressions the same as in TS3. For all these 8 experimental queries, Table 7.18 displays the total number of sub-queries and total number of post filters (= total feasible filters + total skipped filters) generated for each of the three sources. Feasible filters can be applied to refine the results; while skipped filters have been ignored by the system due to impossibility or tremendous cost. For each of these 8 tested queries, the average number of generated sub-queries for a source is 2.375, the average number of feasible filters is 1.29, and the average number of skipped filters is 0.583. For example, the fourth query expression (QE4) in Table 7.5 can be broken into 3 sub-queries and 1 feasible filter for the NCSTRL search engine (see Table 4.5 in Chapter 4) and 2 sub-queries and 1 feasible filter for the ACM-DL search engine (see Table 4.6 in Chapter 4).

Table 7.18 The number of generated sub-queries and post filters

	Total sub-queries	Total feasible filters	Total skipped filters
ACM-DL	22	7	2
CORA	9	13	7
NCSTRL	26	11	5

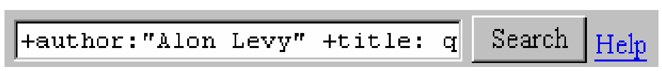


Figure 7.5 Simple query interface of the Cora search engine

From Figure 7.5, we can see that the CORA search engine provides a very simple user interface (only one input-box). Therefore, for each conjunctive query, its all terms have been put into this input-box. When translating all 8 queries into the query format of the CORA search engine, 9 sub-queries (1.125 per query) and 20 filters (2.5 per query) are generated. Compared with the CORA search engine, the NCSTRL search engine provides more restrictive query input form (see Figure 1.2). Therefore, 26 sub-queries (3.25 per query) and 16 filters (2 per query) are generated. Generally speaking, the simpler the user interface, the less sub-queries and more filters may be generated; the richer the user interface, the more sub-queries and fewer filters may be generated. The number of generated sub-queries and filters also depends on both the original query itself and the query capability of the target source.

Chapter 8

Conclusions

This chapter consists of three parts. We first summarize this thesis in section 7.1. Then in section 7.2, we introduce some spheres to which the meta-search technologies can be applied. Finally, in section 7.3, we discuss some future research directions.

8.1 Summary

Searching the Web (especially the Invisible Web) accurately is becoming increasingly critical as the Web grows. In this dissertation we have provided a formal description of the query capability of heterogeneous search engines and an algorithm for translating queries from a mediator to a specific target search engine. From the previous discussions, we know that there is great diversity among search engines. Exact and efficient query translation is a complicated and significant task.

The contributions of this dissertation are: (1) We formally analyze the query user interfaces of Internet information sources and propose a query model that meticulously describes the query capabilities of heterogeneous sources. (2) We develop a constraint-based query-translation algorithm that will automatically and efficiently resolve the conflicts between diverse query models supported by different sources and a two-phase query subsuming mechanism is put forward to compensate for the functional discrepancies between sources, in order to make a more accurate query translation. (3) Control constraint rules are employed to dynamically construct the adaptive user interface of an information integration system, and this will make the user queries consistent with the formats of specific target services to the full and make the system scale well.

Our experiments show that an information integration system with an adaptive, dynamically generated user interface, coordinating the constraints among the heterogeneous sources, will greatly improve the effectiveness of integrated information searching, and will utilize the query capabilities of sources as far as possible. Now, the adaptive meta-search engine architecture proposed in this paper has been applied to the information integration of scientific publications-oriented search engines. It can also be applied to other generic or specific domains of information integration, such as integrating all kinds of (especially special-purpose) WWW search engines (or search tools) and online repositories with quite different user interfaces and query models. With the help of source wrapping tools, they can also be used to integrate queryable information sources delivering semi-structured or non-structured data, such as telephone directories, product catalogues, weather reports, software directories, stock quotes, job listings, and so on.

8.2 Application spheres

The World Wide Web is rapidly emerging as an important medium for the dissemination of information related to a wide range of topics. Undoubtedly, now searching the (visible/invisible) Web is the most important approach to finding information for almost all users. Any organizations, companies and individuals will publish information on the Web if they want it to be known publicly. A meta-search engine can be used in the following spheres:

In enterprises, it can help people who are working on market research, decision support and competitive intelligence. By using it, enterprise analysts can simply formulate a single query in a uniform user interface to locate the information they need, rather than accessing several different internal and external sources separately. Therefore, they can easily monitor all marketing and commercial information concerning their businesses worldwide and can answer and respond very quickly to questions on specific subjects.

In organizations, researchers, librarians, and other information workers can profit from it. Now almost all publishing houses (e.g., ACM, IEEE, Elsevier, Springer, etc.)

and news presses (CNN, BBC, etc.) publish electronic journals, conference proceedings, and news on the Web. General-purpose search engines (e.g., Google) cannot index these databases (e.g., ACM) or cannot timely index the frequently updated information (BBC news website updates its content every minute). Therefore, using a meta-search engine to search on these databases (Web pages) can help users retrieve these kinds of valuable information.

It can serve individuals as a personal web agent. When a teenager wants to search for some information of a movie star, she or he will be very happy if a meta-search engine can return tons of results from all kinds of sources, such as websites of fan clubs, movie databases, movie companies, celebrities' photo galleries, and so on. A housewife can also use a meta-search engine that integrates several shopping websites as a personal shopping assistant.

8.3 Future research directions

This section presents some future research areas that can be integrated into our current meta-search engine prototype: question answering, interactive relevance feedback, synthesizing the query capabilities of different search engines, wrapper maintenance, and so on.

Question answering

Current Web search engines can retrieve documents that include keywords but they cannot retrieve precise information in answer to precise queries, e.g., "what is the cheapest way to travel from Darmstadt, Germany to Santa Fe, NM, USA?", or "what is the fastest way to travel from Santa Fe, NM, USA to Tokyo, Japan?". Answering such queries requires a large-scale cooperatively-built knowledge base. Combining a meta-search engine with natural language parsing technologies can yield reliable answers. There are some challenges:

- How to semantically understand and classify natural questions containing interrogative words such as "where", "when", "which", "how", "how long", "how many", "what", and so on?

- How to translate natural language questions into queries understood by search engines that only support keywords or Boolean queries?
- How to analyze the results documents and extract answers from them?
- How to utilize the semantic and syntactic information available both in the queries and in the documents?
- How to judge which answer is the best one? That is: how to rank the answers set?

Interactive relevance feedback

People often try to search the Web with poorly specified information needs. Existing search engines expect users to give well-specified queries, which is difficult for the people whose information needs are vague. Relevance feedback, which modifies the original query based on the user's judgment of previous search results, may support the users somewhat. It is essential to provide users with a sophisticated user interface that allows users to control the relevance feedback search process and understand the relationships between queries and retrieved documents.

Synthesizing the query capabilities of different search engines

The query capability of one single search engine is limited. Synthesizing the query capabilities of different search engines can achieve more powerful retrieving effect. For example, Mary wants to search for the following information: "what is the cheapest way to travel from Darmstadt, Germany to Santa Fe, NM, USA?". Unfortunately, there is no search engine or information source that can answer this question. Therefore, she searches for some Web sites (e.g., "Deutsche Bahn", "Lufthansa", etc.) for tickets' prices of different traffic means. After that, she may get tens of traveling plans, for example, (1) "Darmstadt-(*bus*)-Frankfurt airport-(*Lufthansa airline*)-Huston-(*Continental airline*)-Albuquerque-(*bus*)-Santa Fe"; (2) "Darmstadt-(*train*)-Köln airport-(*Canada airline*)-New York-(*Delta airline*)-Denver-(*American airline*)-Albuquerque-(*bus*)-Santa Fe"; and so on. Finally, she chooses a cheapest way from them.

A meta-search engine can also do this kind of things for Internet information seekers. It searches for different Web sites, analyzes the results and synthesizes the final answers. One big challenge is how to break down the original multi-information source query into a collection of sub-queries and how to target each sub-query to a certain single source. For example, one airline company may provide a very cheap last-minute ticket for traveling from Berlin via Tel Aviv to Chicago. This sub-traveling plan can be the way for users to cross the Atlantic.

Wrapper maintenance

Because the information on the WWW changes frequently, the administrators of meta-search engine will timely modify the wrapper if they detect the changes in the query user interfaces of target search engines, digital libraries and information sources. For example, in Figure 7.1, there are three cropped screenshots of query input interfaces of ACM-DL that have been evolving these two years from Figure 7.1(a) to Figure 7.1(b) to Figure 7.1(c). The conventional way is to manually modify the wrapper. This method is tedious and time-consuming, especially when a meta-search engine integrates thousands of sources or more. But this method can exactly and correctly maintain the wrapper. Considering the tremendous information and fast growth of the WWW, automatically or semi-automatically maintaining (detecting and modifying) the wrappers is very important. However, automatically maintaining wrappers is not an easy thing. The meta-search engine can draw some rules from the information of the controls in the HTML query page and use heuristics to recognize the functionalities of new controls in the query page. One feasible way for maintaining results pages could be using Hidden Markov Models on learning the state and transition structure of results pages.

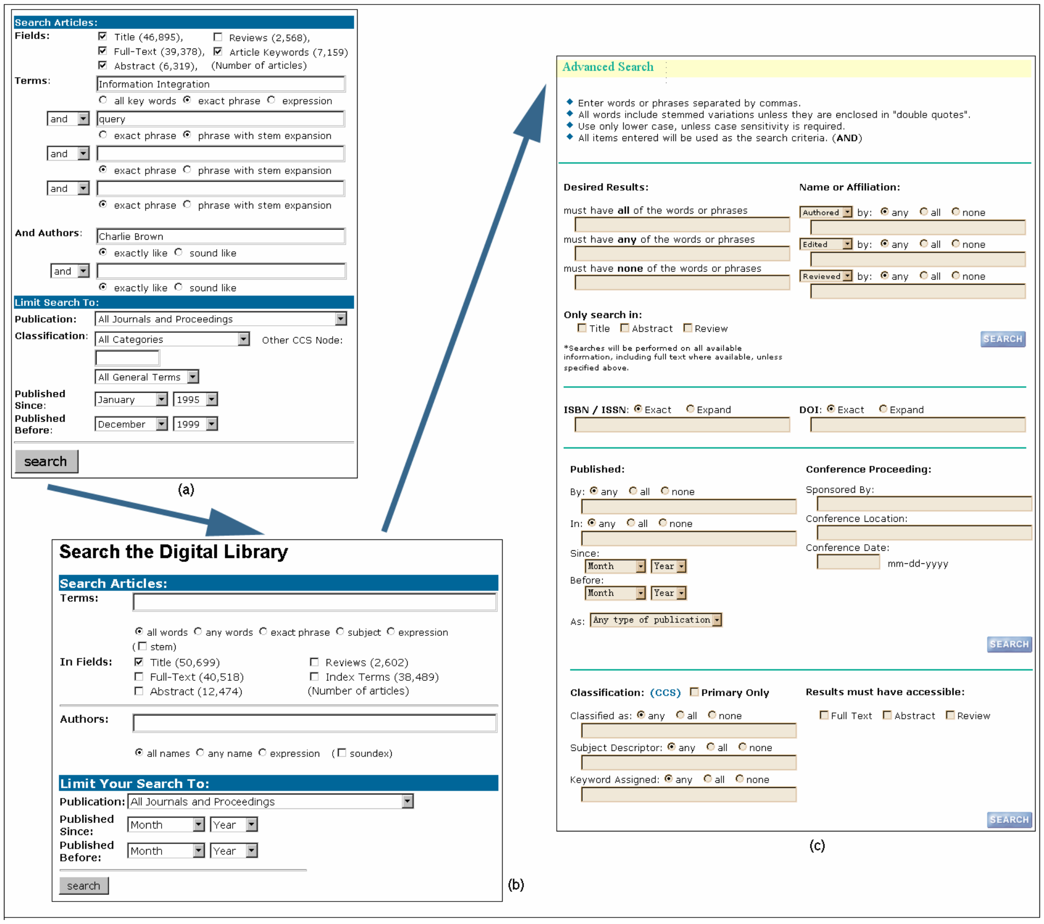


Figure 8.1 a query user interface changing example of information sources

Other issues

There are many other interesting issues: e.g. Combining multimedia search engines; Providing users with guidance help and friendly user interface for query constructing and result browsing; Researching and applying collaborative filtering and recommending technologies; More efficient source selection methods; Ranking results from heterogeneous sources, and so on.

References

[1] Google search engine (http://www.google.com/).

[2] AltaVista search engine (http://www.altavista.com/).

[3] Inforseek search engine (http://www.go.com/).

[4] HotBot search engine (http://hotbot.lycos.com/).

[5] Northern Light Search (http://www.northernlight.com/).

[6] WebCrawler search engine (http://www.webcrawler.com/).

[7] ACM-Digital Library (http://www.acm.org/dl/newsearch.html).

[8] IDEAL - International Digital Electronic Access Library (http://www.idealibrary.com/servlet/useragent?func=showSearch).

[9] NCSTRL – Networked Computer Science Technical Reference Library (http://cs-tr.cs.cornell.edu/).

[10] (http://www.weather.com/).

[11] Computer Science Bibliography (http://www.informatik.uni-trier.de/~ley/db/).

[12] (http://www.mp3.com/).

[13] Elsevier Science (http://www.elsevier.nl/homepage/search.htt?pubidt=525444&sarea=sac).

[14] Kluwer academic publishers Online (http://www.wkap.nl/kaphtml.htm/CATSEARCH).

[15] ERCIM Technical Reference Digital Library (http://ncstrl.gmd.de:80/Dienst/UI/2.0/Search?tiposearch=ercim&langver=en).

[16] Cora Computer Science Research Paper Search Engine (http://cora.whizbang.com/).

[17] Websearch.about.com (http://websearch.about.com/internet/websearch/library/weekly/aa061199.htm).
[18] DeliteOnline (http://www.ipsi.fhg.de/delite/Projects/DeliteOnline/index.html).
[19] Inference Find (http://www.infind.com/).
[20] Cyber411 (http://www4.c4.com).
[21] Internet Sleuth (http://www.isleuth.com/).
[22] MAMMA (http://www.mamma.com/).
[23] User-oriented Mediation and Presentation of Scientific Multimedia Information (http://www.tu-darmstadt.de/iuk/global-info/sfm-7/).
[24] SavvySearch (http://search.com).
[25] AskJeeves (http://www.askjeeves.com).
[26] MetaCrawler (http://www.metacrawler.com).
[27] Dogpile (http://www.dogpile.com/index.gsp).
[28] Highway 61 (http://www.highway61.com).
[29] I.SEE (http://www.cs.rpi.edu/~bufic/I.SEE/Engine/index.html).
[30] Infomine Multiple Database Search (http://infomine.ucr.edu/search.phtml).
[31] Lycos Invisible Web Catalog (http://dir.lycos.com/Reference/Searchable%5FDatabases/).
[32] AlphaSearch (http://www.calvin.edu/library/searreso/internet/as/).
[33] WebData (http://www.webdata.com/).
[34] Common Object Request Broker Architecture, http://www.omg.org/corba/.
[35] eXtensible Markup Language, http://www.w3.org/Metadata/RDF/.
[36] Resource Description Framework, http://www.w3.org/Metadata/RDF/.
[37] (http://it.ncsa.uiuc.edu/~mag/work/XMLTalk/index.html).
[38] (http://www.xml.com/xml/pub/98/10/guide1.html).
[39] (http://www.w3.org/).
[40] Dewey Decimal Classification (http://www.oclc.org/dewey/).
[41] Library of Congress Classification Outline (http://www.loc.gov/catdir/cpso/lcco/lcco.html).

[42] The ACM Computing Classification System (1998) (http://www.acm.org/class/1998/TOP.html).

[43] AMS Mathematics Subject Classification (2000) (http://www.ams.org/msc/).

[44] Online Computer Library Center (http://www.oclc.org/oclc/menu/home1.htm).

[45] Cyber411 (http://www.c4.com).

[46] Internet Sleuth (http://www.isleuth.com).

[47] Oneseek (http://www.oneseek.com).

[48] XSchema (http://purl.ocls.org/NET/xschema).

[49] CGI- Common Gateway Interface (http://www.w3.org/CGI/).

[AB98] Sibel Adali and Corey Bufi. A Flexible Architecture for Query Integration and Mapping. In Proceedings of the Third IFCIS Conference on Cooperative Information Systems (CoopIS'1998), Pages 341-353. New York, USA, August 20 - 22, 1998.

[ABT97] Sibel Adali, Corey Bufi, and Yaowadee Temtanapat. Integrated Search Engine. In Proceedings of the IEEE Knowledge and Data Engineering Exchange Workshop, KDEX97. Pages 140-147, Newport Beach, California, USA, November 4, 1997.

[ACHK93] Yigal Arens, Chin Y. Chee, Chunhan Hsu, and Craig Knoblock. Retrieving and Integrating Data from Multiple Information Sources. International Journal on Intelligent and Cooperative Information System, Vol. 2, No. 2, Pages 127-158, 1993.

[ACPS96] Sibel Adali, Kasim S. Candan, Yannis Papakonstantinou, and V. S. Subrahmanian. Query Caching and Optimization in Distributed Mediator Systems. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Pages 137-146, Montreal, Canada, June 3-6, 1996.

[AFT98] Laurent Amsaleg, Michael J. Franklin, and Anthony Tomasic. Dynamic query operator scheduling for wide-area remote access. Journal of Distributed and Parallel Databases, Vol. 6, No. 3, Pages 217-246, July 1998.

[AH00] Ron Avnur and Joseph M. Hellerstein. Eddies: Continuously adaptive query processing. In Proceedings of the 2000 ACM SIGMOD

International Conference on Management of Data. Pages 261-272, Dallas, Texas, USA, May 16-18, 2000.

[AKMP98] Jose Luis Ambite, Craig A. Knoblock, Ion, Muslea, and Andrew Philpot. Compiling Source Descriptions for Efficient and Flexible Information Integration. Technical report, Information Sciences Institute, University of Southern California, USA. 1998.

[AKS96] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen. Query reformulation for dynamic information integration. Journal of Intelligent Information Systems, Special Issue on Intelligent Information Integration, 6(2/3), Pages 99-130. 1996.

[AS96] Maristella Agosti and Alan Smeaton (editors). Information Retrieval and Hypertext. Kluwer Academic Publishers, 1996.

[ASD+91] Rafi Ahmed, Philippe De Smedt, Weimin Du, William Kent, Mohammad A. Ketabchi, Witold Litwin, Abbas Rafii, and Ming-Chien Shan. The Pegasus Heterogeneous Multidatabase System. IEEE Computer, Vol. 24, No. 12, Pages 19-27. 1991.

[BB98] Krishna Bharat and Andrei Broder. A technique for Measuring the Relative Size and Overlap of Public Web Search engines. In Proceedings of the 7th World Wide Web Conference, Pages 379-388, Brisbane, Australia, April 14-18, 1998.

[BBB+97] Roberto J. Bayardo, Bill Bohrer, Richard S. Brice, Andrzej Cichocki, Jerry Fowler, Abdelsalam Helal, Vipul Kashyap, Tomasz Ksiezyk, Gale Martin, Marian H. Nodine, Mosfeq Rashid, Marek Rusinkiewicz, Ray Shea, C. Unnikrishnan, Amy Unruh, and Darrell Woelk. InfoSleuth: Semantic Integration of Information in Open and Dynamic Environments. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Pages 195-206, Tucson, AZ USA, May 11 - 15, 1997.

[BBE98] Athman Bouguettaya, Boualem Benatallah and Ahmed K. Elmagarmid. Interconnecting Heterogeneous Information Systems. Kluwer Academic Pub; ISBN: 0792382161. 1998.

[BBEK98] Andreas Barth, Michael Breu, Albert Endres, and Arnoud de Kemp. (Eds.) Digital Libraries in Computer Science: The MeDoc Approach. Springer, 1998.

[BDM+95] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. Harvest: A Scalable, Customizable Discovery and Access System. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado - Boulder. August 26, 1994.

[BDMS94] C. Mic Bowman, Peter B. Danzig, Udi Manber, and Michael F. Schwartz. Scalable Internet resource discovery: research problems and approaches. Communications of the ACM, Vol. 37, No. 8, Page 98-114, August 1994.

[Berg00] Michael K. Bergman. The deep Web: Surfacing the hidden value. BrightPlanet, www.complete-planet.com/Tutorials/DeepWeb/index.asp. 2000.

[BFPV01] Luc Bouganim, Francoise Fabret, Pabio Porto, and Patrick Valduriez. Processing Queries with Expensive Functions and Large Objects in Distributed Mediator Systems. In Proceedings of the 17th International Conference on Data Engineering (ICDE 2001), Pages 91-98, IEEE Computer Society Press, Heidelberg, Germany, April 2-6, 2001.

[BGL+99] Chaitan Baru, Amarnath Gupta, Bertram Ludäscher, Richard Marciano, Yannis Papakonstantinou, Pavel Velikhov, and Vincent Chu. XML-Based Information Mediation with MIX. In Proceedings of ACM SIGMOD International Conference on Management of Data, Pages 597-599, Philadelphia, PA, USA, May 31 - June 3, 1999.

[Bir95] William P. Birmingham. An agent-based architecture for digital libraries. D-Lib Magazine, July 1995.

[BJN97] Franz Baader, Manfred A. Jeusfeld, and Werner Nutt. (Ed.) Intelligent Access to Heterogeneous Information. Proceedings of the 4th Workshop KRDB-97 (Knowledge Representation meets Databases) Athens, Greece, August 30, 1997.

[BKP+98] Michelle Q. Wang Baldonado, Seth Katz, Andreas Paepcke, Kevin Chen-Chuan Chang, Hector Garcia-Molina, and Terry Winograd. An Extensible Constructor Tool for the Rapid, Interactive Design of Query Synthesizers. In Proceedings of 3rd ACM International Conference on Digital Libraries, Pages 19-28, Pittsburgh, PA, USA, June 23-26, 1998.

[BLCG92] Tim Berners-Lee, Robert Cailliau, and Jean-François Groff: The World-Wide Web. Computer Networks and ISDN Systems, Vol. 25, No. 4-5, Pages 454-459, 1992.

[BLC+94] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen, and Arthur Secret: The World-Wide Web. Communication of the ACM, Vol. 37, No. 8, Pages 76-82, August 1994.

[BL96] Tim Berners-Lee: WWW: Past, Present, and Future. IEEE Computer, Vol. 29, No. 10, Pages 69-77, August 1996.

[BMD00] Peter Bruza, Robert McArthur, and Simon Dennis. Interactive Internet search: keyword, directory and query reformulation mechanisms compared. In Proceedings of the 23rd ACM SIGIR Conference on Research and Development in Information Retrieval, Pages 280-287, Athens, Greece. July 24-28, 2000.

[BP98] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In Proceedings of WWW7 / Computer Networks 30(1-7), Pages 107-117, Brisbane, Australia, April 14-18, 1998.

[Bra97] David Brake. Lost in Cyberspace, New Scientist, June 28, 1997.

[BRV98] Laura Bright, Louiqa Raschid, and Maria-Esther Vidal. Optimization of Wrappers and Mediators for Web Accessible Data Sources (WebSources). In Proceedings of CIKM'98 Workshop on Web Information and Data Management (WIDM'98), Washington, DC, USA, November 6, 1998.

[BS97] Sonia Bergamaschi and Claudio Sartori. An Approach for the Extraction of Information from Heterogeneous Sources of Textual Data. Proceedings of the 4th KRDB Workshop, Athens, Greece, August 30, 1997.

[BT98] Philippe Bonnet and Anthony Tomasic. Partial Answers for Unavailable Data Sources. In Proceedings of the 3rd International Conference on Flexible Query Answering Systems (FQAS'98), Pages 43-54, Roskilde, Denmark, May 13-15, 1998.

[BW97] Michelle Q. Wang Baldonado and Terry Winograd. SenseMaker: An Information-Exploration Interface Supporting the Contextual Evolution of a User's Interests. In Proceedings of the ACM Conference on Human

Factors in Computing Systems (CHI '97), Pages 11-18, Atlanta, Georgia, USA, March 22-27, 1997.

[BW98] Michelle Q. Wang Baldonado and Terry Winograd. A GUI-Based Version of the SenseMaker Interface for Information Exploration. http://cs-tr.cs.cornell.edu:80/Dienst/UI/1.0/Display/stanford.cs/CS-TN-98-67. 1998

[Cal00] James P. Callan. Searching for Needles in a World of Haystacks, In IEEE Data Engineering Bulletin, Special Issue on Next Generation Web Search, Luis Gravano (Ed.), Vol. 23, No. 3, Pages 33-37, September 2000.

[CBC98] Boris Chidlovskii, Uwe M. Borghoff, and Pierre-Yves. Chevalier. Boolean Query Translation for Brokerage on the Web. In Proceedings of the 2nd International Conference EuroMedia/WEBTEC'98, Pages 37-44, Leicester, U.K. January 5-7, 1998.

[CC94] Wesley W. Chu and Qiming Chen. A Structured Approach for Cooperative Query Answering. IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No. 5, Pages 738-749, October 1994.

[CC03] Jack G. Conrad and Joanne R. S. Claussen. Early User – System Interaction for Database Selection in Massive Domain-Specific Online Environments. ACM Transactions on Information Systems, Vol. 21, No. 1, Pages 94-131, January 2003.

[CDTW00] Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. NiagaraCQ: A scalable continuous query system for Internet databases. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Pages 379-390, Dallas, Texas, USA, May 16-18, 2000.

[CGJM02] Jack G. Conrad, Xi S. Guo, Peter Jackson, and Monem Meziou. Database selection using complete physical and acquired logical collection resources in a massive domain-specific operational environment. In Proceedings of 28th VLDB Conference, Pages 71-82, Hong Kong, China, August 20-23, 2002.

[CDSS98] Sophie Cluet, Claude Delobel, Jérôme Siméon, and Katarzyna Smaga. Your Mediators Need Data Conversion. In Proceedings of the 1998

ACM SIGMOD International Conference on Management of Data. Pages 177-188, Seattle, WA, USA, June 2-4, 1998.

[CGM97] Chen-Chuan K. Chang and Héctor García-Molina. Evaluating the Cost of Boolean Query Mapping. In Proceedings of the 2nd ACM International Conference on Digital Libraries, Pages 103-112, Philadelphia, PA, USA, July 23 - 26, 1997.

[CGM99] Chen-Chuan K. Chang and Héctor García-Molina. Mind Your Vocabulary: Query Mapping Across Heterogeneous Information Sources. In Proceedings of the 1999 ACM SIGMOD International Conference On Management of Data, Pages 335-346, Philadelphia, PA, USA, May 31 - June 3, 1999.

[CGM00] Chen-Chuan K. Chang and Héctor García-Molina. Approximate Query Translation Across Heterogeneous Information Sources. In Proceedings of the 26th VLDB Conference, Pages 566-577, Cairo, Egypt, 2000.

[CGMP96] Chen-Chuan K. Chang, Héctor Garcia-Molina, and Andreas Paepcke. Boolean Query Mapping Across Heterogeneous Information Sources. IEEE Transactions on Knowledge and Data Engineering, Pages 515-521, Vol. 8, No. 4, August 1996.

[CGMP99] Chen-Chuan K. Chang, Héctor Garcia-Molina, and Andreas Paepcke. Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System. ACM Transactions on Information Systems, Vol. 17, No. 1, Pages 1-39, January 1999.

[Chri96] Eliot Christian. GILS: What is it? Where's it going? D-Lib Magazine, December 1996.

[CHS+95] Michael J. Carey, Laura Haas, Peter M. Schwarz, Manish Arya, William F. Cody, Ronald Fagin, Myron Flickner, Allen W. Luniewski, Wayne Niblack, Dragutin Petkovic, John Thomas, John Williams, and Edward L. Wimmers. Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. In Proceedings of 1995 IEEE Workshop on Research Issues in Data Engineering (RIDE-95), Pages 124-131, Taipei, Taiwan, March 6-7, 1995.

[CJ82] M. I. Crystal and G. E. Jakobson. FRED, a front end for databases. Online, Vol. 6, No.5, Pages 27-32, September 1982.

[CLC95] James P. Callan, Zhihong Lu, and W. Bruce Croft. Searching Distributed Collections with Inference Networks. In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pages 21-28, Seattle, Washington, USA, June 9-13, 1995.

[CP84] Stefano Ceri and Giuseppe Pelagatti. Distributed Databases: Principles and Systems. McGraw-Hill Book Company, New York, 1984.

[CPS+99] Michael Christoffel, Sebastian Pulkowski, Bethina Schmitt, Peter Lockemann, and Christoph Schütte. The UniCats Approach - New Management for Books in the Information Market. In Proceedings of the International Conference IuK99 - Dynamic Documents. Jena, Germany, March 22-24, 1999.

[CPW+97] Steve B. Cousins, Andreas Paepcke, Terry Winograd, Eric A. Bier, and Kenneth A. Pier. The Digital Library Integrated Task Environment (DLITE). In Proceedings of the 2nd ACM International Conference on Digital Libraries, Pages 142-151, Philadelphia, PA, USA, July 25-28, 1997.

[CR98] I-Min A. Chen and Doron Rotem. Integrating Information from Multiple Independently Developed Sources. In Proceedings of the 7th International Conference on Information Knowledge Management. (CIKM'98), Pages 242-250, Bethesda, Maryland, USA, November 3-7, 1998.

[Cra84] Walt Crawford. MARC for Library Use: Understanding the USMARC Formats. Knowledge Industry Publications, Inc., White Plains, NY. 1984.

[CYC+96] Wesley W. Chu, Hua Yang, Kuorong Chiang, Michael Minock, Gladys Chow, and Chris Larson. CoBase: A Scalable and Extensible Cooperative Information System, Journal of Intelligence Information Systems, Vol. 6, No. 2-3, Pages 223-259, Kluwer Academic Publishers, May, 1996.

[DEW97] Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalable comparison-shopping agent for the world-wide web. In Proceedings First International Conference on Autonomous Agents (Agents 1997), Pages 39-48, Marina del Rey, CA, USA, February 5-8, 1997.

[DFGL98] Markus Dreger, Norbert Fuhr, Kai Großjohann, and Stephan Lohrum. Provider Selection Design and Implementation of the Medoc Broker. In: Digital Libraries in Computer Science: The Medoc Approach, Lecture Notes in Computer Science 1392, Pages 67-78, ISBN 3-540-64493-8, Springer 1998.

[DH97] Daniel Dreilinger and Adele E. Howe. Experiences with Selecting Search Engines Using Meta-Search. ACM Transactions on Information Systems. Vol. 15, No. 3, Pages 195-222, July 1996.

[DM97] Stefan Deßloch and Nelson Mendonça Mattos. Integrating SQL databases with content-specific search engines. In Proceedings of the 23rd VLDB Conference, Pages 528-537, Athens, Greece, August 25-29, 1997.

[Etz96] Oren Etzioni. The World-Wide Web: Quagmire or Gold Mine? Communications of the ACM, Vol. 39, No. 11, Pages 65-68, November 1996.

[FAD+99] Dieter Fensel, Jürgen Angele, Stefan Decker, Michael Erdmann, Hans-Peter Schnurr, Steffen Staab, Rudi Studer, and Andreas Witt. On2broker: Semantic-Based Access to Information Sources at the WWW. In Proceedings of WebNet 99, Pages 366-371. Hololulu, Hawaii, USA, October 25-30, 1999.

[FBA+90] Daniel H. Fishman, David Beech, Jurgen Annevelink, E. C. Chow, Tim Connors, J. W. Davis, Waqar Hasan, C. G. Hoch, William Kent, S. Leichner, Peter Lyngbæk, Brom Mahbod, Marie-Anne Neimat, Tore Risch, Ming-Chien Shan, and W. Kevin Wilkinson. Overview of the Iris DBMS. In Won Kim and Frederick H. Lochovsky (eds.): Research Foundations in OO and Semantic DBS, Pages 174-199, 1990.

[FDFP95] Adam Farquhar, Angela Dappert, Richard Fikes, and Wanda Pratt. Integrating Information Sources Using Context Logic. In Proceedings of the AAAI-95 Spring Symposium on Information Gathering from Distributed Heterogeneous Environments. Stanford University, California, USA, March 27-29, 1995.

[FFMM94] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. KQML as an Agent Communication Language. In Proceedings of the Third International Conference on Information and Knowledge

Management (CIKM'94), Pages 456-463, Gaithersburg, Maryland, USA, ACM Press, November 29 - December 2, 1994.

[FLMS99] Daniela Florescu, Alon Levy, Ioana Manolescu, and Dan Suciu. Query optimization in the presence of limited access patterns. In Proceedings of ACM SIGMOD International Conference on Management of Data, Pages 311-322, Philadelphia, PA, USA, May 31- June 3, 1999.

[Frei98] Dayne Freitag. Information extraction from HTML: application of a general machine learning approach. Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, Pages 517-523, Madison, Wisconsin, USA, July 26-30, 1998.

[FSF95] Béatrice Finance, Veronique Smahi, and Jerome Fessy. Query Processing in IRO-DB. In Proceedings of International Conference on Deductive and Object-Oriented Databases (DOOD'95), Pages 299-318, Singapore, December 4-7, 1995.

[Fuhr92] Norbert Fuhr. Probabilistic Models in Information Retrieval. The Computer Journal, Vol. 35, No. 3, Pages 243-255, 1992.

[Fuhr99] Norbert Fuhr. A decision-theoretic approach to database selection in networked IR. ACM Transactions on Information Systems (TOIS), Vol. 17, No. 3, Pages 229-249, May 1999.

[Gal99] Avigdor Gal. Semantic Interoperability in Information Services: Experiencing with CoopWARE, SIGMOD Record, Vol. 28, No. 1, Pages 68-75, March 1999.

[GBL98] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In Proceedings of the third ACM Conference on Digital libraries, Pages 89-98, Pittsburgh, PA, USA, June 23-26, 1998.

[GCGMP97] Luis Gravano, Chen-Chuan K. Chang, Héctor García-Molina, and Andreas Paepcke. STARTS: Stanford Protocol Proposal for Internet Retrieval and Search. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Pages 207-218, Tucson, AZ, USA, ACM Press, New York, NY, May 11 - 15, 1997.

[GGM95] Luis Gravano and Héctor García-Molina. Generalizing GlOSS for vector-space databases and broker hierarchies. In Proceedings of the

21st VLDB Conference, Pages 78-89, Zurich, Switzerland, Morgan Kaufman, Los Altos, California. September 11-15, 1995.

[GGM97] Luis Gravano and Héctor García-Molina. Merging ranks from heterogeneous Internet sources. In Proceedings of the 23rd VLDB Conference, Pages 196-205, Athens, Greece, August 25-29, 1997.

[GGMT94] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. The effectiveness of GlOSS for the text database discovery problem. In Proceedings of 1994 ACM SIGMOD International Conference on Management of Data, Pages 126-137, Minneapolis, MN, USA, May 24 - 27, 1994.

[GGMT99] Luis Gravano, Héctor García-Molina, and Anthony Tomasic. GlOSS: Text source discovery over the Internet. ACM Transactions on Information Systems (TOIS), Vol. 24, No. 2, Pages 229-264, June, 1999.

[GIG01] Noah Green, Panagiotis G. Ipeirotis, and Luis Gravano. SDLIP + STARTS = SDARTS : A protocol and toolkitD for metasearching. In Proceedings of 2001 ACM/IEEE Joint Conference on Digital Libraries (JCDL 2001), Pages 207-214, Roanoke, Virginia, USA, June 24-28, 2001.

[GIS03] Luis Gravano, Panagiotis G. Ipeirotis, and Mehran Sahami. QProber: A System for Automatic Classification of Hidden-Web Databases. ACM Transactions on Information Systems, Vol. 21, No. 1, Pages 1-41, January 2003.

[GLBG99] Eric J. Glover, Steve Lawrence, William P. Birmingham, and C. Lee Giles. Architecture of a metasearch engine that supports user information needs. In Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management (CIKM'99), Pages 210-216, Kansas City, MO, USA, Pages 210-216, November 2-6, 1999.

[GMHI+95] Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. Integrating and Accessing Heterogeneous Information Sources in TSIMMIS and its Components Translators and Common Model. In Proceedings of the AAAI-95 Spring Symposium on Information Gathering from

Distributed Heterogeneous Environments. Stanford University, California, USA, March 27-29, 1995.

[GMLY99] Hector Garcia-Molina, Wilburt Labio, and Ramana Yerneni. Capability-Sensitive Query Processing on Internet Sources. In Proceedings of the 15th International Conference on Data Engineering (ICDE 99), Pages 50-59, Sydney, Australia, IEEE Computer Society Press, March 23 - 26 1999.

[GMPQ+97] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. The TSIMMIS approach to mediation: Data models and Languages. In Journal of Intelligent Information Systems (JIIS), Vol. 8. No. 2, Pages 117-132, Kluwer Academic Publishers, The Netherlands, March/April 1997.

[GMS94] Cheng Hian Goh, Stuart E. Madnick, and Michael Siegel. Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment. In Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94), Pages 337-346, Gaithersburg, Maryland, USA, November 29 - December 2, 1994.

[GMV99] Nicola Guarino, Claudio Masolo, and Guido Vetere. OntoSeek: Content-Based Access to the Web, IEEE Intelligent Systems, Vol. 14, No. 3, Pages 70-80, May 1999.

[GMW99] Roy Goldman, Jason McHugh, and Jennifer Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99), Pages 25-30, Philadelphia, Pennsylvania, USA, June 3-4, 1999.

[GMY99] Hector Garcia-Molina and Ramana Yerneni. Coping with Limited Capabilities of Source. In Proceedings of the Datenbanksysteme in Büro, Technik und Wissenschaft (BTW 1999), Pages 1-19, GI-Fachtagung, Freiburg, Germany, March 1-3, 1999.

[GR94] Jade Goldstein and Steven F. Roth. Using aggregation and dynamic queries for exploring large data sets. In proceedings on Human factors in computing systems (CHI'94), Pages 23-29, Boston, USA, April 24 - 28, 1994.

[Gra00] Luis Gravano. Letter from the Special Issue Editor, In IEEE Data Engineering Bulletin, Special Issue on Next Generation Web Search, Luis Gravano (Ed.), Vol. 23, No. 3, Page 2. September 2000.

[GRVB98] Jean-Robert Gruser, Louiqa Raschid, Maria Esther Vidal, and Laura Bright. Wrapper Generation for Web Accessible Data Sources. In Proceedings of Third IFCIS Conference on Cooperative Information Systems (CoopIS'98), Pages 14-23, New York, NY, USA, August 20-22, 1998.

[GW00] Roy Goldman and Jennifer Widom. WSQ/DSQ: A Practical Approach for Combined Querying of Databases and the Web. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. Pages 285-296, Dallas, Texas, USA, May 16-18, 2000.

[GWG96] Susan Gauch, Guijun Wang, and Mario Gomez. ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines. Journal of Universal Computer Science, Vol. 2, No. 9, Pages 637-649, September 1996. http://www.profusion.com

[Hei95] Sandra Heiler. Semantic Interoperability. Computing Surveys, 27(2), Pages 271-273, June 1995.

[HFAN98] Gerald Huck, Peter Fankhauser, Karl Aberer, and Erich J. Neuhold. JEDI: Extracting and Synthesizing Information from the Web; In: Michael Halper (Ed.), Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems (CoopIS'98), Pages 32-43. New York City, New York, USA, August 20-22, 1998.

[HGMN+97] Joachim Hammer, Hector Garcia-Molina, Svetlozar Nestorov, Ramana Yerneni, Markus M. Breunig, and Vasilis Vassalos. Template-based wrappers in the TSIMMIS system. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Pages 532-535. Tucson, AZ, USA, May 11-15, 1997.

[HHN00] Lieming Huang, Matthias Hemmje, and Erich J. Neuhold. ADMIRE: An Adaptive Data Model for Meta Search Engines. In Computer Networks 33 (1-6), Pages 431-448. Elsevier Science, 2000. /Also in

Proceedings of the 9th International World Wide Web Conference, Amsterdam, The Netherlands, May 15-19, 2000.

[HTHN00] Lieming Huang, Ulrich Thiel, Matthias Hemmje, and Erich J. Neuhold. Constraints-based Query Translation across Heterogeneous Information Sources. In Proceedings of the 2000 International Conference on Information Society in the 21st Century: Emerging Technologies and New Challenges (IS2000), Pages 153-162, Aizu-Wakamatsu City, Fukushima, Japan. November 5-8, 2000.

[HTHN01a] Lieming Huang, Ulrich Thiel, Matthias Hemmje, and Erich J. Neuhold. Adaptively Constructing the Query Interface for Meta-Search Engines. In Proceedings of the 2001 International Conference on Intelligent User Interfaces (ACM IUI 2001). Pages 97-100, Santa Fe, New Mexico USA, January 14-17, 2001. ACM Press.

[HTHN01b] Lieming Huang, Ulrich Thiel, Matthias Hemmje, and Erich J. Neuhold. Distributed Information Search with Adaptive Meta-Search Engines. In Proceedings of the 13th Conference on Advanced Information Systems Engineering (CAiSE'01), Pages 315-329, Interlaken, Switzerland, June 4-8, 2001. LNCS 2068, Springer Verlag.

[HTHN01c] Lieming Huang, Ulrich Thiel, Matthias Hemmje, and Erich J. Neuhold. Adaptive Web Meta-Search Enhanced by Constraint-based Query Constructing and Mapping.. In Proceedings of The Second International Conference on Web-Age Information Management (WAIM 01), Pages 400-407, Xi-an, China, July 9-11, 2001. LNCS 2118, Springer Verlag.

[HTHN02] Lieming Huang, Ulrich Thiel, Matthias Hemmje, and Erich J. Neuhold. Constraints-based Query Translation across Heterogeneous Sources for Distributed Information Retrieval. In book: Enabling Society with Information Technology, Pages 27-37, 2002. ISBN 4-431-70327-6, Tokyo, Springer.

[HK87] R Hull and R. King. Semantic database modeling: Survey, application and research issues. ACM Computing Surveys, Vol. 19, No. 3, Pages 201-260, September 1987.

[HKWY96] Laura M. Hass, Donald Kossmann, Edward L. Wimmers, and Jun Yang. An Optimizer for Heterogeneous Systems with NonStandard

Data and Search Capabilities. Data Engineering Bulletin, Vol. 19, No. 4, Pages 37-44, 1996.

[HKWY97] Laura M. Hass, Donald Kossmann, Edward L. Wimmers, and Jun Yang. Optimizing Queries across Diverse Data Sources. In Proceedings of the 23rd International Conference on Very Large Databases (VLDB97), Pages 276-285, Athens, Greece, August 25-29, 1997.

[HM85] Dennis Heimbigner and Dennis McLeod. A federated architecture for information management. ACM Transactions on Office Information systems, Vol. 3, No. 3, Pages 253-278, July 1985.

[HM93] Joachim Hammer and Dennis McLeod. An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. In International Journal of Intelligent & Cooperative Information Systems, World Scientific, Vol. 2, No. 1, Pages 51-83, 1993.

[HT99] David Hawking and Paul Thistlewaite. Methods for Information Server Selection. ACM Transactions on Information Systems, Vol 17, No 1, Pages 41-76, January 1999.

[Hyl96] Jeremy Hylton. Identifying and Merging Related Bibliographic Records. Master of Engineering thesis, M. I. T. Department of EECS, June, 1996. Also published as LCS Technical Report MIT/LCS/TR-678.

[IC91] IEEE Computer. Special Issue on Heterogeneous Distributed Databases, Vol. 24, No. 12, December 1991.

[IFF+99] Zachary G. Ives, Daniela Florescu, Marc Friedman, Alon Levy, and Daniel S. Weld. An Adaptive Query Execution Engine for Data Integration. In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Pages 299-310, Philadelphia, PA, USA, May 31- June 3, 1999.

[IG02] Panagiotis G. Ipeirotis and Luis Gravano. Distributed Search over the Hidden Web: Hierarchical Database Sampling and Selection. In Proceedings of 28th VLDB Conference, Hong Kong, China, August 20-23, 2002.

[IGS01] Panagiotis G. Ipeirotis, Luis Gravano, and Mehran Sahami. Probe, Count, and Classify: Categorizing Hidden-Web Databases. In

Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, CA USA, May 21-24, 2001.

[ISO93] ISO 8777:1993 Information and Documentation – Commands for Interactive Text Searching, International Organization for Standardization, Geneva, Switzerland, 1993.

[KF97] Bertin Klein and Peter Fankhauser. Error tolerant Document Structure Analysis. IEEE International Conference on the Advances in Digital Libraries (ADL) '97, Pages 116-127, Washington D.C., USA, May 7-9, 1997.

[Kir96] Thomas Kirk. Information Manifold: knowledge based access to information in the WWW. In Proceedings of the WWW5 Workshop on Artificial Intelligence-based tools to help W3 users. Paris, France, May 6, 1996.

[Klei99] Jon M. KleinBerg. Authoritative Sources in a Hyperlinked Environment. Journal of the ACM (JACM), Vol. 46, No. 5, Pages 604-632, September 1999.

[KLSS95] Thomas Kirk, Alon Levy, Yehoshua Sagiv, and Divesh Srivastava. The Information Manifold; In Proceedings of the AAAI-95 Spring Symposium on Information Gathering from Distributed Heterogeneous Environments. Stanford University, California, USA, March 27-29, 1995.

[KM91] Brewster Kahle and A. Medlar. An Information System for Corporate Users: Wide Area Information Servers. Technical Report TMC199, Thinking Machine Corporation, April 1991.

[KM96] Jonghyun Kahng and Dennis McLeod. Dynamic classification ontologies for discovery in cooperative federated databases. In Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96), Pages 26-35, Brussels, Belgium, June 19-21, 1996.

[KMP98] László Kovács, András Micsik, and Balázs Pataki. AQUA: An advanced user interface for the Dienst digital library system. In the eighth DELOS Workshop on User Interfaces for Digital Libraries, Stockholm, Sweden, October 21-23 1998.

[KS94] Vipul Kashyap and Amit P. Sheth. Semantic-based information brokering. In Proceedings of the 3rd International Conference on Information Knowledge Management (CIKM'94), Pages 363-370, Gaithersburg, Maryland, USA, November 29 – December 2, 1994.

[KS95] David Konopnicki and Oded Shmueli. W3QS: A query system for the WWW. Proceedings of 21th International Conference on Very Large Data Bases, Pages 54-65, Zurich, Switzerland, Morgan Kaufmann, September 11-15, 1995.

[KW01] Alfons Kemper and Christian Wiesner. HyperQueries: Dynamic Distributed Query Processing on the Internet. In Proceedings of the 25th VLDB Conference, Pages 551-560, Roma, Italy, September 11-14, 2001.

[LC00] Chen Li and Edward Chang. Query Planning with Limited Source Capabilities. In Proceedings of the 16th International Conference on Data Engineering (ICDE 2000), Pages 401-412, IEEE Computer Society Press, San Diego, California, USA, February 28 – March 3 2000.

[Levy99] Levy, Alon. More on Data Management for XML. May 9th, 1999. Available at http://www.cs.washington.edu/homes/alon/widom-response.html

[LG98a] Steve Lawrence and C. Lee Giles. Inquirus, the NRCI meta search engine. In Proceedings of the Seventh International World Wide Web Conference, Pages 95-105, Brisbane, Australia, Elsevier Science, April 14-18, 1998.

[LG98b] Steve Lawrence and C. Lee Giles. Searching the World Wide Web. Science (280), Pages 98-100, April 1998.

[LHS+95] Ee-Peng Lim, San-Yih Hwang, Jaideep Srivastava, Dave Clements, and M. Ganesh. Myriad: Design and Implementation of a Federated Database Prototype. Software Practice and Experience, Vol. 25, No. 5, Pages 553-562, John Wiley & Sons, May 1995.

[Lib99] Library, University of California, Berkeley. "What are Meta-Search Engines? When to use and not use them?" Available: http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/MetaSearch.html UPDATED January 25,1999.

[Liu98] Jian Liu. Guide to Meta-Search Engines. http://www.indiana.edu/~librcsd/search/meta.html 1998

[LL99] Ee-Peng. Lim and Ying Lu. Harp: a distributed query system for legacy public libraries and structured databases. ACM Transactions on Information Systems, Vol. 17, No. 3, Pages 294-319, July 1999.

[LLD96] Carl Lagoze, C. Lynch, and R. Daniel. The Warwick Framework: Container Architecture for Aggregating Sets of Metadata. 1996.

[LMR90] Witold Litwin, Leo Mark and Nick Roussopoulos. Interoperability of multiple autonomous databases. ACM Computing Surveys, Vol. 22, No. 3, Pages 266-293, September 1990.

[Lovi68] Julie Beth Lovins. Development of a stemming algorithm. Mechanical Translation and Computational Linguistics Vol. 11, No. 1-2, Pages 22-31, 1968.

[LP95] Ling Liu and Calton Pu. The distributed interoperable object model and its application to large-scale interoperable database systems. In Proceedings of the 1995 International Conference on Information and Knowledge Management (CIKM'95), Pages 105-112, Baltimore, Maryland, USA, November 29 - December 2,1995.

[LP97] Ling Liu and Calton Pu. Dynamic Query Processing in DIOM. IEEE Bulletin on Data Engineering, Volume 20, Number 3, Pages 30-37, September 1997.

[LPL96] Ling Liu, Calton Pu, and Yooshin Lee. Adaptive Query Mediation across Heterogeneous Information Sources. In Proceedings of the International Conference on Cooperative Information Systems (CoopIS96), Pages 144-156, Brussels, Belgium, June 10-13, 1996.

[LPVV99] Bertraim Ludäscher, Yannis Papakonstantinou, Pavel Velikhov, and Victor Vianu. View Definition and DTD Inference for XML. Post-ICDT Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats, Jerusalem, 1999.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying Heterogeneneous Information Sources Using Source Descriptions. In Proceedings of the 22nd VLDB Conference, Pages 251-262, Bombay, India, September 3-6, 1996.

[LSDK95] Carl Lagoze, E. Shaw, J. Davis, and D. Krafft. Dienst: Implementation Reference Manual. Cornell Computer Science Technical Report, May 1995.

[LSH95] Ee-Peng Lim, Jaideep Srivastava, San-Yih Hwang. An Algebraic Transformation Framework for Multidatabase Queries. Distributed and Parallel Databases, Vol. 3, No. 3, Pages 273-307, Kluwer Academic Publishers, July 1995.

[LSK95] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. Data Model and Query Evaluation in Global Information Systems. Journal of Intelligent Information Systems. Special Issue on Networked Information Discovery and Retrieval, Vol. 5, No. 2, Pages 121-143, Kluwer Academic Publishers, September 1995.

[LSRH97] Sean Luke, Lee Spector, David Rager, and James Hendler. Ontology-based Web Agents. In Proceedings of the First International Conference on Autonomous Agents (AutonomousAgents97), Pages 59-66, Marina del Rey, CA, USA, W. L. Johnson, ed. New York: Association for Computing Machinery, February 5 - 8, 1997.

[LXLN99] Yong Lin, Jian Xu, Ee-Peng Lim, and Wee-Keong Ng. ZBroker: a query routing broker for Z39.50 databases. In Proceedings of CIKM'99, Pages 202-209, Kansas City, Missouri, USA, November 2-6, 1999.

[MAG+97] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, and Jennifer Widom. Lore: A Database Management System for Semistructured Data. SIGMOD Record, Vol. 26, No. 3, Pages 54-66, September 1997.

[Mar82] Richard S. Marcus. User assistance in bibliographic retrieval networks through a computer intermediary. IEEE Transaction on Systems, Man, and Cybernetics, Vol. 12, No. 2, Pages 116-133, 1982.

[MGKS96] John Mylopoulos, Avigdor Gal, Kostas Kontogiannis, and Martin Stanley. A generic integration architecture for cooperative information systems. In Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96), Pages 208-217, Brussels, Belgium, June 19-21, 1996.

[MI89] Gordon MacAlpine and Peter Ingwersen. Integrated information retrieval in a knowledge worker support system. In Proceedings of the twelfth annual international ACM SIGIR conference on research and development in information retrieval, Pages 48-57, Cambridge, MA, USA, June 25-28, 1989.

[MIR93] Renée J. Miller, Yannis E. Ioannidis, and Raghu Ramakrishnan. The use of information capacity in schema integration and translation. In Prococeedings of the 18th International Conference on Very Large Databases, Pages 120-133, Dublin, Ireland, August 24-27, 1993.

[MK94] Ulla Merz and Roger King. DIRECT: A Query Facility for Multiple Databases. ACM Transactions on Information Systems, Vol. 12, Issue 4, Pages 339-359, October 1994.

[MK99] Atsuyuki Morishima and Hiroyuki Kitagawa. InfoWeaver: dynamic and tailor-made integration of structured documents, Web, and databases. In Proceedings of the fourth ACM conference on Digital libraries, Pages 235-236, Berkeley, CA, USA, August 11 - 14, 1999 .

[MKSI96] Eduardo Mena, Vipul Kashyap, Amit P. Sheth, and Arantza Illarramendi. OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. In Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS '96), Pages 14-25, Brussel, Belgium, June 19-21, 1996.

[MLY+98] Weiyi Meng, King-Lup Liu, Clement T. Yu, Xiaodong Yang, Yuhsi Chang, and Naphtali Rishe. Determining Text Databases to Search in the Internet. In Proceedings of the 24th VLDB Conference, Pages 14-25, New York City, New York, USA, August 24-27, 1998.

[MN99] Andrew McCallum and Kamal Nigam. Text Classification by Bootstrapping with Keywords, EM and Shrinkage. ACL '99 Workshop for Unsupervised Learning in Natural Language Processing. Pages 52-58, June 21, 1999.

[MNRS99] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. A Machine Learning Approach to Building Domain-Specific Search Engines. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Pages 662-667, Stockholm, Sweden, July 31 - August 6, 1999.

[Mor82] Joan M. Morrissey. An Intelligent Terminal for Implementing Relevance Feedback on Large Operational Retrieval Systems. In Proceedings of Research and Development in Information Retrieval (SIGIR'82), Pages 38-50, LNCS 146, Springer-Verlag, Berlin, Germany, May 18-20, 1982.

[MR98] George A. Mihaila and Louiqa Raschid. Locating Data Repositories using XML. W3C Workshop on XML and Querying the Web (QL 1998), Boston, Massachussets, USA, December 3-4, 1998.

[MRC95] Jock D. Mackinlay, Ramana Rao, and Stuart K. Card. An organic user interface for searching citation links, In Proceedings of CHI '95 on Human factors in computing systems, Pages 67-73, Denver, CO, USA, ACM Press, May 7-11, 1995.

[MS98] M. Marchiori and J. Saarela. Query + Metadata + Logic = Metalog. W3C Query Languages meeting in Boston, December 3-4, 1998.

[MW99] Jason McHugh and Jennifer Widom. Query Optimization for XML. In Proceedings of the Twenty-Fifth International Conference on Very Large Data Bases, Pages 315-326, Edinburgh, Scotland, UK. Morgan Kaufmann, September 7-10, 1999.

[MYL02] Weiyi Meng, Clement Yu, and Kling-Lup Liu. Building Efficient and Effective Metasearch Engines. In ACM Computing Surveys, Vol. 34, No. 1, Pages 48-89, March 2002.

[Neg79] A. E. Negus. Development of the Euronet-Diane Common Command Language. In Proceedings of the 3rd International Online Information Meetings, Pages 95-98, 1979.

[NGT98] Hubert Naacke, Georges Gardarin, and Anthony Tomasic. Leveraging Mediator Cost Models with Heterogeneous Data Sources. In Proceedings of the 14th International Conference on Data Engineering (ICDE 1998), Pages 351-360, IEEE Computer Society Press, Orlando, Florida, USA, February 23-27, 1998.

[NR97] Anisoara Nica and Elke A. Rundensteiner. DIIM: a foundation for translating loosely-specified queries into executable plans in large-scale information systems. In Proceedings of the second IFCIS

International Conference on Cooperative Information Systems (CoopIS '97), Pages 213-222, Kiawah Island, South Carolina, USA, June 24-27, 1997.

[Par99] Soyeon Park. User Preferences When Searching Individual and Integrated Full-text Databases. Proceedings of the fourth ACM conference on Digital libraries, Pages 195-203, Berkeley, CA, USA, August 11 - 14, 1999.

[PF98] James Powell and Edward A. Fox. Multilingual Federated Searching Across Heterogeneous Collections. In D-Lib Magazine (ISSN 1082-9873), September 1998.

[PFC+00] Allison L. Powell, James C. French, James P. Callan, Margaret E. Connell, and Charles L. Viles. The impact of Database Selection on Distributed Searching. Proceedings of the 23rd ACM SIGIR Conference on Research and Development in Information Retrieval, Pages 232-239. Athens, Greece. July 24-28, 2000.

[PGMU96] Yannis Papakonstantinou, Hector Garcia-Molina, and Jeffrey Ullman. MedMaker: A Mediation System Based on Declarative Specifications. In Proceedings of the Twelfth International Conference on Data Engineering (ICDE 1996), Pages 132-141, New Orleans, Louisiana, USA, IEEE Computer Society, February 26 - March 1, 1996.

[PGMW95] Yannis Papakonstantinou, Hector Garcia-Molina, Jennifer Widom. Object Exchange Across Heterogeneous Information Sources. In Proceedings of 11th International Conference on Data Engineering, Pages 251-260, Taipei, Taiwan, IEEE Computer Society, March 6-10, 1995.

[PHW+99] Bhujanga Panchapagesan, Joshua Hui, Gio Wiederhold, Stephan Erickson, Lynn Dean, and Antoinette Hempstead. The INEEL Data Integration Mediation System. In Proceedings of the International ICSC Symposium on Advances in Intelligent Data Analysis (AIDA'99), Rochester, New York, USA, June 22-25,1999.

[Port80] Martin. F. Porter. An algorithm for suffix stripping. Program, Vol. 14, No. 3, Pages 130-137, July 1980.

[PTK00] Danny C. C. Poo, Teck-Kang Toh, and Christopher S. G. Khoo. Design and implementation of the E-referencer. Data & Knowledge

Engineering, Vol. 32, No. 2, Pages 199-218, Elsevier publisher, February 2000.

[PW80] S. E. Preece, and M. E. Williams. Software for the searcher's workbench. In Proceedings of the 43rd American Society for Information Science Annual Meeting, volume 17, Pages 403-405, 1980.

[RGM00] Sriram Raghavan and Hector García-Molina. Crawling the hidden Web. Technical Report 2000-36, Computer Science Department, Stanford University, December 2000.

[Rijs79] C. J. van Rijsbergen. Information Retrieval. Butterworths, London, 1979.

[RRM93] Ramana Rao, Daniel M. Russell, and Jock D. Mackinlay. System components for embedded information retrieval from multiple disparate information sources. In Proceedings of the sixth annual ACM symposium on User interface software and technology (UIST 1993), Pages 23-33, Atlanta, GA, USA, November 3 - 5, 1993.

[SAB+95] V. S. Subrahmanian, Sibel Adali, Anne Brink, Ross Emery, James J. Lu, Adil Rajput, Timothy J. Rogers, Robert Ross, Charles Ward. HERMES: Heterogeneous Reasoning and Mediator System. 1995.

[SAD+94] Ming-Chien Shan, Rafi Ahmed, J. Davis, Weimin Du, and William Kent. Pegasus: A heterogeneous information management system. In W. Kim, editor, Modern Database Systems, chapter 32, ACM Press (Addison-Wesley publishers), Reading, MA, USA, 1994.

[Sal83] Gerald Salton. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.

[Sal89] Gerald Salton. Automatic Text Processing: The transformation, analysis, and retrieval of information by computer. Addison-Wesley, 1989.

[Sal91] Gerald Salton. Developments in automatic text retrieval. Science (252), Pages 974-980, 1991.

[SC97] Alan F. Smeaton and Francis Crimmins. Relevance Feedback and Query Expansion for Searchig the Web: A Model for Searching a Digital Library. In: Research and Advanced Technology for Digital Libraries, First European Conference, ECDL'97, Pisa, C. Peters and C.

Thanos (Eds.), Springer Verlag Lecture Notes in Computer Science #1324, pages 99-112, Pisa, Italy, September 1-3, 1997.

[SE95a] Erik W. Selberg and Oren Etzioni. Multi-service search and comparison using the MetaCrawler. In Proceedings of the 4th International World Wide Web Conference, Pages 195-208, Boston, MA, USA, December 1995.

[SE95b] Erik W. Selberg and Oren Etzioni. Experiments with Collaborative Index Enhancement. Technical Report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, USA, January 1997.

[SE00] Atsushi Sugiura and Oren Etzioni. Query Routing for Web Search Engines: Architecture and Experiments. In Proceedings of the 9th International World Wide Web Conference, Pages 417-429, Amsterdam, The Netherlands, May 15-19, 2000.

[SGG96] Terence R. Smith, Steven Geffner, and Jonathan Gottsegen. A general framework for the meta-information and catalogs in digital libraries. In Proceedings of the First IEEE Metadata Conference, IEEE Computer Society Press, Silver Spring, Maryland, USA, April 16-18, 1996.

[SGMT97] Adelheit Stein, Jon Atle Gulla, Adrian Müller, and Ulrich Thiel. Conversational Interaction for Semantic Access to Multimedia Information. In: Mark T. Maybury (Ed). Intelligent Multimedia Information Retrieval. Pages 399-421, Menlo Park, CA, AAAI / The MIT Press, 1997.

[SGT97] Adelheit Stein, Jon Atle Gulla, and Ulrich Thiel. Making Sense of Users' Mouse Clicks: Abductive Reasoning and Conversational Dialogue Modeling. In Anthony Jameson; Cécile Paris & Carlo Tasso (Eds.), User Modeling: Proceedings of the Sixth International Conference (UM97), Pages 89-100, Chia Laguna, Sardinia, Italy, June 2-5, 1997.

[SHMM99] Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz. Analysis of a Very Large Web Search Engine Query Log. ACM SIGIR Forum, Vol. 33, No. 1, Fall 1999.

[SL90] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM

Transaction on Database System, Vol. 22, No. 3, Pages 183-236, September 1990.

[SV98] Subbu N. Subramanian and Shivakumar Venkatataman. Cost-Based Optimization of Decision Support Queries using Transient Views. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data. Pages 319-330, Seattle, Washington, USA, June 2-4, 1998.

[Tol82] D. E. Toliver. OL'SAM: An intelligent front-end for bibliographic information retrieval. Information. Technology and Libraries, Vol. 1, No. 4, Pages 317-326, December 1982.

[TRÖH99] Mary Tork Roth, Fatma Özcan, and Laura M. Haas. Cost models do matter: Providing cost information for diverse data sources in a federated system. In Proceedings of the 25th VLDB Conference, Pages 599-610, Edinburgh, Scotland, September 7-10, 1999.

[TRS97] Mary Tork Roth and Peter M. Schwarz. Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources. In Proceedings of the 23rd VLDB Conference, Pages 266-275, Athens, Greece, August 25-29, 1997.

[TRV96] Anthony Tomasic, Louiqa Raschid, and Patrick Valduriez. Scaling Heterogeneous Databases and the Design of Disco. In Proceedings of the 16th International Conference on Distributed Computing Systems, Pages 449-457, Hong Kong, May 27-30, 1996.

[TRV97] Anthony Tomasic, Louiqa Raschid, and Patrick Valduriez. A Data Model and Query Processing Techniques for Scaling Access to Distributed Heterogeneous Databases in Disco. Invited paper in the IEEE Transactions on Computers, special issue on Distributed Computing Systems, 1997.

[TTC+90] Gomer Thomas, Glenn R. Thompson, Chin-Wan Chung, Edward Barkmeyer, Fred Carter, Marjorie Templeton, Stephen Fox, and Berl Hartman. Heterogeneous distributed database systems for production use. ACM Computing Surveys, Vol. 22, No. 3, Pages 236-266, September 1990.

[UFA98] Tolga Urhan, Michael J. Franklin, and Laurent Amsaleg. Cost-based query scrambling for initial delays. In Proceedings of the 1998 ACM

SIGMOD International Conference on Management of Data. Pages 130-141, Seattle, WA, USA, June 2-4, 1998.

[Ull97] Jeffrey D. Ullman. Information Integration Using Logical Views. In Proceedings of the sixth International Conference on Database Theory (ICDT'97), Pages 19-40, Delphi, Greece, January 8-10, 1997.

[VGJL94] Ellen M. Voorhees, Narendra Kumar Gupta, Ben Johnson-Laird. The Collection Fusion Problem. In Proceedings of the Third Text Retrieval Conference (TREC-3), Gaithersburg, Maryland, USA, 1994.

[VGJL95] Ellen M. Voorhees, Narendra Kumar Gupta, Ben Johnson-Laird. Learning Collection Fusion Strategies. In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pages 172-179, Seattle, Washington, USA, July 9-13, 1995.

[VT97] Ellen M. Voorhees and Richard M. Tong. Multiple Search Engines in Database Merging. In Proceedings of the 2nd ACM International Conference on Digital Libraries, Pages 93-102, Philadelphia, PA, USA, July 25-28, 1997.

[VP97] Vasilis Vassalos and Yannis Papakonstantinou. Describing and Using Query Capabilities of Heterogeneous Sources. In Proceedings of the 23rd VLDB Conference, Pages 256-265, Athens, Greece, August 25-29, 1997.

[VZ98] Shivakumar Venkataraman and Tian Zhang. Heterogeneous Database Query Optimization in DB2 Universal DataJoiner. In Proceedings of the 24th VLDB Conference, Pages 685-689, New York City, New York, USA, August 24-27, 1998.

[WHC+96] Darrell Woelk, Michael N. Huhns, Philip Cannata, Nigel Jacobs, Tomasz Ksiezyk, Greg Lavender, Greg Meredith, Kayliang Ong, Wei-Min Shen, Munindar Singh, and Christine Tomlinson. Carnot Prototype. In Omran Bukhres and Ahmed K. Elmagarmid (eds.) Object-Oriented Multidatabase Systems, Prentice Hall International, chapter 18, Pages 621-651. 1996.

[Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. IEEE Computer, Vol. 25, No. 3, Pages 38-49, March, 1992.

[Wil86] M. E. Williams. Transparent information systems through gateways, front ends, intermediaries, and interfaces. Journal of the American Society for Information Science, Vol. 37, No. 4, Pages 204-214, July 1986.

[WMB94] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. Managing Gigabytes: Compressing and Indexing Documents and Images, Von Nostrand reinhold, New York, 1994.

[WMYL01] Zonghuan Wu, Weiyi Meng, Clement Yu, and Zhuogang Li. Towards a highly scalable and effective metasearch engine. In Proceedings of the Tenth International World Wide Web Conference (WWW01). Pages 386-395, Hong Kong, China, May 1-5, 2001.

[WZRW87] S. K. Michael Wong, Wojciech Ziarko, Vijay V. Raghavan, and P. C. N. Wong. On Modeling of Information Retrieval Concepts in Vector Spaces. ACM Transactions on Database Systems, Vol. 12, No. 2, Pages 299-321, June 1987.

[YL97] Budi Yuwono and Dik Lun Lee. Server ranking for distributed text retrieval systems on the Internet. In Proceedings of the Fifth International Conference on Database systems for Advanced Applications (DASFAA 1997), Pages 41-50, Melbourne, Australia, April 1-4, 1997.

[YLGMU99] Ramana Yerneni, Chen Li, Hector Garcia-Molina, and Jeffrey Ullman. Computing Capabilities of Mediators. In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, Pages 443-454, Philadelphia, PA, USA, May 31 - June 3 1999.

[YMWL01] Clement Yu, Weiyi Meng, Wensheng Wu, and Kling-Lup Liu. Efficient and Effective Metasearch for Text Databases Incorporating Linkages among Documents. In Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data. Pages 187-198, Santa Barbara, California, USA, May 21-24, 2001.

[Z93] Z39.58-1992 Common Command Language for Online Interactive Information Retrieval. NISO Press, Bethesda, Maryland, USA, 1993.

[Z95] Z39.50 Maintenance Agency. Information Retrieval (Z39.50): Application Service Definition and Protocol Specification. ftp://ftp.loc.gov/pub/z3950/official/. 1995.

[ZG00] Xiaolan Zhu and Susan Gauch. Incorporating Quality Metrics in Centralized/Distributed Information Retrieval on the World Wide Web. In Proceedings of the 23rd ACM SIGIR Conference on Research and Development in Information Retrieval, Pages 288-295. Athens, Greece. July 24-28, 2000.

[ZRV+02] Vladimir Zadorozhny, Louiqa Raschid, Maria Esther Vidal, Tolga Urhan and Laura Bright. Efficient Evaluation of Queries in a Mediator for WebSources. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data. Pages 85-96, Madison, Wisconsin, USA, June 3-6, 2002.

[ZRZB01] Vladimir Zadorozhny, Louiqa Raschid, Tao Zhan, and Laura Bright. Validating a cost model for wide area applications. In Proceedings of the 9th International Conference on Cooperating Information Systems, International Conference (CoopIS 2001), Pages 371-385, Trento, Italy, September 5-7, 2001.

[ZSB86] S. Zinn, M. Sellers, and D. Bohli. OCLC's intelligent gateway service: Online information access for libraries. Library High Technologies, Vol. 4, No. 3, Pages 25-29, 1986.

List of Acronyms

CSC	Classification Selection Control
CSCItem	Classification Selection Control Item
CSCSet	Classification Selection Control Set
CTRLSet	Control Set
FM	Field Modifier
FMSet	Field Modifier Set
L	Logical Operator Control
LSet	Logical Operator Control Set
PRI	Priority Boolean Expression
Q	Query expression
QIC	Query Input Control
RDC	Result Display Control
RDCItem	Result Display Control Item
RDCSet	Result Display Control Set
SEQ	Sequential Boolean Expression
T	Term
TQ	Term Qualifier
TQSet	Term Qualifier Set
TSet	Term Set
TreeCSC	Tree-structured Classification Selection Control
VSQ	Vector-Space Query expression

Curriculum Vitae

Lieming HUANG

Education:

Sep. 1998 to Sep.2003
Ph.D. (Dr. -Ing.) in Department of Computer Science (Informatik), Darmstadt University of Technology, Darmstadt, Germany

Sep. 1994 to Aug. 1997
Master of Engineering in Computer Science (M. Eng), Chinese Academy of Sciences, Beijing, P.R.China
Thesis: Intelligent User Interface Support for DBMS

Oct. 1989 to Jul. 1994
Bachelor of Science in Computer Science (B.Sc), Dept. Computer Science & Technoloy, Peking University, Beijing, P.R.China

Sep. 1986 to Jul. 1989
Liuzhou Senior Middle School, Liuzhou, Guangxi, P.R.China

Sep. 1983 to Jul. 1986
Junior Middle School of Liuzhou Steel&Iron Corporation, Liuzhou, Guangxi, P.R.China

Sep. 1978 to Jul. 1983
Primary School of Liuzhou Steel&Iron Corporation, Liuzhou, Guangxi, P.R.China

Published Papers during PhD research:

Lieming Huang, Ulrich Thiel, Matthias Hemmje, Erich J. Neuhold. (2002).
Constraints-based Query Translation across Heterogeneous Sources for Distributed Information Retrieval. In book: (Q. Jin, J. Li, N. Zhang, J. Cheng, C. Yu, S. Noguchi, Eds.) Enabling Society with Information Technology, Pages 27-37, 2002. ISBN 4-431-70327-6, Springer Verlag Tokyo.

Lieming Huang, Ulrich Thiel, Matthias Hemmje, Erich J. Neuhold. (2001)
Adaptive Web Meta-Search Enhanced by Constraints-based Query Constructing and Mapping. In: (X. Wang, G. Yu, H. Lu, Eds.) Proceedings of the Second International Conference on Web-Age Information Management (WAIM 01), Pages 400-407, Xi'an, China, July 9-11, 2001. LNCS 2118, ISSN 0302-9743, ISBN 3-540-42298-6, Springer-Verlag Berlin Heidelberg.

Lieming Huang, Ulrich Thiel, Matthias Hemmje, Erich J. Neuhold. (2001)
Distributed Information Search with Adaptive Meta-Search Engines. In: (K. R. Dittrich, A. Geppert, M. C. Norrie, Eds.) Proceedings of The 13th Conference on Advanced

Information Systems Engineering (CAiSE'01), Pages 315-329, Interlaken, Switzerland, June 4-8, 2001. LNCS 2068, ISSN 0302-9743, ISBN 3-540-42215-3, Springer-Verlag Berlin Heidelberg.

Lieming Huang, Ulrich Thiel, Matthias Hemmje, Erich J. Neuhold. (2001)
Adaptively Constructing the Query Interface for Meta-Search Engines. In: (J. C. Lester, Ed.) Proceedings of the 2001 ACM International Conference on Intelligent User Interfaces (ACM IUI 2001), Pages 97-100, Santa Fe, New Mexico USA - January 14-17, 2001. ISBN 1-58113-325-1, ACM Press.

Lieming Huang, Ulrich Thiel, Matthias Hemmje, Erich J. Neuhold. (2000)
Constraints-based Query Translation across Heterogeneous Information Sources. In: (Q. Jin, J. Li, N. Zhang, J. Cheng, C. Yu, S. Noguchi, Eds.) Proceedings of the 2000 International Conference on Information Society in the 21st Century: Emerging Technologies and New Challenges (IS2000). Pages 153-162. Aizu-Wakamatsu City, Fukushima, Japan. November 5-8, 2000.

Lieming Huang, Matthias Hemmje, Erich J. Neuhold. (2000)
ADMIRE: An Adaptive Data Model for Meta Search Engines. Computer Network 33 (1-6), Pages 431-448. 2000. ISSN 1389-1286, Elsevier Science.

Lieming Huang, Matthias Hemmje, Erich J. Neuhold. (2000)
ADMIRE: An Adaptive Data Model for Meta Search Engines. In Proceedings of the 9th International World Wide Web Conference (WWW9), Pages 431-448. Amsterdam, The Netherlands, May 15-19, 2000. ISBN 0-444-50515-6, Elsevier Science.

Working Experiences:

Sep. 1998 to 2004
Computer scientist, Fraunhofer-IPSI, Darmstadt, Germany

Aug. 1997 to Aug. 1998
Research assistant in Computer Science, Chinese Academy of Sciences, Beijing, P.R.China