Design and Implementation of the E-Referencer

Danny C. C. Poo
Teck-Kang Toh

School of Computing
National University of Singapore
Lower Kent Ridge Road
Singapore 119260
dpoo@comp.nus.edu.sg

Christopher S. G. Khoo

Centre for Advanced Information Systems
School of Applied Science
Nanyang Technological University
Singapore 639798
assgkhoo@ntu.edu.sg

Design and Implementation of the E-Referencer

Abstract. An expert system Web interface to online catalogs called E-Referencer is being developed. An initial prototype has been implemented. The interface has a repertoire of initial search strategies and reformulation strategies that it selects and implements to help users retrieve relevant records. It uses the Z39.50 protocol to access library systems on the Internet. This paper describes the design and implementation of the E-Referencer. A preliminary evaluation of the strategies is also presented.

1. Introduction

E-Referencer is a web-based interface to online catalogs that is being developed as a tool to test and experiment on various search strategies that are aimed at helping online catalog users in their searches. The design of the system makes use of the expert system technology in the selection of search strategies. Through the conduct of experiments and with some modifications and fine-tuning, it is hope that the E-Referencer can be used as an effective searching tool for all online catalog users.

The E-Referencer uses the Z39.50 Information Retrieval protocol [1] to communicate with the various library systems, and the Java Expert System Shell (JESS) [2] to implement the knowledge base of the system. At the present moment, the knowledge base of the E-Referencer consists of:

1. a *conceptual knowledge base* that maps free-text keywords to concepts represented by the Library of Congress (LC) subject headings
2. *search strategies* coded in the system, including
   - *initial search strategies*, used to convert the user's natural language query to an appropriate Boolean search statement
   - *reformulation strategies*, used for refining a search based on the results of the previous search statement
3. *rules* for selecting an appropriate search strategy.

The E-Referencer processes a user's natural language query, selects a suitable search strategy and formulates an appropriate search statement for the library system. Based on the user's relevance feedback on the search result, it further selects a strategy for reformulating the search. The process goes on until the user is satisfied with the final result.

This paper begins by discussing the state of implementation of current search systems as implemented in Online Public Access Catalogues System (OPACS). The discussion then proceeds to illustrate the improvement in search results obtained by using an initial prototype of the E-Referencer. The design and implementation of the E-Referencer covers the remaining sections of the paper.

2. Experiences in Online Catalog Searches

2.1 User Difficulties in searching OPACS

Traditional text-based boolean search systems as incorporated in the design of OPACS are difficult to use. In the 1984 Online Catalog Evaluation Projects sponsored by the Council of Library Resources, Markey found that users have the following problems when performing subject searches of online catalogues [6] :

- Users have problems matching their terms with those indexed in the online catalogue.
- They have difficulty identifying terms broader or narrower than their topic of interest.
- They do not know how to increase the search results when too little or nothing is retrieved.
- They do not know how to reduce the search results when too much is retrieved.
- They lack understanding of the printed LCSH (Library of Congress Subject Heading)

Hildreth also pointed out that "conventional informational retrieval systems place the burden on the user to reformulate and re-enter searches until satisfactory results are obtained." [7] Indeed the boolean search system, which was the conventional informational retrieval system used then, requires too much knowledge from their users. The new Web-based library OPAC search systems today are not much different from those in the eighties in terms of functionality. Borgman indicated that most of the improvements to online catalogues in recent years were in surface features rather than in the core functionality. Also, online catalogues "were designed for highly skilled searchers, usually librarians, who used them frequently, not for novices or for end-users doing their own searching." [8].

Other studies conducted by Cousins [3], Dalrymple [4], Ensor [5], Lancaster [6] have found deficiencies in the present-day online catalogue systems and identified the problems users have. For example, Cousins [3] analyzed the types of subject queries that users brought to two online catalog systems. She found that many subject queries were not expressed at the level of specificity that was appropriate or suitable for searching the system. She concluded that online catalogue systems should provide more information about the document content (e.g. content pages), facilities for browsing the thesaurus and the classification scheme, facilities for browsing records arranged by class number, ranked display of search results, help with query formulation, and relevance feedback.

2.2 Recent Developments in Improving the Search System

To help the users search more effectively, recent research efforts have concentrated on improving the search system. The more notable developments are:

- Introduction of 'best match' or statistically-based search systems that incorporate numerically-based algorithms for estimating the relevance of documents to the users' query.
- Use of knowledge-based search systems that encode expert knowledge to provide advice or assistance to users for searching.
- Development of the Z39.50 Standard for Information Retrieval that provides common access to multiple online catalogues.

2.2.1 'Best Match' Search Systems

There are many variations of 'best match' or statistically-based search systems. They support a mixture of features like document ranking, relevance feedback and query expansion. These systems use either the vector-based information retrieval model or the probabilistic model to re-index the collection of documents to support more effective retrieval. The user is allowed to enter his search query in natural language. The search query is then treated as a list of unstructured keywords. Stop-words such as '`is`', '`a`', and '`the`', which are not very useful in the search process are removed. The remaining content-bearing words are stemmed to a common form. Inverse frequency weights are then calculated for each of the remaining query stems; whereby the greatest weights are then assigned to those terms that occur least frequently in the document collection. The weight of each document in the collection is then calculated by taking the sum of the weights of the common terms that occur in both the query and the document. Subsequently, the documents are sorted base on their calculated weights. This process is known as *document ranking*.

In order to determine the relevancy of the documents, a certain number of top ranking documents are displayed to the user. This process is known as *relevance feedback*. Based on the user's feedback, terms are extracted from the relevant documents; the selected terms are then weighted and ranked.

A certain number of the top ranking terms could be automatically added to the original query. Alternatively, they can be displayed to the user and the latter can decide if the new terms are to be included in the original query. This process is known as *query expansion*.

Examples of best match systems include the Interactive System for Teaching Retrieval (INSTRUCT) developed in the University of Sheffield [11], the Okapi System by the City University in London [12] and almost all the current web search engines.

There are advantages in using such systems. First, the user does not need to compose their queries using logical operators. Second, the ranking of documents means that the user is more likely to find the required document at the top of the list. Third, the use of relevance feedback information to refine the query, by automatic or semi-automatic query expansion provides a useful mechanism for the user to clarify his query and locate the required information quickly.

However, these are also problems with such systems. Firstly, there is a need to re-index the existing database and this can be very costly and time-consuming. This may explain why very few library OPAC boolean search systems are 'best match' systems. Secondly, a large amount of records are usually returned in each search. This is evident in the searches carried out on the web. It is unlikely for users to browse through all the returned records. A small number of the records is finally seen. Lastly, such numerically-based systems do not utilize any extra information or domain knowledge present that may be used to help users in their search.

2.2.2 Knowledge-based Search Systems

Another approach is to make use of techniques from research in expert systems. An intermediary system that removes the need for users to have knowledge of boolean search can be provided. Expert systems encode the knowledge of a human expert in the domain of interest and the strategies the expert uses for reasoning. In the field of information retrieval, the expert is probably the reference librarian and his/her expert knowledge lies in his/her ability to convert natural language queries into boolean queries, and refining and reformulating the query according to his/her knowledge of the subject area being searched.

A number of such systems based on expert system techniques like production rules, semantic nets and frames have been developed over the years; for example:

- Gauch's Query Reformulation system uses production rules to reformulate the query by manipulating the boolean operators, or by adding related terms, or by replacing terms with broader or narrower terms from a thesaurus [13].
- PLEXUS, an expert referral system that uses a frame-based representation of topics in the domain of gardening to map words to semantic primitives. It incorporates some search strategies expressed as production rules, and builds a temporary user profile based on the users' gardening experience, knowledge and location. This profile is used to set up the level of help for the user [14].
- RUBRIC (Rule-Based Retrieval Information by Computer) uses production rules to define a hierarchy of retrieval concepts. Domain knowledge is modeled as a collection of concepts. Each concept contains a description, the relationship between the concept with other concepts, and the rules that describe the patterns of text that should be present to support the use of the retrieval concept. In this system, the user is required to provide to the system domain specific concepts [15].
- Drabenstott and her colleagues developed a prototype online catalogue that uses search trees or decision trees to represent how experienced librarians select a search strategy and formulate a search statement. The decision tree is represented as a flowchart [16-18].

The main advantage of using the knowledge-based approach is in the ability to reuse the existing database index. By building on top of existing infrastructure, libraries are more likely to adopt the knowledge-based solution to augment their existing OPAC boolean search systems. Furthermore, the classification systems in library catalogues such as the Dewey Classification System and the Library of Congress Subject Classification System, are already well-defined. Such structured domain information and knowledge can thus be incorporated in any system to help users in their searches.

2.2.3 Z39.50 Information Retrieval Protocol

For many years, users have been limited to performing searches at the library premises. Users have to also make use of the OPAC search system as provided by each library in searching the catalogs. Although most systems are boolean search systems offering similar search capabilities, users still need to learn each of the different search interfaces in order to be proficient in accessing information from the online catalogs.

In order to overcome this compatibility issue, the Z39.50 Information Retrieval Protocol was developed to standardize the informational retrieval process. Commercial products and freeware search systems that implement the Z39.50 protocol are now available. For a list of Z39.50 products and freeware, refer to http://lcweb.loc.gov/z3950/agency/projects/software.html.

We have evaluated a few of these Z39.50 compliant search systems, namely:

- Sitesearch WebZ software from Online Computer Library Center (OCLC)
  http://www.oclc.org/oclc/sitesearch/components.htm
- Chalmers from Chalmers Library, Chalmers University of Technology
  http://www.lib.chalmers.se/prov/Z3950/gateway.html
- Willow from University of Washington
  http://www.washington.edu/willow
- WWW/Z39.50 Gateway from Library of Congress
  http://lcweb.loc.gov/z3950/gateway.html

Most of all these Z39.50 compliant search systems provide access to multiple databases but they only support boolean logic queries. Users are still burdened with the responsibility of query formulation and reformulation. Susannah noted that “Almost all literature on Z39.50 and its implementation focuses on the issues related to the implementor and the development of the standard. In the ten years since work on Z39.50 began, little attention has been given to the end user, the one who is supposed to ultimately benefit from the implementation of the standard.” [19].

2.3 Our Approach

A study conducted by Robertson and his colleagues found that there is very little overall difference in performance between the ‘best match’ search system, INSTRUCT, and the knowledge-based system, TomeSearcher [20].

In order to help current users of library online catalogs search more effectively, the knowledge-based approach is currently the most suitable and feasible approach. The approach as taken in the paper is to build an intermediary system on top of the existing facilities as provided by OPACS. In this way, we can reuse the existing infrastructure to reduce cost, and to take advantage of the domain knowledge present in the structure of online catalog classification systems and the expertise of the librarians in searching catalogs to enhance the search session of the users.

The task of interfacing is made simpler with the widespread acceptance and use of the Z39.50 Information Retrieval Protocol. Our expert intermediary system, the E-Referencer, encompasses the expert knowledge of reference librarians. Librarians are trained in formulating queries in boolean logic and reformulating them to obtain sufficient relevant records. Such formulation and reformulation strategies are domain-independent and can be applied to any boolean search systems.

The E-Referencer also makes use of the domain knowledge present in the structure of classification scheme of the online catalogs to assist users. In a library OPAC system, the documents and records are classified based on information such as subject heading and call number. This information is fully utilized in helping users maximize their searches.

2.4 Evaluation of E-Referencer and Results

The system, E-Referencer version 2.0, has been implemented and is accessible at the URL *http://revelation.comp.nus.edu.sg/ERef2.0/*.

An evaluation of the system was carried out using an earlier version of the E-Referencer on 12 queries that were selected from among those submitted by the university staff and students for literature searches. The queries were selected to cover a wide range of subjects. A complete list and description of the 12 queries is given below:

Query No.	Topic
A96-7	Digital library projects
A96-14	Cognitive models of categorization
A97-16	Internet commerce
B97-1	Making a framework for surveying service quality in academic libraries
B97-3	Looking to do a comprehensive literature review of the Sapir-Whorf Hypothesis
D96-2	Software project management
D96-16	Decision under uncertainty
D97-1	Thermal conductivity of I.C. Packaging
D97-2	Fault-Tolerant Multiprocessor Topologies
D97-4	Face recognition
D97-7	A study on computer literacy and use by teachers in primary schools
N97-13	Expert systems in library reference service

The Nanyang Technological University (NTU) library system in Singapore was used in the evaluation. Searches were performed by entering the query topics into the traditional search system provided by NTU's library at *http://www.ntu.edu.sg/library/opacs.html*, and also on the E-Referencer prototype that we developed. The same set of queries was also given to an expert librarian who performed her own search and reformulation using the NTU library search system.

To illustrate the process of searching by the three systems, we shall use one of the queries above: A96-7 *"Digital library projects"*.

a. <u>NTU Library Search System</u>

No record was returned when the query string was entered in this search system.

b. <u>E-Referencer</u>

The same query string was then entered into the E-Referencer; Initial Search Strategy 1 was carried out as follows:

- Stop-words were replaced with ANDs

    "Digital library projects"

- The words in between the AND and OR operators were assumed to be adjacent

    "Digital library projects"

- Individual words were stemmed and truncation signs added

    "Digit? librar? project?"

- The formulated query string *"Digit? librar? project?"* was sent to the NTU library search system.

No record was retrieved and Broadening Strategy 1 activated:

- The operator AND was inserted between all adjacent words

    "Digit? AND librar? AND project?"

- Once again, the NTU library search system was sent the query *"Digit? AND librar?AND project?"*

The search yielded four records, they include:

1. Conversion of the microfilm to digital imagery: a demonstration project: performance report on the production conversion phase of Project Open Book / by Paul Conway, principal investigator.
2. Digital library visualization tool / by Yee Mun Sung.
3. Library development for mixed analog-digital circuit simulation / submitted by Ng Kian Ching, Ng Meng Hui.
4. Electronic services in academic libraries : ALA survey report / by Mary Jo Lynch, project director.

c. Expert Librarian

The expert librarian was able to retrieve 6 records from the given query string.

Searches using other query string from the above topics were also carried out. The results of the searches were given to two judges, who were asked to indicate whether the records retrieved were relevant, marginally relevant or not relevant. For this evaluation, records that were judged to be marginally relevant were considered to be non-relevant. The precision measure (proportion of records retrieved that are relevant) was calculated for the first 20 records displayed. (The expert system currently displays only 20 records to the user for relevance judgment.) The mean precision for the 2 sets of relevance judgments was then calculated for each query. The consolidated results are given in Table 1.

Query No.	Search by NTU Search System			Search by E-Referencer			Search by Librarian		
	No. Displayed	No. Relevant	Precision	No. Displayed	No. Relevant	Precision	No. Displayed	No. Relevant	Precision
A96-7	0	0	0.00	4	0.5	0.13	17	6	0.35
A96-14	0	0	0.00	20	1	0.05	3	2	0.67
A97-16	4	1	0.25	4	1	0.25	20	705	0.38
B97-1	0	0	0.00	20	3	0.15	13	7	0.54
B97-3	0	0	0.00	2	0	0.00	4	3.5	0.88
D96-2	11	10.5	0.95	11	10.5	0.95	20	14	0.70
D96-16	0	0	0.00	20	9	0.45	20	9	0.45
D97-1	0	0	0.00	3	1.5	0.50	20	5	0.25
D97-2	0	0	0.00	5	3.5	0.70	6	6	1.00
D97-4	0	0	0.00	8	2	0.25	15	5	0.33
D97-7	0	0	0.00	19	0	0.00	14	2.5	0.18
N97-13	0	0	0.00	4	3.5	0.88	5	4	0.80
Average	**1.25**	**1.0**	**0.10**	**10**	**3.0**	**0.36**	**13.1**	**6.0**	**0.54**

Table 1: Comparison of Search Results

Note:

1. The figures given for No. Relevant and Precision are the average for 2 sets of relevance judgments by 2 persons.
2. The evaluation is based on the first 20 records retrieved. E-Referencer currently displays only 20 records for relevance judgment.

Clearly, Table 1 shows that the E-Referencer performed much better than the traditional library search system. The reformulation strategies used by the E-Referencer have helped the user retrieve more relevant records.

However, the result shows that the E-Referencer did not perform as well as an expert librarian. While this may be true from Table 1, it must be pointed out that the evaluation is skewed towards the expert librarian. This is because the librarian was able to execute several search statements, and continually refine the search after examining the records retrieved in earlier search formulations. While the above results show the final search set as obtained by the librarian, it is only the first non-null set retrieved that is shown for the E-Referencer.

The evaluation suggests that the E-Referencer can be an efficient tool for helping online catalog users to search better. The strategies as used in the prototype still need to be refined further. Current work is aimed at realizing this.

A complete description of the search strategies can be found in [16]. In the rest of the paper, we shall focus on the design and implementation of the E-Referencer.

3. Conceptual Design

For many years, when library users have problems finding what they need using the search systems in the libraries, they would approach the reference librarians. The latter are people who are more knowledgeable and proficient with the catalog search system. In general, librarians would ask what the user is looking for, clarifying the subject or topic when necessary before constructing a query in boolean logic to search the catalog. If too little records are retrieved, the librarian would either change some of the boolean operators in the query, like changing the AND operator to OR to get more records, or try using similar keywords or broader subject headings in the reformulation. Similarly if too many records are retrieved, the librarian may try modifying the boolean operators or use narrower subject headings to reduce the search results. This process goes on until the user is satisfied with the search result.

Basically, there are two types of knowledge present in a search session:

1. Domain knowledge – Classification information of records in OPACS such as Subject Headings, and Dewey and Library of Congress Call Numbers. These information can be used to help users clarify their search topic and locate the relevant documents. The hierarchy present in the various classification schemes can also be used to broaden or narrow a search.
2. Domain-independent knowledge – Search strategies that librarians used to formulate the user's original query to the format expected by the OPAC (boolean logic), and reformulation of the query by modifying the boolean logic operators to get more or less records.

In the design of the E-Referencer, we have decided to incorporate these two types of knowledge into the system. A conceptual knowledge base of domain knowledge has been incorporated into the E-Referencer to map keywords to concepts represented in the subject headings. The domain-independent knowledge of formulation and reformulation rules has been implemented as search strategies in the E-Referencer. For a complete description of search strategies used in the E-Referencer, the reader is referred to an earlier published paper [21].

4. System Design and Implementation

Having illustrated the potential of the E-Referencer, we will now describe the design and implementation of the E-Referencer.

4.1 Design Approach and Considerations

The approach we have adopted in developing our system is that of rapid prototyping and incremental development. We first implemented an initial prototype using simple-minded strategies specified by an experienced librarian. We then carry out experiments to evaluate the system and compare its performance with that of experienced librarians. From this, we identify the areas the system is deficient and how it can be improved.

The prototype system had been designed to study the reasons why experienced librarians are more superior in their searches than ordinary users, and how expert search systems can be designed to match what experienced librarians can do. This approach of rapid prototyping is necessary because at the start of the project, we do not know which strategy is the most effective one to use. A detailed planning and design approach is thus not feasible. This cycle of incremental development, testing and redevelopment has allowed us to add new features and refine the system gradually.

The increasingly popular Z39.50 Informational Retrieval protocol was used to provide a common interface to multiple online catalogs.

Search strategies as used by librarians have been incorporated into the knowledge base of the prototype system using JESS.

E-Referencer uses a three-tier design architecture consisting of a client, proxy and server. The client handles user interaction, the server (a Z39.50 server) contains the data and search strategies, and a proxy sits between the client and server. This approach is necessary because the subject heading database required by the conceptual knowledge base is huge, around 300 megabyte. It is not feasible to send this large database across the network to every client in order to extract subject headings of usually a few keywords (at most ten). In our design, the proxy houses the database and handles all the processing required on the conceptual knowledge base. Keywords are submitted by clients to the proxy, which will then search the subject heading database and return the relevant subject headings to the clients. The traffic generated in this case is greatly reduced since only the keywords and the resulting subject headings are returned. Another advantage of this approach is that we can log all the activities carried out by the different clients; the log information can be useful for further analysis.

3.2 Design of the E-Referencer

The design of the E-Referencer is shown in Figure 1. It consists of the following modules:

Client Modules

a. The Graphical User Interface (GUI) Module handles the interaction between the user and E-Referencer.
b. The Network Interface Module communicates with the E-Referencer proxy.
c. The Expression Module provides functions for manipulating a search expression.
d. The Control Module is the heart of the expert system. It controls and calls the various functions of the system and has the following components:
    - A Knowledge Base of search strategies
    - A Fact Base which stores the intermediate search results and information needed to select the next search strategy.
    - An Explanation Facility for explaining why and how certain strategies were chosen.
e. The Knowledge Module contains wrapper functions for integrating the expert system script of the Control Module with the other modules of E-Referencer.

Proxy Modules

a. The Proxy Controller Module accepts new connections from clients and activates the appropriate modules to handle the various clients' requests.
b. The Keyword-Subject Association Module provides a list of subject headings that associates with the keywords users specify in their query. The subject headings are used to augment the user's original query to perform a more accurate search.

c. The OCLC Z39.50 Client API provides functionality for connecting to, searching and retrieving information from the various library systems that support the Z39.50 protocol.

d. The Z39.50 Interface Module provides a clean interface to the OCLC Z39.50 Client API. It isolates the rest of the system from changes to the OCLC Z39.50 Client API.

3.3 Client Modules Design and Implementation

a. Graphical User Interface Module

The widespread use of graphical-based operating systems like Windows, OS/2 and X-Windows have greatly increased the demand for programs written with graphical user interfaces. The proper usage of graphical user interface provides a very simple and friendly way for the user to interact with the system. The mouse pointer allows for easy manipulation of the system and the use of graphical items like buttons and scroll windows allows the system to present its information to the user clearly and effectively. Thus, the E-Referencer, which will eventually be used by ordinary users, has to support a graphical user interface.

In addition, we hope to make the E-Referencer easily accessible to all online catalog users, and thus a Web-based graphical interface is required in the design of the E-Referencer.

We have also designed the user interface to be simple, so that it is easy to use. The interface contains only one keyboard input area for the user to enter the query string, so that users will not need to spent too much time learning how to use the system. Limited information on the search results is displayed; the information includes title, author and publisher information. The records retrieved are also arranged in a list for easy browsing.

Since the E-Referencer is also used as an experimental tool to help us refine and test our search strategies, we have also included a server and strategy option in the design. The server option allows us to search different Z39.50 servers, while the strategy option allows us to use different strategies for reformulation purposes.

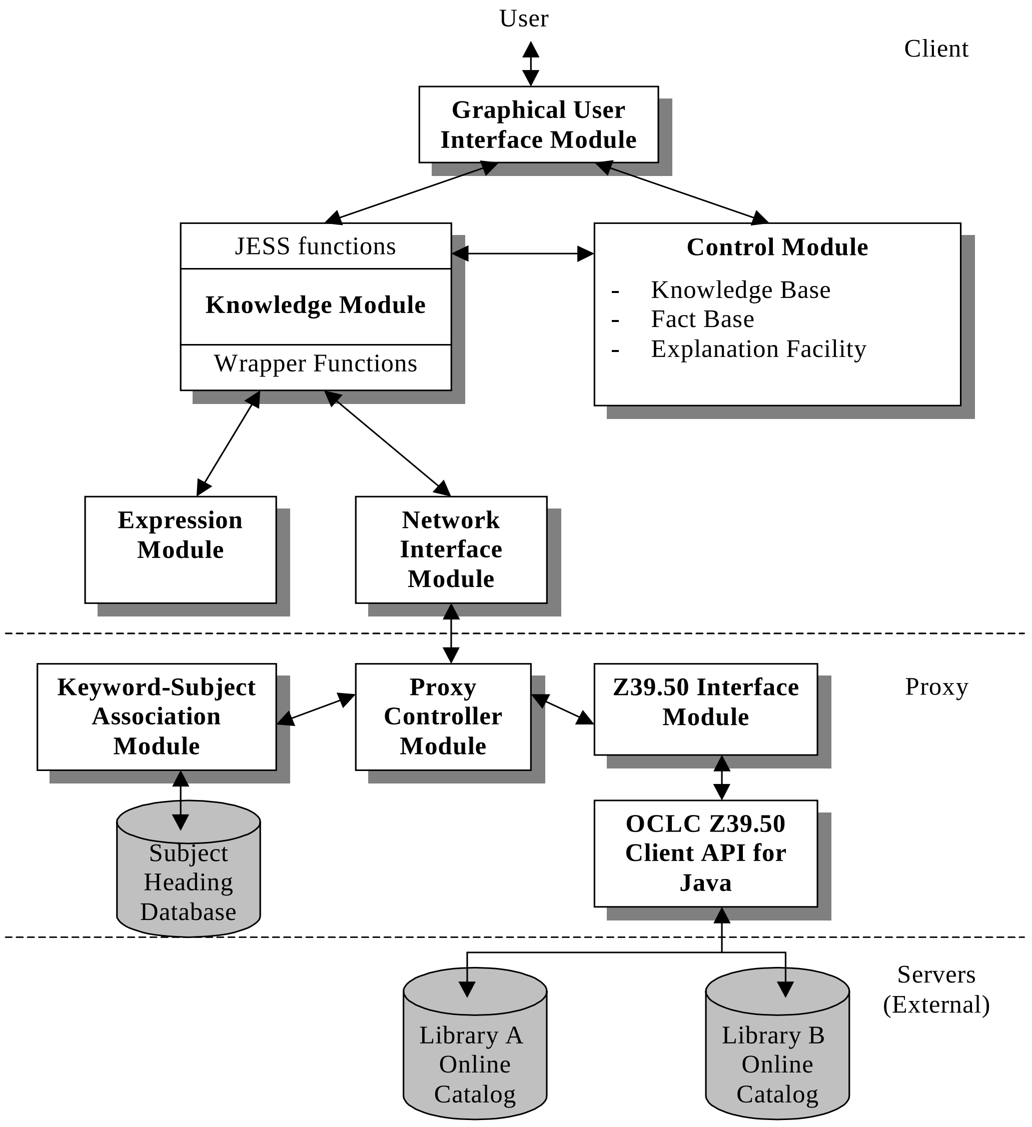


Figure 1: Design of E-Referencer

A screenshot of the main user interface and the feedback window is shown in Figure 2 and Figure 3.

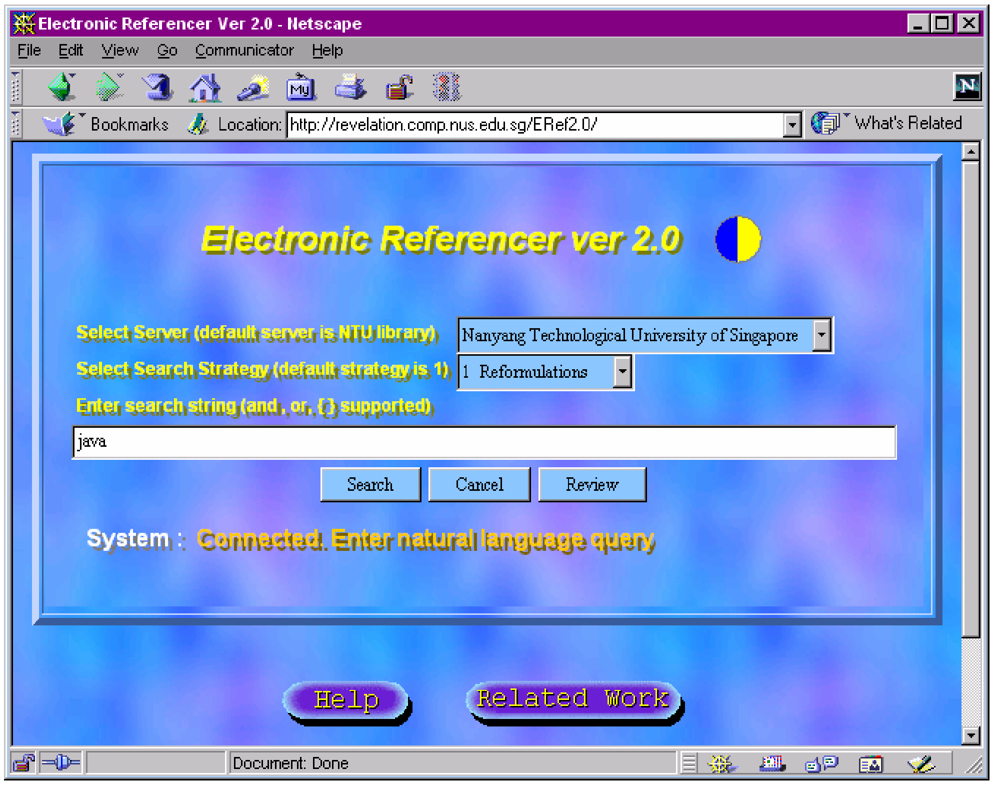

Electronic Referencer Ver 2.0 - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://revelation.comp.nus.edu.sg/ERef2.0/
What's Related
Electronic Referencer ver 2.0
Select Server (default server is NTU library)
Nanyang Technological University of Singapore
Select Search Strategy (default strategy is 1)
1 Reformulations
Enter search string (and , or, () supported)
java
Search
Cancel
Review
System : Connected. Enter natural language query
Help
Related Work
Document: Done

Figure 2: E-Referencer Frame (main GUI)

b. Expression Module

This module provides functions to manipulate the search expression that a user keys into the E-Referencer. The functions implemented are:

- Removing stop-words like `a`, `an`, `is`, `the`, which does not help in the search session.
- Stemming words to remove suffixes to get a common form for retrieving more records. Porter's algorithm [22] available at the Glasgow IDOM – IR Resources Web site (*http://www.dcs.gla.ac.uk/idom/ir_resources/liguistic_utils/*) is selected because of its simplicity and fast implementation.
- Conversion between AND, OR and Adjacent operators in search string.

  Example:

  ```
  AND operators        -> "expert" AND "systems"
  OR operators         -> "expert" OR "systems"
  Adjacent Operators   -> "expert systems"
  ```

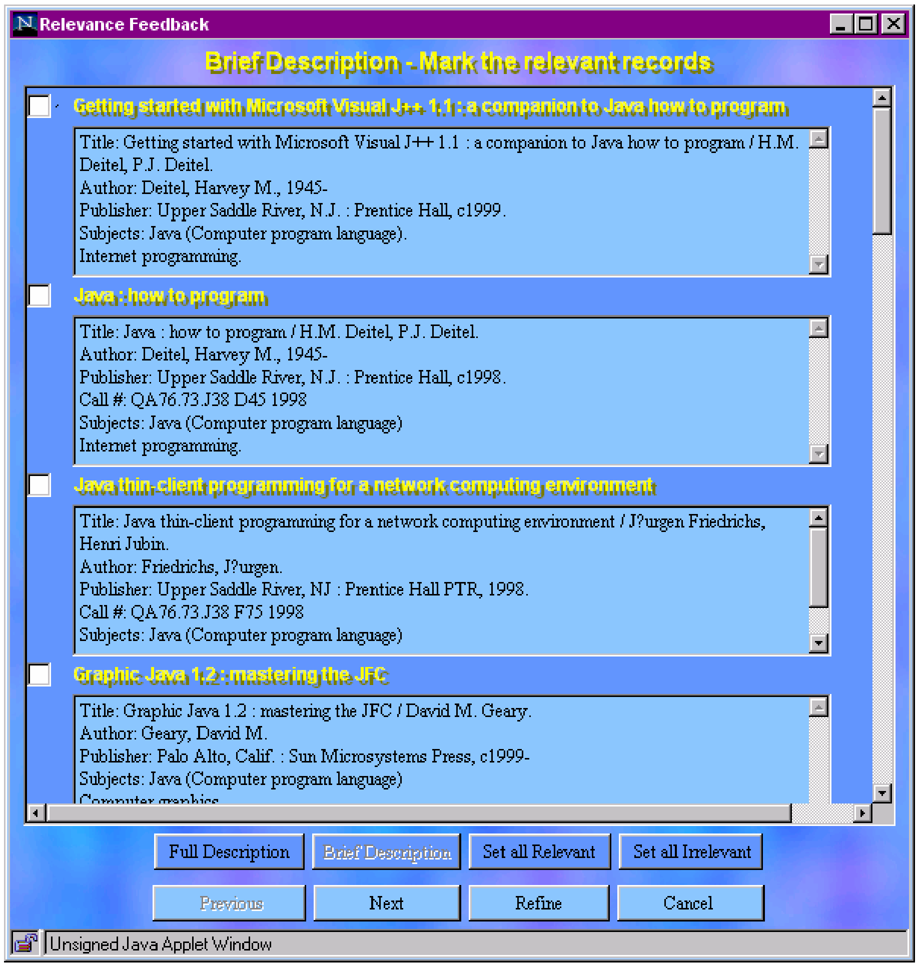
- Create combinations of two or three keywords

  Example:

  ```
  Expert Systems Internet Intranet
  
     Two keyword combination ->
  ```

```
"Expert Systems" OR "Expert Internet" OR "Expert Intranet" OR "Systems
Internet" OR "Systems Intranet" OR "Internet Intranet"

Three keyword combination ->
"Expert  Systems  Internet"  OR  "Expert  Systems  Intranet"  OR  "Expert
Internet Intranet" OR "Systems Internet Intranet"
```



Brief Description - Mark the relevant records

Getting started with Microsoft Visual J++ 1.1 : a companion to Java how to program

Title: Getting started with Microsoft Visual J++ 1.1 : a companion to Java how to program / H.M. Deitel, P.J. Deitel.
Author: Deitel, Harvey M., 1945-
Publisher: Upper Saddle River, N.J. : Prentice Hall, c1999.
Subjects: Java (Computer program language).
Internet programming.

Java : how to program

Title: Java : how to program / H.M. Deitel, P.J. Deitel.
Author: Deitel, Harvey M., 1945-
Publisher: Upper Saddle River, N.J. : Prentice Hall, c1998.
Call #: QA76.73.J38 D45 1998
Subjects: Java (Computer program language)
Internet programming.

Java thin-client programming for a network computing environment

Title: Java thin-client programming for a network computing environment / J?urgen Friedrichs, Henri Jubin.
Author: Friedrichs, J?urgen.
Publisher: Upper Saddle River, NJ : Prentice Hall PTR, 1998.
Call #: QA76.73.J38 F75 1998
Subjects: Java (Computer program language)

Graphic Java 1.2 : mastering the JFC

Title: Graphic Java 1.2 : mastering the JFC / David M. Geary.
Author: Geary, David M.
Publisher: Palo Alto, Calif. : Sun Microsystems Press, c1999-
Subjects: Java (Computer program language)

Figure 3: Feedback Window

c. Control Module

JESS, developed by Sandia Labs, is used to represent the expert knowledge in the E-Referencer. JESS is written entirely in Java and implements a subset of the CLIPS [23] language, which uses production rules to represent knowledge. A production rule consists of a list of facts followed by a list of actions. When all the facts in the list are asserted, the rule is said to have fired and the list of actions is executed. Actions could include asserting more facts, which could fire more rules. In JESS, all production rules are specified in a script. It comes with a set of standard functions and provides a mechanism to wrap functions written in Java as JESS functions. All these functions can then be called from within the

JESS script, which allows for greater flexibility in manipulating the system. JESS was chosen because JESS syntax is simple and search strategies can be specified as production rules easily. Furthermore, the ability to integrate Java functions into JESS makes it very favorable for our use since E-Referencer is written in Java.

JESS uses an inference engine, based on the Rete (Greek word for net) algorithm [24], to process the production rules. The Control Module creates this JESS inference engine, which processes the production rules contained in the JESS script. The components of the Control Module are:

- A Knowledge Base of search strategies. The strategies are represented as production rules in the JESS script `EReferencer.clp`.

  Example of a search strategy:

  ```
  If  "No. of records retrieved = 0" and
      "No. of words in query = 1"
  then
      assert "Broadening Strategy 6" (prompt user to enter synonyms)
  ```

- A Fact Base which stores the intermediate search results and information needed to select the next search strategy. The intermediate search results and information are represented as asserted facts in JESS, which could lead to the firing of other production rules, in some case, the firing of a rule which contains another search strategy.

  Example of facts:

  ```
  "No. of records retrieved = 10"
  "No. of words in query = 5"
  ```

- An Explanation Facility, to explain why and how certain strategy is chosen. The explanations are embedded in the production rules.

Below is a sample piece of code for a search strategy implemented as a production rule in the JESS script. For a more thorough understanding of the JESS syntax and rules, refer to the JESS README *http://herzberg.ca.sandia.gov/jess/README.html.*

```
(defrule BROADENING_STRATEGY1
; Expression has adjacency operators. Convert them into ands.

   ?expr <- (Expr SearchExpr ?str)
   ?strategy <- (BroadeningStrategy 1)
```

```
   =>
   (retract ?strategy)
   (miscPrintout "Broadening Strategy 1: convert adjacent operators to and")
   (miscPrintout "Checking if adjacency operators are present.")
   (if (exprHasAdjWords ?str)                                            10
       then
           (retract ?expr)
           (miscPrintout "Adjacency operators found.")
           (bind ?newstr (exprAdjToAnd ?str))
           (assert (Expr SearchExpr ?newstr))                            15
           (assert (KeyWordSearch))
       else
           (miscPrintout "No adjacency operators found.")
           (assert (BroadeningStrategy))
   )                                                                     20
)
```

The sample code is the production rule for representing the strategy “`broadening strategy 1`”. The lines before the => represent the list of facts, while the lines after => represent the list of actions.

The Fact Base is implemented as facts being asserted. The fact `(Expr SearchExpr ?str)` is always asserted. It is used to store the current search query in the variable *?str*. Thus, in this case, the above rule is fired when the fact `(BroadeningStrategy 1)` is asserted in the Fact Base. Upon firing the rule, the fact `(BroadeningStrategy 1)` is retracted from the Fact Base to prevent further firing of the same rule.

The Explanation Facility is implemented as code embedded in different rules in the script. The code at line 8, 9, 13 and 18 is part of the Explanation Facility and they indicate to the user the strategy used and the actions executed. In this sample code, Broadening Strategy 1 is used, and a check is performed to determine if there are any adjacent words in the query.

Line 14 shows how the JESS script calls the JESS function `exprAdjToAnd` from the Knowledge Module. This function is a Java function that is wrapped as a JESS function, and the method *Call* of the private class `ExprAdjToAnd` will be invoked.

Other facts are then being asserted into the Fact Base so as to fire other rules (lines 15, 16), which may then select another strategy or perform some other functions.

d. Knowledge Module

In order to integrate the rest of the modules written in Java (e.g. the Expression, Subject, Z39.50 Interface and GUI Modules) with JESS, we need to create wrappers for them. All these wrapper functions are grouped according to modules. That is, all the wrapper functions for Expression are grouped under Expression Functions, all the wrapper functions for Z39.50 Interface are grouped under Z3950 Functions and etc. A prefix is then added to each wrapper function to denote the module that they belong to. The Knowledge Module thus contains all these groups of wrapper functions.

Example of wrapper functions:

```
       Expression Functions { exprStem, exprRemoveStopWord, exprAndToOr ...}
       Z3950 Functions      { z39Connect, z39Search, z39Display …}
       GUI Functions        { GUIFeedbackDialog, GUIFrame …}
```

When facts in the Fact Base of the Control Module are asserted, certain rules in the Knowledge Base are fired to select an appropriate strategy. The actions in the strategy make calls to the wrapper functions in the Knowledge Module to perform some required operations.

For example, given these rules

```
Rule 1
If     “No. of records retrieved = 0”   and
       “No. of words in query > 1”
Then
       Assert fact “Broadening Strategy 1”
              :
              :

Rule 2
If     “Broadening Strategy 1”
Then
       ExprAdjToAnd <- wrapper function call: Convert Adjacent to And
              :
              :
```

When a user enters the query *“Expert Systems”* the fact `“No. of words in query = 2”` is asserted in the Fact Base. The search is performed but no record is retrieved. The fact `“No. of records retrieved = 0”` is then asserted. These two facts then fire rule 1 in the Knowledge Base, and it asserts the fact `“Broadening Strategy 1”`; this in turn fires rule 2. Rule 2 calls the wrapper function `ExprAdjToAnd` in the Knowledge Module to activate the appropriate Java function in the Expression Module to convert the Adjacent operators in the original query to AND operators. A new query *“Expert AND Systems”* is eventually created and used to perform another search.

Each function group of wrappers is implemented as a public Java class; they extend the JESS `Userpackage` class. The various wrapper functions are created as private classes and added into the JESS inference engine.

Sample code:

```
public class ExprFunctions implements Userpackage
{
    public void Add(Rete engine)
    {
        engine.AddUserfunction(new exprRemoveStop());
        engine.AddUserfunction(new exprStem());
                    :
                    :
    }
}
```

The above code shows how the class `ExprFunctions` implements the group of wrapper functions for the Expression Module. Other groups are implemented similarly.

Each wrapper function is implemented as a private class extending from the JESS `Userfunction` class. The latter contains a private attribute *_name* to store the name the programmer use for invoking the function in the JESS script. The public method *Call* is then used to define the operations to be performed when the function is called in the JESS script.

Sample code:

```
class exprRemoveStop implements Userfunction
{
    Expression ex = new Expression();
    int _name = RU.putAtom( "exprRemoveStop" );
    public int name() { return _name; }

    public Value Call(ValueVector vv, Context context)
    throws ReteException
    {
        String expr = "";

        if ((vv.size() == 2) && (vv.get(1).type() == RU.STRING))
        {
            expr = vv.get(1).StringValue();
            expr = ex.removeStop(expr);
        }
        return new Value(expr, RU.STRING);
    }
}
```

This code shows how the class `ExprRemoveStop` is used to wrap the Java function `expr.RemoveStop` of the Expression Module into the JESS function `exprRemoveStop`. All other Java functions are wrapped in the same way.

e. Network Interface Module

This module sets up the connection between the client and the proxy. Search requests issued by the client are submitted to the proxy through this module. The search result as returned by the proxy is collected by this module before it is displayed to the user. All networking functions are implemented via the `Socket` class in Java.

3.4 Proxy Modules Design and Implementation

a. Proxy Controller Module

As the name implies, this module controls all the activities and transactions that are carried out in the proxy. It accepts connections from E-Referencer client applets, and establishes connection with the selected Z39.50 server on behalf of the clients. The Z39.50 Interface module is invoked for the connection. If a client applet requests for subject headings that are associated with the keywords it submits, the Keyword-Subject Association Module will be invoked by the controller to retrieve the relevant subject headings. If logging is required at a later stage, it can be implemented in this module since this module handles all transactions.

The Proxy Control Module is implemented using Java Threads. When the proxy starts up, a controller thread is created to listen to client requests. Each time a new client's connection request is received, the controller thread would instantiate two new threads to handle all future requests from that client. One thread will handle all activities between the client and the proxy while the other thread will handle all activities between the proxy and the Z39.50 server the client is connecting to. With two separate threads, there is continuous communication since the blockage of one communication channel will not affect the other. When the connection to the client dies, the two threads will also be killed and reclaimed.

b. OCLC Z39.50 Client API

The E-Referencer is developed as a system capable of searching the many different online library catalogs available. Thus, a standard System Interface module to all these online catalogs is needed. By implementing the System Interface Module based on the Z39.50 protocol, users will be able to access all existing and newly created Z39.50 compliant online catalogs.

A search on the Web for existing resources that can be used in our development effort revealed the following class libraries and toolkits:

- *OCLC Z3950 Client API.* The API is written entirely in Java and implements the latest version of the Z39.50 protocol, version 3 released in 1995. The API is provided free and comes with source code.
- *Index Data, Yet Another Z39.50 Toolkit, YAZ* is a toolkit for implementing the Z39.50v3 protocol. The toolkit supports both the Z39.50 and the ISO10163 SR protocol. Both the Origin (client) and Target (server) roles of the protocol are supported. The toolkit is written in C. YAZ is also provided free.
- *ZedKit for Unix.* The Z39.50 Application Development Libraries is developed for the German Library Project DBV OSI II and also the ONE project co-funded by the European Commission Libraries Programme, and is written is C/C++.

We tested the OCLC Z39.50 Java API by using the sample client application, `zclient,` that comes with the API, to connect to a few Z39.50 compliant online catalogs. Since there are many different Z39.50 server implementers like INNOPAC, DRA and etc., it is necessary to test the OCLC API, by connecting it to a representative few of the various servers implemented by the different vendors.

The API was tested with the DRA server at the NTU Library, the INNOPAC server at National University of Singapore (NUS) and the Ameritech HORIZON server at Clarke College, Dubuque, Iowa. Although some fine-tuning was needed due to differences in implementation by the different vendors, we managed to connect to the various servers, specify the database to search, send some sample search queries and retrieved the records from all the servers.

From the tests, we found the OCLC Z39.50 Client Java API most suitable for our use. The API is written entirely in Java, which makes it easy to integrate into our design framework, and it supports many functions of the latest version of the Z39.50 protocol.

c. The Z39.50 Interface Module

Having decided to adopt the OCLC Z39.50 Client API for development, we then designed the Z39.50 Interface Module; this module serves as a “wrapper” module on top of the OCLC Z39.50 Client API. The module provides simple Z39.50 functions like connect, search, retrieve, close etc. for use by the E-Referencer; these functions are implemented using the OCLC Z39.50 Client API.

There are two advantages for adopting this design. Firstly, the Z39.50 Interface Module can isolate the rest of the program from the OCLC Z39.50 Client API code. In the event of changes to the OCLC Z39.50 Client API or even if there is a need to change to a different API, we will just need to modify the codes in the Z39.50 Interface Module, while keeping the rest of the program code intact. Secondly, by grouping the primitive OCLC API procedures into higher level procedures like search, retrieve etc., we have simplified the coding of the other modules that make use of Z39.50 functions.

A new *`Z39client`* class is created to provide Z39.50 functions like connect, search, retrieve etc. for the proxy. Each function is implemented as a method and makes calls to the OCLC Z30.50 Client API to perform its operation.

From our implementation, we found that the `zclient` application implemented most of the Z39.50 functions supported by the API. We modified the `zclient` source code to create our own *`z39client`* class. Most of the existing methods in `zclient` were modified and reused, but some of the functions as provided by `zclient` were too simple and we had to create new methods to suit our needs. For example, the *display* method in `zclient` does not guarantee that the requested number of records are retrieved and displayed. Therefore, we created a *retrieve* method in our *`z39client`* class, which uses a loop to continuously call the *display* method to retrieve records, and guarantees that the specified number of records are retrieved.

d. Keyword-Subject Association Module

The conceptual knowledge base that maps free-text keywords to concepts represented as LC (Library of Congress) subject headings is a very useful tool for users to clarify their search topic. This conceptual knowledge base is implemented as the Keyword-Subject Association Module in the proxy. The module accesses a subject heading database that contains all the keyword-subject heading mappings for all the keywords found in the LC bibliographic catalogue from 1980-1998. As mentioned earlier, this large database is one of the main reasons that necessitated the three-tier design architecture.

The major task in implementing this module is in the creation of good data structures for storing the keyword-subject heading mappings to allow for efficient retrieval. Since the database is located at a centralized proxy, disk space is not a constraint. An inverted file is thus used to store the keyword-subject heading mappings, to allow for fast and efficient retrieval.

A Subject Heading Map is also created to map each subject heading to a unique number. This is used in the keyword-subject heading inverted file for representing the subject headings using numbers. The use of this greatly reduces the size of the inverted files, because the numbers require much less storage than the subject heading strings. This Subject Heading Map is implemented as a sorted text file so that we can use a binary search algorithm to map subject headings to their unique numbers and vice-versa quickly.

5. Conclusion

The current online catalog search systems do not provide enough assistance for users in their searches. There have been various attempts to develop new systems that utilize different approached to help users in this area. Such systems can be broadly categorized as ‘best match’ systems or knowledge-based systems. Both systems have their merits and problems. However, we felt that the knowledge-based approach is more suitable for use in this domain of online catalog searching, and have thus have adopted it to solve the above-mentioned problem.

A Web-based search interface known as E-Referencer has been developed to provide an accessible and useful search tool for online catalog users. Presently, the E-Referencer is also used as a tool for experimenting with search strategies to create a system that is capable of helping users search effectively. Early test results are encouraging and refinements are still being done on the E-Referencer. We hope that with more testing and iterations of refinements, it can eventually be deployed for widespread use as an effective searching tool for online catalog users.

References

1. Z39.50 Maintenance Agency. URL http://lcweb.loc.gov/z3950/agency/.
2. JESS, the Java Expert System Shell. URL http://herzberg.ca.sandia.gov/jess/.
3. Cousins, S. A.: In their own words: An examination of catalogue users' subject queries. J. Amer. Soc. Inf. Sci. 46 (1992) 329-341.
4. Dalrymple, P.W.: Retrieval by Reformulation in Two Library Catalogs: Toward a Cognitive Model of Searching Behavior. J. Amer. Soc. Inf. Sci. 41 (1990) 272-281.
5. Ensor, P.: User Practices in Keyword and Boolean Searching on an Online Public Access Catalog. Inf. Tech. Libr. 11 (1992) 210-219.
6. Lancaster, F.W., Connell, T.H., Bishop, N., McCowan, S.: Identifying Barriers to Effective Subject Access in Library Catalogs. Libr. Reso. Tech. Serv. 35 (1991) 377-391.
7. Markey, K.: Subject Searching in Library Catalogs: Before and After the Introduction of Online Catalogs. OCLC Online Computer Library Center, Dublin, OH. (1984).
8. Hildreth, C.: Beyond Boolean: Designing the Next Generation of Online Catalogs. Libr. Trends 35 (1987) 647-667.
9. Borgman, C.L.: Why are Online Catalogs Still Hard to Use? J. Amer. Soc. Inf. Sci. 47 (1996) 493-503.
10. Khoo, C., Poo, C.C.D.: An Expert System Front-End as a Solution to the Problems of Online Catalogue Searching. In: Information Services in the 90s: Congress Papers. Library Association of Singapore, Singapore (1991) 6-13.
11. Al-Hawamdeh, S., Ellis, D., Mohan, K.C., Wade, S.J., and Willet, P.: Best match of document retrieval: development and use of INSTRUCT. Proceedings of the Twelfth International Online Information Meeting, (1998) 761-767.
12. Robertson, S.E.: Overview of the Okapi Projects. J. of Doc. Vol. 53, no.1 (1997) 3-7.
13. Guach. S: Search improvement via automatic query reformulation. ACM Transactions of Information Systems, 9 (1991)
14. Vickery, A., Brooks, H.M.: PLEXUS – The expert system for referral. Information Processing & Management, 23 (1987) 99-117.
15. Tong, R.M., Applebaum, L.A., Askmann, V.N., Cunningham, J.F.: Conceptual information retrieval using RUBRIC. In C.T.Yu and C.J.Van Rijsbergen(Eds.), Proceedings of the tenth Annual International ACM SIGIR Conference on Research and Development in Formation Retrieval (1987) 247-253.
16. Drabenstott, K.M.: Enhancing a New Design for Subject Access to Online Catalogs. Libr. Hi Tech, 14 (1996) 87-109.
17. Drabenstott, K.M., Weller, M.S.: Failure Analysis of Subject Searches in a Test of a New Design for Subject Access to Online Catalogs. J. Amer. Soc. Inf. Sci. 47 (1996) 519-537.
18. Drabenstott, K.M., Weller, M.S.: The Exact-Display Approach for Online Catalog Subject Searching. Inf. Proc. Manag. 32 (1996) 719-745.
19. Z39.50: An Overview of Development and the Future.
URL http://www.cqs.washington.edu/~camel/z/z.html.
20. Robertson, A.M., Willet, P., Vickery, A., Thompson, W.: Comparison of Statistically-based and knowledge-based approaches to information retrieval. Inf. 90 (1990) 282-286.
21. Khoo, C.S.G., Poo, D.C.C., Liew, S.-K., Hong, G., Toh, T.-K.: Development of Search Strategies for E-Referencer, an Expert System Web Interface to Online Catalogs. In: Toms, E., Campbell,

D.G., Dunn, J. (eds.): Information Science at the Dawn of the Millennium: Proceedings of the 26th Annual Conference of the Canadian Association for Information Science. CAIS, Toronto (1998).
22. Porter, M.F.: An Algorithm for Suffix Stripping. Program 14 (1980) 130-137.
23. CLIPS: A Tool for Building Expert Systems. URL http://www.ghg.net/clips/CLIPS.html (1997).
24. Forgy, C.L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. Artificial Intelligence 19 (1982), 17-37.

Query No.	Search by NTU Search System			Search by E-Referencer			Search by Librarian		
	No. Displayed	No. Relevant	Precision	No. Displayed	No. Relevant	Precision	No. Displayed	No. Relevant	Precision
A96-7	0	0	0.00	4	0.5	0.13	17	6	0.35
A96-14	0	0	0.00	20	1	0.05	3	2	0.67
A97-16	4	1	0.25	4	1	0.25	20	705	0.38
B97-1	0	0	0.00	20	3	0.15	13	7	0.54
B97-3	0	0	0.00	2	0	0.00	4	3.5	0.88
D96-2	11	10.5	0.95	11	10.5	0.95	20	14	0.70
D96-16	0	0	0.00	20	9	0.45	20	9	0.45
D97-1	0	0	0.00	3	1.5	0.50	20	5	0.25
D97-2	0	0	0.00	5	3.5	0.70	6	6	1.00
D97-4	0	0	0.00	8	2	0.25	15	5	0.33
D97-7	0	0	0.00	19	0	0.00	14	2.5	0.18
N97-13	0	0	0.00	4	3.5	0.88	5	4	0.80
Average	**1.25**	**1.0**	**0.10**	**10**	**3.0**	**0.36**	**13.1**	**6.0**	**0.54**

Table 1: Comparison of Search Results

Note:

1. The figures given for No. Relevant and Precision are the average for 2 sets of relevance judgments by 2 persons.
2. The evaluation is based on the first 20 records retrieved. E-Referencer currently displays only 20 records for relevance judgment.

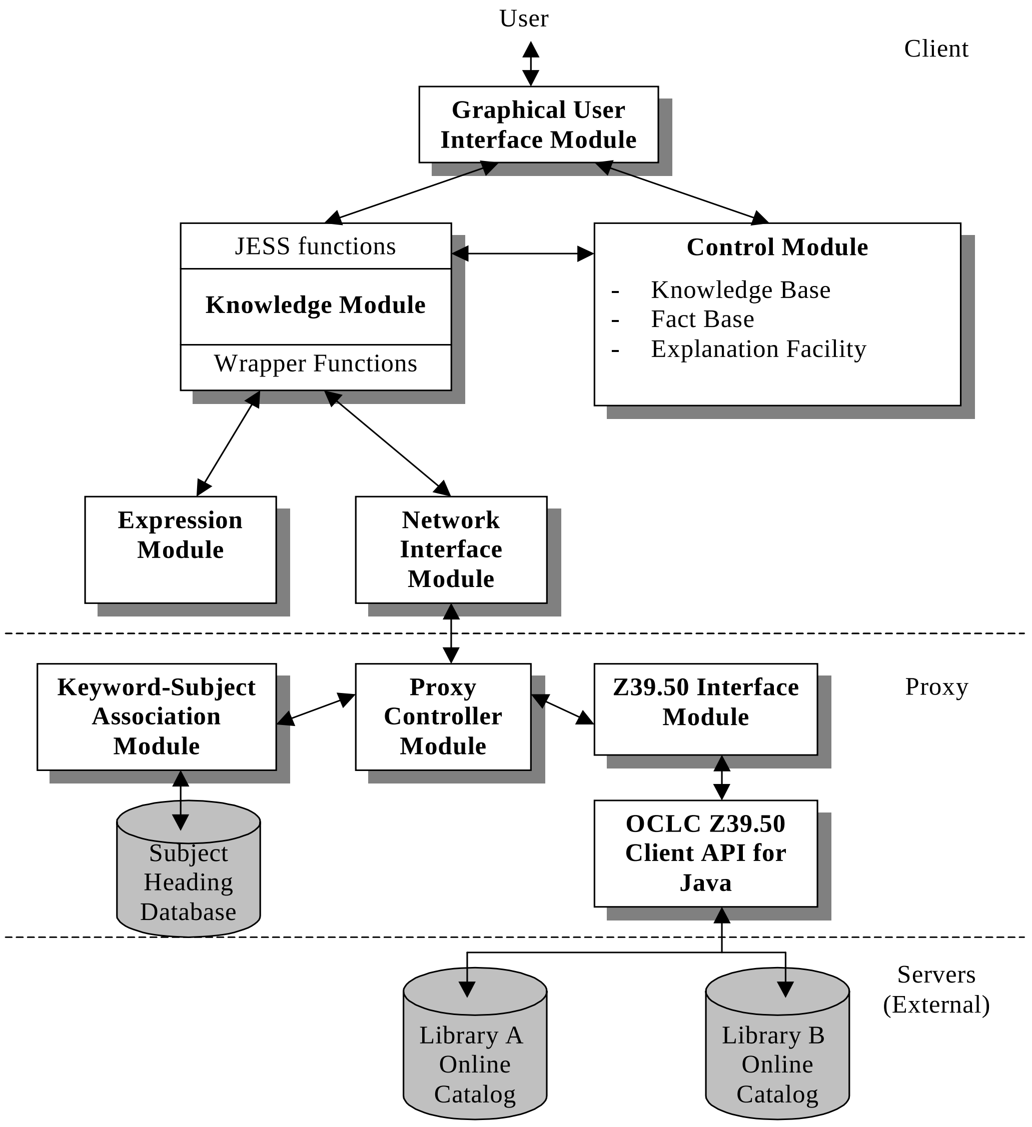
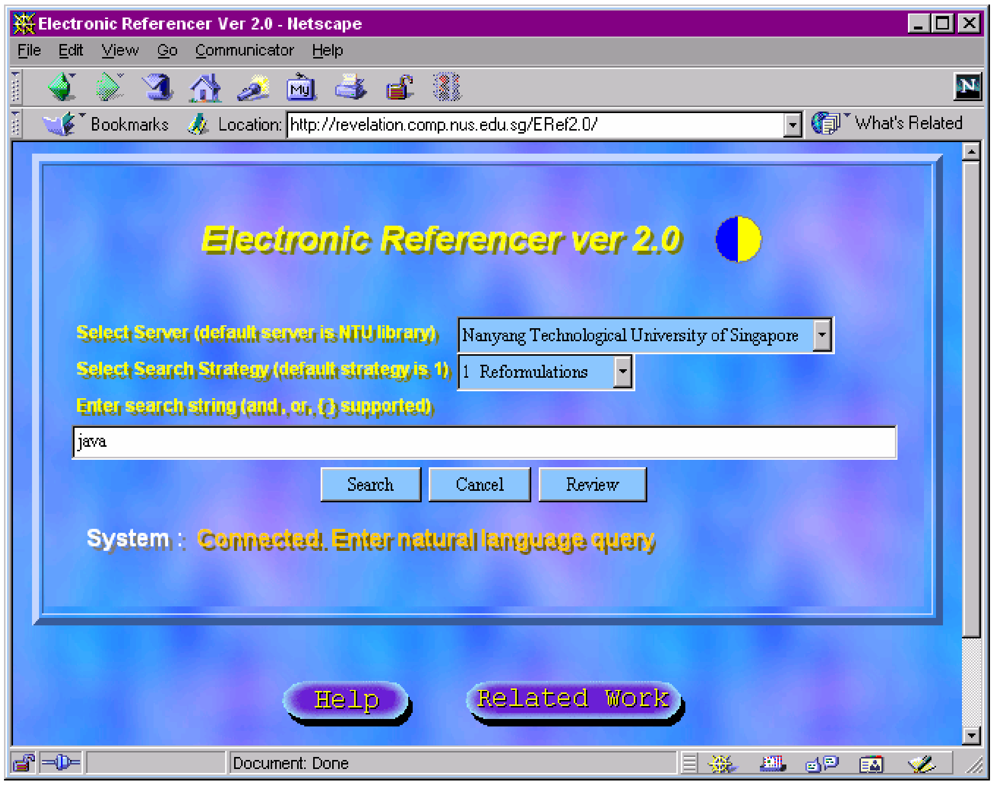


Figure 1: Design of E-Referencer


Electronic Referencer Ver 2.0 - Netscape
File Edit View Go Communicator Help
Bookmarks
What's Related
Search
Cancel
Review
Help
Related Work

Location: http://revelation.comp.nus.edu.sg/ERef2.0/

Electronic Referencer ver 2.0

Select Server (default server is NTU library): Nanyang Technological University of Singapore

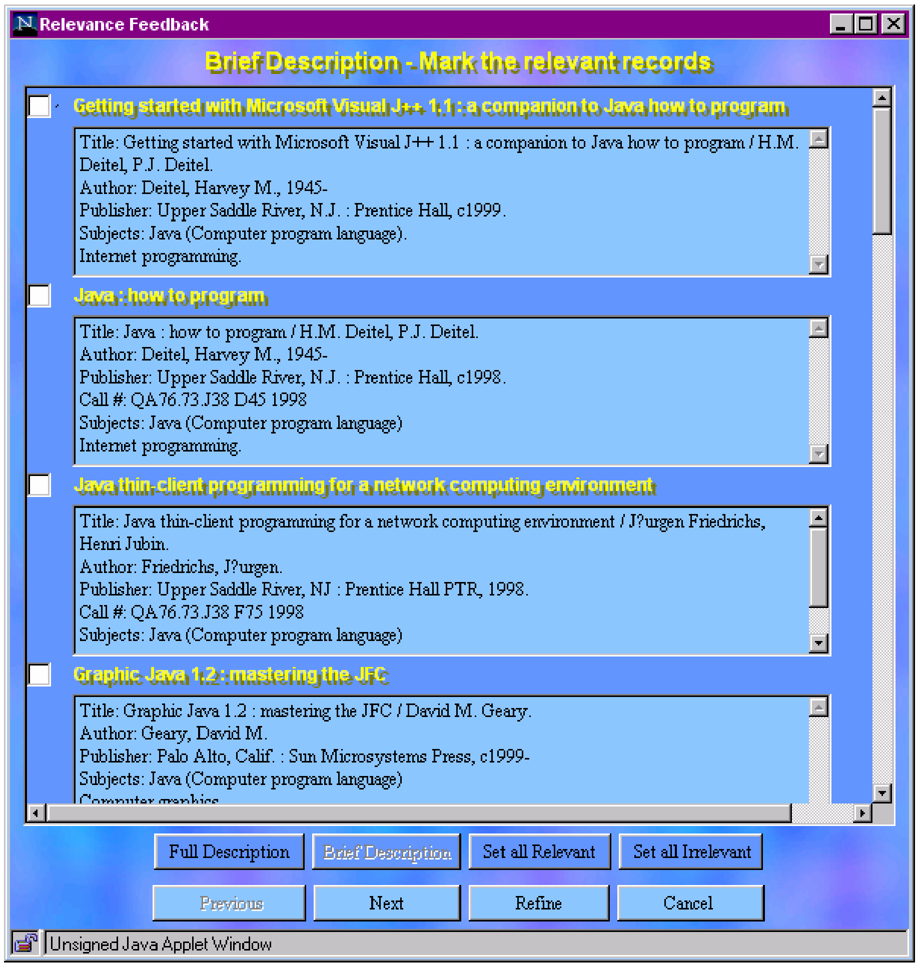
Select Search Strategy (default strategy is 1): 1 Reformulations

Enter search string (and , or, ( ) supported): java

System : Connected. Enter natural language query

Document: Done

Figure 2: E-Referencer Frame (main GUI)



Getting started with Microsoft Visual J++ 1.1 : a companion to Java how to program

Title: Getting started with Microsoft Visual J++ 1.1 : a companion to Java how to program / H.M. Deitel, P.J. Deitel.
Author: Deitel, Harvey M., 1945-
Publisher: Upper Saddle River, N.J. : Prentice Hall, c1999.
Subjects: Java (Computer program language).
Internet programming.

Java : how to program

Title: Java : how to program / H.M. Deitel, P.J. Deitel.
Author: Deitel, Harvey M., 1945-
Publisher: Upper Saddle River, N.J. : Prentice Hall, c1998.
Call #: QA76.73.J38 D45 1998
Subjects: Java (Computer program language)
Internet programming.

Java thin-client programming for a network computing environment

Title: Java thin-client programming for a network computing environment / J?urgen Friedrichs, Henri Jubin.
Author: Friedrichs, J?urgen.
Publisher: Upper Saddle River, NJ : Prentice Hall PTR, 1998.
Call #: QA76.73.J38 F75 1998
Subjects: Java (Computer program language)

Graphic Java 1.2 : mastering the JFC

Title: Graphic Java 1.2 : mastering the JFC / David M. Geary.
Author: Geary, David M.
Publisher: Palo Alto, Calif. : Sun Microsystems Press, c1999-
Subjects: Java (Computer program language)

Figure 3: Feedback Window