Enhancing Search Capabilities of Legacy Internet Resources

Marie Erie, Michelle LeBlanc and Vijay Raghavan*
The Center for Advanced Computer Studies
University of Southwestern Louisiana

Abstract: Many information resources available over the Internet only provide browsing and pre-defined navigation capabilities. If the user is interested in simple word matches on documents as a whole, search engines such as Yahoo or AltaVista may be adequate. For resources of high volume, the user should be able to specify preferences and just obtain the relevant portions of a resource. In this paper, we study the problem of enhancing the search capability to an existing resource by implementing and evaluating two approaches: use of (i) structured (relational) database or, (ii) HTML documents enhanced with meta tags.

I. Introduction

The World Wide Web (WWW) has enabled widely dispersed audiences to access a spectrum of information resources. A variety of Internet search tools [1] exist for retrieving links to documents of potential user interest. Yahoo and AltaVista are examples of search engines that normally examine entire documents, but may allow the user to search only the title or only the body of HTML [2] documents. Searches are for simple matches on limited combinations of words or a phrase input by the user. Results are returned as a series of pages with links to documents that satisfy the request. Based on the brief text associated with a link, the user may then navigate to a document, or a collection of documents within a given Web site, and perhaps decide to search within that resource. Of course, a resource of interest may be obtained by other paths, such as a link from a parent document or directly from a known URL address. Once obtained, it is often the case that the server offers the user little more than simple navigational links based on a predefined breakdown of the information at the site. Alternatively, one could use the FIND function of the browser to look for simple word patterns. For successful and efficient searching through resources of high volume, the user should be able to specify one or more terms of interest (e.g. of a certain semantic type) in a search request that returns only that portion of the document that matches the request.

To this end, in this paper, we address the issue of enhancing the search capabilities of legacy Internet resources, (i.e. old-style pre-existing resources developed prior to the advent of more modern methods). In section II, we begin by establishing the requirements and desired features of a search tool designed to augment an existing resource. Section III introduces two approaches to creating a user search interface by which the user may find desired information from a resource. In section IV.A, we describe a large online document local to the University of Southwestern Louisiana (USL) as an exemplary legacy Internet resource offering only browsing capability. In sections IV.B and IV.C, we detail two implementations of a search tool for this resource. The two approaches are modeled after the architectures described in section III. Both

* CACS, 2 Rex St., P. O. Box 44330, USL, Lafayette, LA 70504
ph: (318) 482 6603, fax: (318) 482 5791, raghavan@cacs.usl.edu

implementations enable the user to specify search terms in particular fields of the resource in order to obtain only that small portion of the resource matching the sets of field-term pairs. In section IV.D, we contrast the two approaches with respect to the requirements outlined in section II. Section V concludes by stating our preferences among the two implementations.

II. Requirements/Preferences for Tools to Enhance Within-Document Searching

Given that the condition of an existing Internet resource may warrant the construction of a tailored search tool *subsequent* to the resource's availability online, we outline features to be considered in designing such a tool. We propose the following criteria for assessing alternative approaches.

- *Ownership of source database*: When the existing Internet resource was developed from a source database that is owned locally and is available, it may be preferable to access the original database than to attempt to define search and target elements by parsing the online document. It may also happen that special agreements could be worked out with the creator of the source database, if not locally available.
- *Automation of data transformation*: If the original database is available in a structure that can be queried via DBMS commands embedded in a host language, then no preprocessing may be required. Otherwise, the original database may be exported to a structured (relational) database. This will require some degree of preprocessing. If the source database is not available, the online document must be parsed for recognizing search and target elements. Thus, in either case, preprocessing may be unavoidable. What is to be considered is the degree to which any preprocessing phases can be automated.
- *Data duplication and storage*: Ideal scenarios come in two flavors: (1) The online document can be parsed for search and target elements and only metadata[1] about each data element's type and location within the online document is maintained locally in a relational database. (2) The source database is locally available in a structure ready for access via host language programming. In the first case, there is no duplication of data and only minimal information about the existing online document is stored locally. In the second case, there is no duplication of data from the online document or from the source database.
- *Implementation effort*: The search tool development can be divided into five phases: (1) preprocessing to obtain structured data or structured metadata, (2) construct user's online input form, (3) retrieve users input, (4) construct query and access data, (5) construct an online results page to be returned to the user. The expenditure of effort and time involved in implementing these tasks should be weighed in the design phase.

[1] *metadata* is "information about information". For example, in an HTML document one may deonte that the first phrase in the document is the title by enclosing that phrase by the metatag pair <title></title>. For bibliographic citations, some other categories of metadata are 'author', 'publisher', and 'date'.

- ♦ *Query types available to user*: The existing Internet resource should be examined to determine the range of query complexity that might be useful to the user. If using a structured (relational) database, allowable query types can be tailored to user needs. If using software, such as Isearch [3,4], for automating the indexing and searching of documents, then the software dictates what type of queries are available to the user.
- ♦ *Maintenance of state*: Between the initiation of a search request by the user via the submission of an input form and the receipt by the user of the final results of the search, there may be several communications between the server and client[2]. When an implementation uses the Hypertext Transfer Protocol (HTTP) [5] and Hypertext Markup Language (HTML) [2] in combination with the Common Gateway Interface (CGI) [6,7], a continuous execution state is not maintained and must be mimicked in the CGI script. One may choose to consider an implementation that integrates with ANSI Z39.50 [8] servers which operate on a protocol that maintains state. (HTML, CGI, and Z39.50 are briefly discussed in section III).
- ♦ *Performance*: Search methods should take advantage of indexes created on search elements. This feature will play a significant role in speeding up the return of search results. A related issue is whether or not a method supports ranked output.
- ♦ *Search tool maintenance*: One of the most prominent features of the WWW is its ever-changing state. Documents are being added or updated at a phenomenal rate. Since the topic of this paper addresses the issue of creating tools to enhance searches of existing Internet resources, the design should take into account the fact that the information in the online document, and perhaps the way it is organized, is likely to change over time. Although the manner in which an Internet resource may change cannot be predicted, providing an implementation that renders various phases modular is likely to equate to less effort in maintenance.

With these guidelines in mind, we now outline two Internet-computing design approaches that may be applied to the task of Internet resource search enhancement.

III. Two Architectures for Searching within an Internet Resource

A. HTML and CGI with a Relational DBMS (RDB/CGI)

HTML documents are one type of document that a WWW server presents for display on the client's browser. HTML tags dictate the document display format. CGI is a mechanism that allows servers to execute external CGI programs that accept user-specific data from the client via HTML forms [7]. Figure 1 shows the architecture for this interface. These HTML documents are authored with FORM, INPUT, and SUBMIT tags. This allows the user to place values into the form. When the user submits the form, the values are passed on to the server via a URL request that calls the CGI program which processes the user input. The figure shows the architecture for a CGI program that queries a structured database (DB) system by using embedded SQL (Structured Query

[2] *server* is a computer or a program that provides a service on the Internet such as FTP and file access.
clients are the computers or software programs, such as a browser, that access these services.

Language) statements. The program completes execution by generating a new HTML document that contains the program's results. The server passes this document back to the Web browser for display. There may be several iterations of this cycle, with new user input, before the user is satisfied that information retrieval is complete. The HTTP protocol by which these processes are spawned does not preserve the state of execution from one cycle to the next. In order to maintain state information, the CGI program is written such that when it outputs the results to the user, as a new HTML document, it includes state information from the previous cycle in hidden fields, denoted by user INPUT tags with a HIDDEN attribute.

A potential bottleneck in this approach is that, since processing is restricted to the server-side, a large client base could place a heavy load on the server. In addition, the lack of statefullness associated with HTTP can lead to excessive data transfer during a cycle of execution.

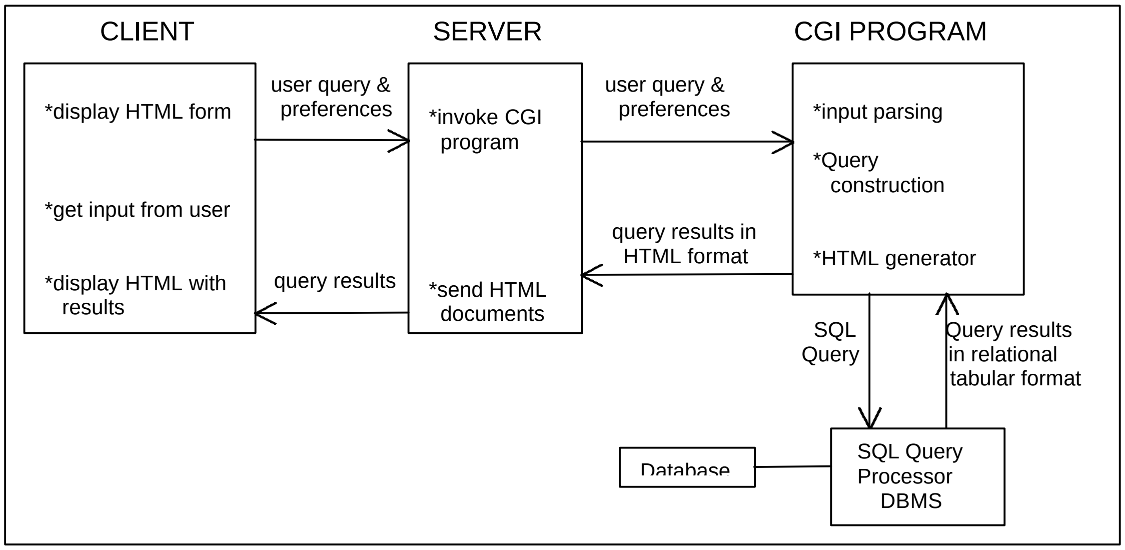

Figure 1. Architecture for RDB/CGI Application

With respect to the topic of this paper, the structured database in the figure can be thought of as the original data from which the online resource was created. In the event the database is not structured or cannot be accessed by a SQL, then the database must be ported to a structured form.

B. HTML Metatag Insertion Integrated with an Information Retrieval System

The Isite/Isearch system [3,4,9] is a text retrieval system that allows one to retrieve documents according to several classes of queries. We describe this alternative approach to providing search enhancement assuming that Isite is used as the information retrieval engine. This alternative uses a client/server architecture involving Z39.50 protocol and thus maintains state information throughout a search session. The architecture of such an application is shown in Figure 2. The architecture is similar to that of RDB/CGI applications shown in Figure 1 but with the distinction that HTTP is a *transaction* oriented protocol whereas Z39.50 is *session* oriented, meaning it preserves state. When a client contacts a server with a Z39.50 protocol request, the server establishes a connection to a Z39.50 server via the Z39.50 Gateway. The Z39.50 Gateway and Server functions in a way somewhat analogous to the CGI program in the RDB/CGI architecture. That is, the Z39.50 server executes the search against a database (or a distributed database). The database system shown uses Isite's Isearch/Iindex software which adheres to Z39.50 information retrieval standards. The Isite database is composed of documents, indexed by Iindex, accessible by Isearch.

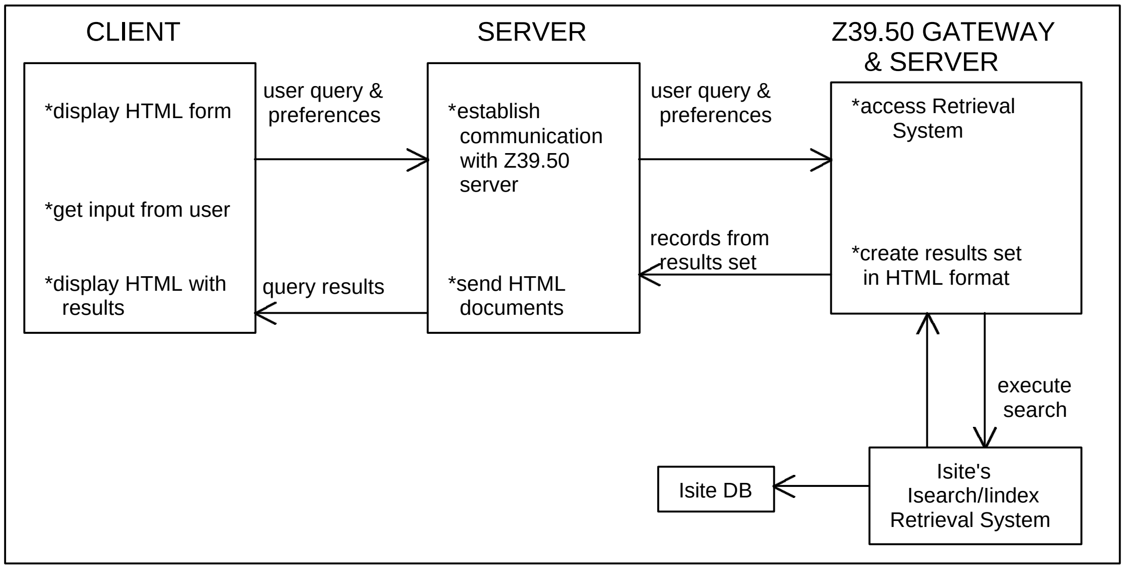

Figure 2. Architecture for Isearch Application

Isearch features give the user many options for composing queries with search and target elements; options not offered by a SQL embedded CGI application without significant programming effort. The Simple Search allows the user to perform case-insensitive search on one or more search elements (fields). Partial matching to the left is allowed. The Boolean Search allows the user to compose a two-term query where the two terms are related by one of the Boolean operators AND, OR, or ANDNOT. "Full Text" is the default search domain unless the user selects a particular element for a term from the term's pull-down menu. The Advanced Search form accepts more complex Boolean queries that are formed by nesting two-term Boolean expressions; for example: "Acadian AND (language OR dialect)". To narrow a search domain from "Full Text" to hits occurring within a single search element, the term is prefixed with the element name and a forward slash. For example, the query "AUTHOR/Dobbs AND TITLE/education" searches for 'Dobbs' only in the 'AUTHOR' element.

The information targeted for return by a query may be specified by choosing target elements from a pull-down menu. The user may also choose to view a maximum number of items in the results set at a time. Isearch is capable of performing a weighted search based on search term frequency. A term's weight increases proportionately to a term's occurrence frequency within a document, but decreases as the number of documents in which the term appears increases. The statistics for all search terms are combined to establish a ranking among the members of the results set. The results set is ordered, for viewing, with the highest ranked results first.

To compare these two design architectures, we selected an existing Internet resource for which a real-world community of users exists and who expressed to us a wish for a better search interface to this resource. We have implemented two different document-specific search engines that address their needs. The next section is devoted to the details of this work.

IV. A Case Study: Two Implementations that Enhance Searches of a Local Internet Resource

A. The *1996 Abridged Bayou State Periodical Index* as an Internet Resource

The *Bayou State Periodical Index* [10] is a guide to Louisiana periodicals which is published yearly by the USL Edith Garland Dupre Library. It is organized into three main sections. The first is a listing of cited Journals with publisher and distribution information. The second and largest section, the Subject Index, lists all journal citations alphabetically by subject. The last section is the Author Index which lists all journal citations alphabetically by author. The existing Internet resource for this index , *The 1996 Abridged Bayou State Periodical Index*, is an HTML document located at URL: http://www.usl.edu/Departments/Library/departments/larm/abspi.html. It contains approximately one quarter of the entries from the unabridged hard-copy version. This resource is limited in that it only allows the user to browse the index sequentially by subject. Additionally, the entire document is partitioned into four segments grouped by alphabetical links for navigational purposes: A-C, D-K, L-P, Q-Z. The citations in the unabridged (hard-copy) index contain one or more of the following attributes: subject, author, title, journal (or periodical), volume, issue number, pages, date, and year. In addition, in the subject index, there may exist cross-references to other subjects, prefixed by *See* or *See Also*. For such resources of high volume, information retrieval would be greatly enhanced by a search engine that allowed the user to go directly to only one or a few citations by specifying values of some subset of the available attributes (search elements). We addressed this problem by implementing the architectures described in section III above. The details for each are outlined in the following subsections.

B. Relational Database Accessed via HTML forms and CGI Scripts (RDB/CGI)

In the implementation discussed in this section, we chose to create a search tool that accesses the original source data used for the hard-copy publication rather than process the abridged version located at the URL noted in the previous section. The URL for this implementation is located at http://www.cacs.usl.edu/cs561-bin/Bayou.cgi.

The original data for the *Bayou State Periodical Index* resides in an AUTHEX *Plus* [11] database system in two main files. The "database" file is composed of records containing data on bibliographic citations. Each record is unique by a <title> field and contains other fields for entering the citation data. A "database" record also includes an attribute <subject> whose value is a list of subjects under which the citation is a member. The "subject" file contains records which are unique by <subject>. Two of the subject file fields relevant to this implementation are the *See* and *See also* subject cross-referencing fields. The "database" and "subject" files are ultimately joined by the <subject> field, even though a "database" record's subject field may contain a list and a "subject" record's subject field may not. Since the AUTHEX *Plus* software does not provide a host language-based access to its data, the "database" and "subject" files were obtained in export format: ASCII files with delimited records and fields. The files were parsed for relevant data items which were then loaded into a relational database, ORACLE® [12]. Proprietary structure existing *within* a field in the AUTHEX *Plus* export files is accounted for at the outset during parsing, thus precluding any need for manual editing of the load data files. The tables comprising the schema are defined as follows:

SUBJECT

SUBJ_NO	SUBJ_NAME

SUBSUB

SUBJ_NO	REF_TYPE	REFERENCE

BIBLIOGRAPHY

BIB_NO	TITLE	JOURNAL	VOLUME	ISSUE	PAGES	MONTH	YEAR

BIBSUB

BIB_NO	SUB_NO

AUTHORS

BIB_NO	NAME

Assuming some knowledge of structured databases, these tables are self-explanatory with perhaps the exception of the SUBSUB table. It's REFERENCE field holds a subject number value for one of the two mutually exclusive subject cross-reference types, *See* and *See also*. REF_TYPE holds a flag to indicate which reference type is in the REFERENCE field. These fields were placed in a separate table since most subject entries have NULL cross-reference fields. The overall design was chosen so as to compress the data while allowing full join capability.

We organized the originating HTML form into four options (via radio buttons) to a journal list, an author list, a subject search form, and an author search form. Selection of one of the four categories followed by selecting the SUBMIT button, invokes a CGI script which, depending on the query type, produces an HTML page specific to that search. Each page has a link back to the originating page so as to permit the user to perform another search. A link to the *1996 Abridged Bayou State Periodical Index* is also provided.

The journal list submission returns the results of a query to the BIBLIOGRAPHY table and is simply a listing of all journals cited in the *Bayou State Periodical Index*. The author list submission similarly returns the results of a query to the AUTHOR table. It is an HTML form which alphabetically lists all authors associated with bibliographic entries. The list is headed by an alphabetical index that allows the user to link to a particular section of the list. Each author's name is the value of a radio button, and when selected and submitted, the CGI script initiates a new database search for all bibliographic citations by the chosen author. The results are returned as an HTML page in the format found in the *Bayou State Periodical Index's* Author Index.

The Subject Search and Author Search forms allow the user to input a subject and author name respectively. These searches allow for partial matching and are case-insensitive. An optional "year" field allows the user to narrow the search further. This field was included in anticipation of the addition to the database of journal citations for future years. All matching bibliographic entries are returned, organized by subject or author, depending on which form was submitted. The results returned for the Subject Search have the added feature that the *See* and *See also* cross-reference fields, if present, have the associated subject as the value of a radio button. The cross-reference may be selected and submitted, taking the user immediately to the results of that subject search.

It might be desirable to add optional author input to the subject form, and vice-versa, allowing the user to refine the search. Furthermore, *any* other attribute of the tables could be included in a form for search purposes, accompanied by minor changes to the CGI script.

C. HTML Documents Enhanced with HTML Metatags

In this approach, the existing resource, *The 1996 Abridged Bayou State Periodical Index*, was standardized by extending the HTML document with metatags. These are tags which delimit data elements in the document to indicate information about those elements. The syntax of their use is the same as that of standard HTML tags but a Web browser ignores them. One HTML document was produced for each set of citations under <subject>. This set of documents was indexed by the Isite software [9] for subsequent searching using the Isearch utility [3,4]. The inserted metatags are recognized by this software. The Dublin Core scheme [13] was used, which has been proposed as a standard for metadata for bibliographic citation data. The Dublin Core metadata element set is a core set in that it is a small number of elements of general applicability. Of this set, the ones relevant to the current implementation are Subject, Title, Author and Date. The extensibility of the Dublin Core scheme allows for the addition of other elements to the set. For enhancing the search and selectively displaying certain target portions of the existing resource, Periodical, Volume and Page elements were added to the set. Figure 3 shows an example of one entry from the index's HTML source.

```
<P><B>ACADEMY OF THE SACRED HEART, New Orleans<BR>
</B>Sacred Heart restores shutters and cupola.<BR>
<I>Preservation in Print</I>v23 n6 p28, Aug, 1996<BR>
<BR>
```

Figure 3: HTML tags used in the index

It was noted that <B> and </B> enclosed the subject, </B> and <I> enclosed the title, <I> and </I> enclosed the periodical, and </I> and </BR> enclosed the volume, pages, and date. The online resource was parsed and reproduced in a form that could be used for indexing and later search. Figure 4 shows the HTML metadata resulting from parsing the citation shown in Figure 3.

```
<Subject><B> ACADEMY OF THE SACRED HEART, New Orleans</B></Subject>
<Title> Sacred Heart restores shutters and cupola.</Title>
<Periodical><I> Preservation in Print</I></Periodical>
<Volume>v23 n6</Volume><Page>p28,</Page>
<date>Aug. 1996</date>
```

Figure 4: HTML meta-data used for citation indexing

The syntax in Figure 3 is not consistent throughout the resource and some manual editing was required to produce accurate HTML metadata for all citations. For example, many citations have no entries for some of the fields. In addition, after the subject field, a small percentage of the citations have the *See* and *See also* fields delimited by the <I></I> tag pair. Similar inconsistencies were found in other fields.

The interface for the indexed abridged dataset offers links to a Simple Search page, a Boolean Search page, and an Advanced Search page. This menu is located at URL: http://www.cacs.usl.edu/cs561-bin/wmb2/wmb2.cgi.

The Simple, Boolean and Advanced Searches are discussed above (in III.B). Search and target elements are: Subject, Title, Periodical, Volume, Page and Date. A range of dates is allowed. This built-in Isearch feature may become relevant in future years when a separate HTML page for another year is expected to be available for processing.

Detailed information about the user's results are available in the 'Search Summary' entry at the top of each search's "results page". It shows how many times each query term occurred and in how many documents, as well as total time required to perform the search. An on-line Help page is accessible via a Netscape browser.

D. Strengths and Limitations of Implementations

Equipped with the above details of both implementations, we turn to evaluating the degree to which they meet the requirements and preferences outlined earlier.

- *Ownership of source database*: The relational database implementation is a true replica of the source database since it offers up to the user the entire unabridged resource and bypasses the legacy Internet resource altogether. The Internet resource represents one quarter of the source database and author information is not included. In this respect, the RDB/CGI scenario is preferable to the second implementation.
- *Automation of data transformation*: The completeness and accuracy of the parsing phase in the RDB/CGI approach is attributed to the precise delimiting of records and

fields in the source database export files. The ordering and adjacency of fields is preserved by delimited NULL values. Thus, data element type identification is not corrupted in the DB porting process. This is to be contrasted with the HTML metatag approach, where parsing the online resource relies on the pairing of HTML tags. The syntax varies from one citation to the next since NULL fields are left undetected. The attribute of a given field may then be ambiguous. The attributes Volume, Number, Page and Date must be extracted from between the single tag-pair </I><BR>. The fact that each attribute is not guaranteed to be present, poses some problem for automation. It is also noted that the abridged online document required some manual editing before the final version became available at its current URL site.

- ♦ *Data duplication and storage*: Both implementations require that a duplication of data reside locally, in addition to the original HTML abridged index. Were it not for the need for manual editing of the original abridged index for proper insertion of the metatags, the HTML metatag method would have the potential for maintaining only the metadata locally, in addition to the existing online Internet resource. Multi-level parsing of the abridged index may yet resolve this issue. By contrast, non-local data storage is not a possibility for the RDB/CGI approach since the Internet version of the resource is never accessed. It is precisely the local structured DB that must be available to the CGI application.
- ♦ *Implementation effort*: Referring to the five phases of application development enumerated in section II under *'implementation effort'*, only the first phase, preprocessing to obtain structured data, poses a challenge to the HTML metatag approach. All other phases are provided for by the software package. For the RDB/CGI case, all phases must be coded by the developer of the search tool. Most phases are straightforward in terms of program development. With respect to query construction, coding effort will depend on the range of query types made available to the user.
- ♦ *Query types available to user*: The RDB/CGI implementation has the potential for a full range of query types equivalent to that of the Isearch software. The simplicity of the periodical index does not warrant the need for complex searches, so the query versatility offered by Isearch may be considered extravagant. With respect to partial matches, the two approaches interpret search terms differently. For a relational approach, an atom is the whole string (possibly, with several words) stored as the value of an attribute, with partial matching occurring at the ends of the string. For Isearch, each individual word in a phrase is atomic. If the user wishes to search on part of an atomic unit, it is the user who inputs the wild card. If a wild card is used in querying a relational DB, then in many systems (e.g. ORACLE®), sequential searching is performed, regardless of index structures created on the search element.

The ability of CGI scripting to provide special-purpose applications enabled us to construct query types that are not possible with Isearch without significantly modifying its source code. Specifically, we were able to return results with values associated with radio buttons: (1) subjects associated with *See* and *See also* in Subject Search returns and (2) author names in the Author List return. The ability to initiate a new search by selecting these buttons is a very attractive feature of the RDB/CGI approach. In the current HTML metatag implementation, the user would have to make a note of any subject referenced by *See* or *See also*, return to some type of search input form, enter the subject name and submit. Since the online periodical index has been generated without author data, author searches are not available in the latter implementation.

- ♦ *Maintenance of state*: It is well documented that a main disadvantage of the RDB/CGI option is its dependence on HTTP, which is a stateless protocol. In this respect, building an application that interfaces Isite/Isearch through a Z39.50 client is clearly preferred.
- ♦ *Performance*: We have not made a quantitative comparison between the two implementations on this criterion. What Isearch does have to offer are the ranking and timing statistics that are returned as part of the results page. As of this writing, we have not indexed the relational DB that is accessed by the CGI script. This is a simple matter and would need to be done before any quantitative comparison can be considered meaningful. Qualitatively, the two implementations appear satisfactory in terms of response time.
- ♦ *Search tool maintenance*: Addition of new data to the online resource poses a maintenance problem for the HTML metatag implementation since the creation of new HTML "citation" documents to be indexed by Isite will require the preprocessing to be redone. For the RDB/CGI option, the preprocessing is fully automated. All that is required for update is a session of parsing AUTHEX export files for a new year and loading the data into the ORACLE® DB.

V. Conclusion

Although the use of HTML metatags for indexing and searching is a new and sometimes desirable approach to the problem of search enhancement, the peculiarities of a given resource may prove cumbersome enough to warrant the use of more traditional methods such as porting one database to another, even at the expense of duplicate storage. The original database for the resource of this study is maintained locally, making the relational database approach a viable option. Weighing the above requirements, we suggest that, when the source database is available, it is preferable to construct a tailored search engine that accesses the source directly rather than adopt a method requiring the processing of an output document from that source.

Acknowledgments: The implementation using HTML metatag insertions and use of Isite/Isearch software as described in section IV.C. was created by Weil M. Badawy, a Ph.D. computer science graduate student at CACS, USL. We acknowledge Ms. Sheryl Moore and Ms. Jean Kiesel of the USL Dupre Library and Ms. Judy Buys of the National Wetlands Research Center for their guidance during the implementation of the systems described in this paper. This work is supported in part by a grant from the U.S. Department of Energy (under Grant No. DE-FG02-97ER1220).

Author biography:

Marie Erie and Michelle LeBlanc are Computer Science Ph.D. students at the Center for Advanced Computer Studies (CACS), University of Southwestern Louisiana. Marie's interests are in medical visualization and physics-based modeling in computer graphics. Michelle's interests include artificial intelligence, database systems, and user interface design.

Vijay Raghavan is a professor of computer science at CACS. His research interests are in information retrieval, database mining and Internet computing. He directs a project, funded by the DoE, aimed at developing an Energy and Environmental Information Resources Center.

REFERENCES

1. V.N. Gudivada, V.V. Raghavan, and R. Kasanagottu, *Information Retrieval on the World Wide Web*, IEEE Internet Computing, Vol. 1, No. 5, Sept.-Oct. 1997, pp. 58-68. Also available at RL:http://www.cacs.usl.edu/~raghavan/raghavan.html
2. *A Beginner's Guide to HTML*, NCSA, URL: http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html
3. *Isearch Text Search Engine*, URL: http://www.cnidr.org/ir/isearch.html
4. *A brief tutorial on using Isearch*, URL: http://vinca.cnidr.org/software/Isearch/Isearch.html
5. Barners-lee, T., Fielding, R., Frystyk, H., *Hypertext Transfer Protocol – HTTP/1.0*, Network Working Group, RFC1945, May 1996., URL: http://www.ics.uci.edu/pub/ietf/http/rfc1945
6. Robinson, D.R.T., The WWW Common Gateway Interface Version 1.1, URL: http://weeble.lut.ac.uk/System-docs/Internet-drafts/draft-robinson-www-interface-01.txt, IETF, Feb. 1996, and http://www.w3.org/pub/WWW/CGI/Overview.html
7. *An Instantaneous Introduction to CGI Scripts and HTML Forms*, Academic Computing Services, University of Kansas, URL: http://www.cc.ukans.edu/~acs/docs/other/forms-intro.shtml
8. *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification*, URL: http://www.lcweb.loc.gov/z3950/agency/
9. *Isite Information System*, URL: http://www.cnidr.org/ir/isite.html
10. *Bayou State Periodical Index 1996*, Jean S. Kiesel and Ashley E. Bonnette, eds., U.S.L. Libraries, Lafayette, LA, 1997.
11. *AUTHEX Plus Manual*, Reference Press, Ontario, Canada, 1990.
12. ORACLE Corp. WWW Home page at URL: http://www.oracle.com/
13. *"Mapping the Dublin Core Metadata Elements to USMARC"*, report 86, URL: gopher://marvel.loc.gov:70/00/.listarch/usmarc/dp86.doc, June 1995.