

Results Merging in the Networked Computer Science Technical Reports Library (NCSTRL)

A Final Binding in TCC 402

Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science

by

Eric J Nelson
December 17, 1997

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid
as defined by the Honor Guidelines for Papers in Humanities Courses.

Eric John Nelson

118 Raymond Ave
Charlottesville, VA 22903
804-296-2720 (H)
ejn2z@virginia.edu

Objective

A challenging full-time position, beginning in January, in the field of information services or related areas. Responsibilities should include software design and development.

Education

University of Virginia *Charlottesville, VA*

Studies in Computer Science towards a Bachelor of Science.
GPA in Major: 3.4

Experience

Summer 1997

ENSCO, Inc *Springfield, VA*

Internship. Developed applications in C++, Visual Basic, and LabView. Gained experience in embedded software and databases.

1996 - Present

University of Virginia *Charlottesville, VA*

Teaching Assistant for CS201-Software Development Methods and CS216-Data Structures. Responsible for grading student projects, conducting formal assistance in C++, administering closed session laboratory exercises, and proctoring examinations.

Summer 1995

Comstor *Chantilly, VA*

System Integrator. Constructed, configured, and tested personal computers.

1993 - 1995

WIN Laboratories, Ltd. *Manassas, VA*

Computer Technician. Conducted training sessions for Department of Social Services staff, technical user support, maintenance and support of large scale government contracts, assisted in installing local area networks. Extensive customer relations.

Summary of qualifications

C++, OWL, MFC, x86 Assembler, VB, VBA, LabView, Lisp, HTML, Perl, SQL
DOS, Win 3.1/95, Unix

Results Merging in the Networked Computer Science Technical Reports Library (NCSTRL)

A Thesis in TCC 402

Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science

by

Eric J Nelson
November 21, 1997

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in Humanities Courses.

Approved _____ Technical Advisor
James C French

Approved _____ TCC Advisor
Michael E Gorman

Abstract

In large digital libraries, results merging schemes based on relevancy to a given query are essential to providing a useful service. Documents estimated to be the most relevant should be the first presented to the user. The Networked Computer Science Technical Reports Library currently does not perform any relevancy estimation in its results merging. I have altered the dienst software that NCSTRL uses to test the feasibility of adding such a scheme. I found that a merging scheme based on relevancy can be added to the NCSTRL system with only basic modification. However, to provide a truly useful service more drastic changes need to be implemented within the current system.

Table of Contents

ABSTRACT	1
TABLE OF CONTENTS	5
INTRODUCTION	6
METHODS	9
DIENST AND NCSTRL.....	9
PROPOSED CHANGES	11
<i>Relevancy Using Title</i>	12
<i>Relevancy Using Abstract</i>	12
FINAL ANALYSIS	14
RESULTS	14
RECOMMENDATIONS.....	15
APPENDIX A - BIBLIOGRAPHY	17
APPENDIX B - GLOSSARY	18

Introduction

In the past, researchers in the field of Computer Science would keep abreast of concurrent research through technical journals. The papers published in these journals were typically based on research completed four to five years prior. In the rapid growth of the computer science field that is far too long to wait for current research. While conducting research, most computer scientists would produce technical reports online accessible through ftp sites or e-mail[1]. These technical reports, while current and up-to-date, were difficult to find and distributed over the vastness that is the Internet today. Fortunately, after several iterations of related systems there is the Networked Computer Science Technical Reports Library.

The Networked Computer Science Technical Reports Library is an international collection of computer science technical reports from CS departments and industrial and government research laboratories, made available for non-commercial and educational use[1]. The NCSTRL collection is distributed among a set of inter-operating servers operated by participating institutions[1]. The establishment of this library allows quick and easy access to otherwise inaccessible documents. The NCSTRL collection is constantly growing, frequently adding more repositories to the collection. As the number of repositories and the number of documents available in each grows, the current search engine becomes less and less efficient. Presently, it must search every site when a single query is submitted unless the sites searched is restricted by the user and uses only a simplistic ranking scheme for presenting results to the user.

As each of the sites continues to add documents to the distributed library, results from queries will begin to return a greater number of documents. Currently, a query may only return a small number of documents that the user can quickly sift through to find those that are helpful. Soon, an average query will return far too many documents for the user to manually determine the most useful document. A merging algorithm that ranks the documents based on estimated relevancy would be far more useful to the researcher posing the query than the current scheme.

NCSTRL is meant to provide quick and easy access to cutting edge research done at universities and research labs around the world. It must be reliable and useful to achieve that goal. Due to its relatively small size, at the present time, its current search engine is capable of meeting this goal. In the future,

however, changes must be made to keep up with NCSTRL's constant expansion. My thesis addresses the results merging algorithm of the NCSTRL search engine. I hope my efforts will result in appropriate and necessary changes.

This project will only immediately benefit those that are users of NCSTRL, computer scientists and researchers around the world. However, NCSTRL is designed to be a test bed for ideas and concepts that will effect the use and design of future digital libraries. My thesis will only extend the usefulness of a system designed to make current research available to computer scientists. Although NCSTRL is available to anyone with Internet access, its narrow scope does not benefit the general public directly. The only negative impacts that these improvements could have on the NCSTRL system would be if it was designed and implemented incorrectly. The merging algorithm would also need to return a more useful result. An incorrect merging algorithm would be no better and in fact worse than the scheme already in place.

I have designed and implemented a results merging strategy to augment the current search engine used for the Networked Computer Science Technical Reports Library. The results merging algorithm is based on the estimated relevancy of returned documents to the query.

At the present time, NCSTRL ranks the documents first by site with those sites containing the most returned documents first and then by date of submission to the library[2]. This does not address the relevance of a document in any way other than the fact that it matched the query. Much of my work in this area was based on meta-search engines used for the World Wide Web as a whole. Two examples, ProFusion and MetaCrawler, use a distributed set of web search services to perform meta-searches[5][6]. Both use such services as InfoSeek, Lycos, OpenText, WebCrawler, etc. A query posed to either ProFusion or MetaCrawler is then posed to each of the services and the distributed results are then merged and presented to the user. The theory is that the meta-results will be much more useful than any of the services alone by addressing a much broader section of the World Wide Web. Each also performs post-processing on the results from each service to provide a listing of URLs in order of estimated relevance. Much of the post-processing is also concerned with removing dead links and redundancy not just ranking URLs.

Applying the techniques used in already operating web meta-search engines will provide even better results than those achieved by ProFusion or MetaCrawler. The meta-search engines must contend

with a vastly heterogeneous environment. NCSTRL is a tightly controlled environment with a relatively singular purpose compared to the World Wide Web. In essence it is a homogeneous environment. Dead links and redundancy will be ignored as each document only appears once in the library and the sites are individually maintained. Also, a ranking scheme based on estimated ranking will be far more accurate than meta-search engines for the web for several reasons. Queries will not be hampered by discrepancies in natural language because of NCSTRL's focused environment. Each site within NCSTRL returns data in exactly the same format. This is one of the biggest problems faced by meta-search engines.

The merging algorithm must also be fast to the point of being transparent to the user. The speed with which results are returned directly effects the usability of those results. For these reasons several result merging algorithms were considered. The first uses only the currently returned information such as author, title, and institute to determine relevance. This takes no more time than the current strategy but allows for a better indication of relevancy. The second would retrieve and analyze the abstract for each document to determine relevancy to a query. This would take more time but would increase the likelihood of an accurate ranking. The final algorithm would use the entire document to determine relevancy. This, of course, is the most time consuming as well as the most accurate.

Methods

*Dienst and NCSTR*L

To accurately define what changes were made to the NCSTR

L search engine, I must first describe how the current system operates. While the changes made to the search engine are relatively small, it is necessary to understand the overall Dienst system to see how and why those changes were made. Dienst is a protocol and server for providing distributed document servers over the World Wide Web. Dienst is the foundation of the Networked Computer Science Technical Report Library. Dienst provides four basic digital library services: repository, index, search, contact, and user interface[2]. Repository services store and deliver multiple-format documents. Index services maintain searchable meta-information on the contents of the repository. Contact services provide interoperability between servers to facilitate a distributed library. User interface services provide a user front-end to the other three services.

The alterations to the results merging algorithm took place in the user interface portion of the dienst software. The dienst server, written in Perl, has several subroutines that handle the merging of results received from the other servers in NCSTR

L and the generation of an HTML page visible to the user via a web browser. These subroutines were altered to provide two new results merging algorithms.

A search on the NCSTR

L database requires an HTTP server and a Dienst server acting in concert. The HTTP server is a standard NCSA or CERN HTTP server with only a slight modification. The server is altered so that any URL going to a location containing "Dienst" is rerouted to a CGI stub. This CGI stub then communicates with the Dienst server using common socket I/O. This is all transparent to the user as it appears that he or she simply accessed an HTML file within a directory structure.

The particular search that I am concerned with is the fielded search that the Dienst server provides to the user. This is an HTML document with forms generated by the server to provide a flexible interface to the distributed database that is NCSTR

L. The user is presented the option of searching over author, title, or abstract. The user can also select the sites within the library to be searched as well as select that all sites be searched.

Currently, the results produced by such a search are returned with only a simplistic ranking scheme. Those institutions returning the most documents are listed first and within each institution the

documents are listed in order of submission to NCSTRL. A search over abstract with the keywords “NCSTRL” or “dienst” would return a list of hits grouped by university. It is possible that most useful document resides at a university that only has a few hits returned and as a result it is displayed at the bottom of the list. Most users would assume that the most desired document would be the one that contained both keywords not just one. However, in this case, the documents containing the most keywords are not listed at the top of the list. On a much broader search one would find a listing of hits that may have the most relevant document a significant distance down the list, many pages in fact. This is a detriment to the usability of the system.

Proposed Changes

The three alternate results merging algorithms that I considered were relevancy estimations using just the title of the document, using the abstract of the document, and the entire document itself. Using the title of the document is not very accurate but it is not network intensive. Using the abstract is much more accurate but is far more network intensive as the abstract must be retrieved for each document listed by the search. Using the entire document is the most network intensive and of course the most accurate.

Unfortunately, due to the current state of NCSTRL, analyzing the entire document does not seem possible at this time. Many documents are not currently available in their entirety, especially earlier documents. Also, when a full document is available it is rarely presented in a text format. Of note is the fact that many abstracts are not available. The remote servers only return documents that have abstracts in their associated bibliographic information when the library is searched over the abstract. For that reason, I altered the dienst server to only do analysis over the abstracts of documents when keywords are entered for the abstract field in the search form. Only when the title is searched over does the altered dienst server use the title to determine relevancy ranking.

The current dienst system is written entirely in Perl. I am concerned only with the user interface portion of the server and as a result, it is the only portion of the code that I familiarized myself with in great detail. All of my changes were made within a single file, `search.pl`. The current dienst software was written in Perl4 and the resources I have available to me at University of Virginia to perform HTTP requests are also for Perl4. For this reason, I made all of my changes with Perl4 constructs rather than the more recent Perl5.

In the fielded search form, the user has three options for searching the library. The user may search over author, title, or abstract to define a query. I have changed the dienst server to use different results merging algorithms based on the type of query the user enters. Searching over just author uses the merging scheme that is already in place. Searching over either title or abstract invokes one of two separate subroutines to determine the order in which hits are displayed to the user. The visible results are only slightly different from those used by the original. Hits from different institutions are no longer separated and so the institution name is displayed with each hit. These two algorithms are discussed below.

Relevancy Using Title

To estimate relevancy the dienst server simply counts the number of keyword matches in the titles of the documents. The hits are then sorted in reverse order of the number keywords each title contains.

The subroutine called when a query is posed over the title uses a loop based on the list of keywords. The keywords are the title portion of the query. Operations such as AND and OR are not considered. For each keyword, a nested loop checks each hit for the appearance of the keyword. A variable associated with each hit is incremented for each occurrence. After the outside loop has exhausted its supply of keywords, the list of hits is then sorted in reverse order over their keyword occurrences.

This algorithm is in fact faster than the results merging scheme already in place. The original scheme sorts over the number of hits returned by each institution and then performs a second sort over the date each document was submitted to the library. Determining relevancy over a title query requires only one sort and has the potential to return a more useful result.

The usefulness of searching over the title is very dependent on how the query is chosen. Most documents will never contain a word more than twice, so for each keyword a hit's associated count for that keyword will most likely be one. For queries where the keywords are ANDed together, this algorithm is virtually useless. The hits that are processed by this algorithm are guaranteed to contain every keyword at least once and as I mentioned before it is extremely rare that a title would contain the same word twice. The algorithm would determine that every document had equal relevancy to the query. On the other hand, determining relevancy over the title becomes fairly useful when the query's keywords are ORed together. In this case, each hit is guaranteed to have at least one keyword in the title not one of each. The hits would then be put into a more useful ordering. The hits that contain each of the keywords appear at the top of the list and those containing only one would appear at the bottom. When using the OR operator, coordination level sorting over the title becomes a quick and useful procedure.

Relevancy Using Abstract

The subroutine invoked when a query is posed over the abstract is very similar to the subroutine used for the title. The keyword matching, counting, and sorting are exactly the same. The difference is that instead of searching for keywords in the title, the abstract must be retrieved for the keyword matching.

Retrieving the abstract is somewhat network intensive so there is a significant lag time compared to the previous merging scheme. Each hit initially returned to the server contains a link to the bibliographic information for the document located on its remote server within the library. This bibliographic information usually contains an abstract for the document. The algorithm can then retrieve this generated HTML page that contains the abstract over the Internet. This means a dienst request for every hit matching the query. This is in addition to the dienst requests the server already made across the library to retrieve the hits matching the query. Obviously, the greater the number of hits the longer it will take to process the results.

Fortunately, this increase in time also brings about an increase in accuracy. The abstract for each document may contain each keyword more than once. An abstract that contains more occurrences of a keyword than another abstract is considered more relevant. This results in queries using the AND operator as being far more useful than when used in posing a query over the title. Queries using the OR operator remain more useful than those using only the AND operator for the same reasons as the title search. It is simple to see that the greater the number of keyword occurrences the more accurate the distribution of relevant hits.

Final Analysis

Results

I was successfully able to alter the dienst software to analyze either the title or abstract of a document to estimate relevancy to a given query. My results were as anticipated with a few exceptions. The relevancy estimation using the title worked just as expected. However, the estimation using the abstract had some problems due to the state of the digital library.

I altered the appearance of the results presented to the user. The original merging scheme presented the hits grouped by institution therefore each displayed hit did not contain a reference to its corresponding institution. I added the name of the institution to the hit as well as the number of keywords found in either the title or the abstract. The readily available keyword count allowed me to measure the accuracy of the new merging algorithm as well as get a sense of the distribution of estimated relevancy within a given query.

The relevancy estimation over title proved to be as quick as the current result merging scheme. It is too difficult to quantitatively measure the speed of the working alteration due to the variable latency involved with the Internet. To the user, results are almost immediate and are therefore readily usable. As mentioned in the previous section, queries using the OR operator have the potential to generate useful results while using the AND operator results in an arbitrary ranking. Searching over the title simply allows the user to distinguish between hits that pertain to all keywords, some keywords, or only one keyword.

Using the abstract provides a much more useful result to the user. Using either the AND or the OR operator yields a useful result. The abstract of a document is likely to contain several instances of a keyword. For the purposes of this thesis, estimated relevancy has been defined as simply the number of keywords found in a given space. An abstract with more keywords is deemed to be more relevant than another. This gives a very good distribution of hits rather than simply a grouping when using just the title. However, since each abstract must be retrieved over the Internet there is a significant time penalty. This penalty is enough to hamper the usefulness of such a merging scheme.

Unfortunately, there are other problems with searching over the abstract. When searching the NCSTRL library, it can be noticed that using the same keywords for title and then abstract can yield disparate results. This is due to the wording of the title and the abstract. The title will generally contain just a few keywords as to the subject of the document. The abstract may or may not contain those same words and the number of the those words is entirely up to the whim of the author. Only an analysis over the entire document can correctly estimate the relevancy to a query. Unfortunately, the current format of NCSTRL makes this very difficult and time consuming.

Currently, many sites in NCSTRL are not providing adequate support in retrieving the bibliographic information of documents. For the relevancy estimation over abstract, the abstract for each document must be retrieved. Some hits, while initially returned by an index server, do not have an abstract directly available at their associated site. This forces the algorithm to give such a hit a relevancy of zero in relation to the query.

Despite the problems mentioned above, the altered results merging scheme produces a more useful result than the original scheme. For any given query the first document listed contains the most keywords in either the title or the abstract depending on how the query was posed. Ideally, this would mean the first document is more likely to be the most useful document for the user if the query was posed correctly. As with search engines for the World Wide Web, how a query is posed directly effects the usefulness of the result to the user. Until semantic searching is fully realized, users will find themselves rewording queries to exact more useful results.

Recommendations

There are several areas where the results merging scheme of the NCSTRL search engine can be improved. The accuracy of the relevancy estimation can only increase by using the whole document for analysis. The time to perform the results merging can also be significantly decreased in two different ways.

Presently, very few documents contained within NCSTRL are available in ASCII text and for that reason I did not implement a merging scheme that uses the full document. The most common formats for documents are either Postscript or Portable Document Format. While far more difficult to perform than with ASCII text, both PS and PDF formats can be searched for keyword matching to implement full

document analysis for relevancy estimation. A merging algorithm can be implemented to identify the format of a document and then use the correct tools to perform keyword matching within the document. This would ultimately provide the most accurate results.

To provide a useful results merging scheme, the amount of time that it takes to analyze the abstract or full document must be significantly reduced. There are two readily apparent ways to speed up the merging process. The analysis of the abstracts or documents can be multitasked. The algorithm as it stands now retrieves an abstract, analyzes it, and then moves on to the next hit and its associated abstract. The time penalty is suffered while the system waits for the abstract to be retrieved. It is possible to allow the server to analyze other abstracts while an abstract is retrieved simply by initiating multiple processes. This way the server is not idle during the retrieval of abstracts or documents.

Secondly, the actual relevancy estimation can be done at each of the remote sites. Since NCSTRL is a homogeneous environment this would be fairly easy to design and implement and would not suffer the same problems as the meta-search engines of the Internet. The local server performing the merger of results could simply sort based on the relevancy ranking returned with the hit. This would be significantly faster than any other method and would yield the same results.

As NCSTRL grows in size both in the number of sites and the number of documents, changes will need to be made to continue provide the same level of service. A given query will return too many documents to not have a results merging scheme based on relevancy in place. This thesis has shown that the current dienst software can be readily altered to provide fundamental relevancy ranking for a given query. By incorporating a results merging scheme into the dienst software, NCSTRL can continue to provide a useful service as a digital library.

Appendix A - Bibliography

1. Networked Computer Science Technical Report home page. [Http://www.ncstrl.org/](http://www.ncstrl.org/). Maintained at Cornell Univerisity.
2. Davis, James. Sept '95. "Creating a Networked Computer Science Technical Report Library." *D-Lib Magazine*. Obtainable through NCSTRL.
3. Schatz, Bruce R. Jan '97. "Information Retrieval in Digital Libraries: Bringing Search to the Net." *Science*. 275:327-334
4. Salton, Gerard. Aug '91. "Developments in Automatic Text Retrieval." *Science*. 253:974-980.
5. Gauch, Susan. Wang, Guijin. "Information Fusion with ProFusion." *Proc. Of WebNet '96: The First World Conference of the Web Society*. October 1996.
6. Gauch, Susan. Wang, Guijun. Gomez, Mario. "ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines." *Journal of Universal Computer Science*.
7. Selburg, Erik. Etzioni, Oren. "Multi-Service Search and Comparison Using the MetaCrawler." *Proc. Of the 1995 World Wide Web Conference*. October 9, 1995.

Appendix B - Glossary

CGI - Common Gateway Interface

Dienst - a protocol and server for distributed document servers over the World Wide Web

HTML - Hypertext Markup Language

HTTP - Hypertext Transfer Protocol

MetaCrawler - A meta-search engine for the World Wide Web

NCSTRL - Networked Computer Science Technical Reports Library

PDF - Portable Document Format, a format for compressing and storing documents

Perl - A programming language ideal for string manipulation and we-based systems

Profusion - A meta-search engine for the World Wide Web

PS - Postscript, a format for compressing and storing documents

Selective Searching and Results Merging in the Networked Computer Science Technical Reports Library (NCSTRL)

A Proposal in TCC 401

Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science

by

Eric J Nelson
March 21, 1997

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid
as defined by the Honor Guidelines for Papers in Humanities Courses.

Approved _____ Technical Advisor
James C French

Approved _____ TCC Advisor
Michael E Gorman

Table of Contents

EXECUTIVE SUMMARY.....	22
RATIONALE AND OBJECTIVES	23
PRELIMINARY IMPACT STATEMENT.....	25
STATEMENT OF PROJECT ACTIVITIES	26
SCHEDULE	28
EXPECTED OUTCOMES.....	29
APPENDIX A - BUDGET AND EQUIPMENT	30
APPENDIX B - ANNOTATED BIBLIOGRAPHY	17
APPENDIX C - PERSONAL BIOGRAPHY	32
APPENDIX D - PRELIMINARY OUTLINE OF TECHNICAL REPORT	33

Executive Summary

I propose to design and implement a meta-searching strategy to augment the current search engine used for the Networked Computer Science Technical Reports Library. NCSTRL (pronounced "ancestral") is an international collection of computer science technical reports from CS departments and industrial and government research laboratories, made available for non-commercial and educational use [1]. NCSTRL uses the dienst protocol to access documents stored in the library over the World Wide Web. This strategy will address two aspects of NCSTRL's current search engine. First a results merging scheme will be developed to rank returned documents based on estimated relevancy rather than the current simple scheme. Secondly, a distributed search engine capable of selecting which sites are appropriate for a particular query will be developed. Both of these changes will results in a search engine that is more efficient and usable than the one currently in use by NCSTRL.

Rationale and Objectives

The Networked Computer Science Technical Reports Library is an international collection of computer science technical reports from CS departments and industrial and government research laboratories, made available for non-commercial and educational use[1]. The NCSTRL collection is distributed among a set of inter-operating servers operated by participating institutions[1]. The establishment of this library allows quick and easy access to otherwise inaccessible documents. The NCSTRL collection is constantly growing, frequently adding more repositories to the collection. As the number of repositories grows, the current search engine becomes less and less efficient. Presently, it must search every site when a single query is submitted unless the sites searched is restricted by the user and uses only a simplistic ranking scheme for presenting results to the user.

My first objective is to develop a selective search algorithm that would allow the search engine to eliminate those sites that definitely do not contain relevant documents from the search and thereby increasing efficiency. This would be transparent to the user who would simply indicate that they wanted all the documents that fit the query. A particular knowledge of the contents of each site would allow the engine to search only those sites that may contain documents pertaining to the query. Done correctly, the selection algorithm would return the same results as a query that did indeed search every repository in the collection. In the future, NCSTRL will be merged with other digital libraries and the number of repositories added to the collection will increase even further[2]. It will not be too long before a transparent selective search will be necessary. A typical user would have no need to specify which sites might contain relevant documents. The search engine would have the ability to determine which sites had the possibility of returning documents relevant to a particular query.

General queries posed to a search engine can generate a massive list of documents spread throughout the collection with most documents being of little use to the user[3]. Currently the search engine returns the documents to the user in order of sites containing the most relevant documents and in order within each site by the type of document and when it was submitted to the collection. My second objective would be to include a more substantial merging algorithm in the search engine would allow the

user to receive a listing of documents in order of their relevancy to the query[4]. A good merging algorithm would provide a more useful search for the user.

Preliminary Impact Statement

As the Networked Computer Science Technical Report Library grows, my proposed changes to the search engine become not so much an enhancement but a necessity. As the number of sites involved in the library grows the less efficient the search engine becomes. There will be more time-outs per search and each site will have to bear the load of each request. A search engine capable of appropriately selecting only those sites which contain relevant documents would be much more efficient than a search engine that performed a broad blanket search. Sites do not contain relevant documents to a query would be less likely to be asked to perform a search. A given search would require less work of the distributed servers than under the current search engine.

As each of the sites continue to add documents to the distributed library, results from queries will begin to return a greater number of documents. Currently, a query may only return a small number of documents that the user can quickly sift through to find those that are helpful. Soon, an average query will return far too many documents for the user to do this manually. A merging algorithm that ranks the documents based on estimated relevancy would be far more useful to the researcher posing the query than the current scheme.

NCSTRL is meant to provide quick and easy access to cutting edge research done at universities and research labs around the world. It must be reliable and useful achieve that goal. Due to its relatively small size at the present time it's current search engine is capable of meeting this goal. In the future, however, changes must be made to keep up with NCSTRL constant expansion. My thesis addresses two of different aspects of the search engine and I hope will result in appropriate and necessary changes.

The only negative impacts that these improvements could have on the NCSTRL system would be if it were designed and implemented incorrectly. Its correctness would depend on the fact that a selective search would return the same results as a distributed search that sent the query to every site associated within the library only more efficiently. The merging algorithm would also need to return a more useful result. An incorrect merging algorithm would be no better than the scheme already in place.

Statement of Project Activities

I propose to design and implement a meta-searching strategy to augment the current search engine used for the Networked Computer Science Technical Reports Library. This strategy contains two parts. First is a results merging algorithm based on the estimated relevancy of returned documents to the query. Second is a selective search algorithm that would attempt to eliminate unnecessary work done by the library for a particular query.

Currently I am researching ways to improve the current mode of presenting returned documents to the user. At the present time, NCSTRL ranks the documents first by site with those sites containing the most returned documents first and then by date of submission to the library[2]. This does not address the relevance of a document in any way other than the fact that it matched the query. Much of my work in this area will be based on meta-search engines used for the World Wide Web as a whole. Two examples, ProFusion and MetaCrawler, uses a distributed set of web search services to perform meta-searches[5][6]. Both use such services as InfoSeek, Lycos, OpenText, WebCrawler, etc. A query posed to either ProFusion or MetaCrawler is then posed to each of the services and the distributed results are then merged and presented to the user. The theory is that the meta-results will be much more useful than any of the services alone by addressing a much broader section of the World Wide Web. Each also performs post-processing on the results from each service to provide a listing of URLs in order of estimated relevance. Much of the post-processing is also concerned with removing dead links and redundancy not just ranking URLs.

Applying the techniques used in already operating web meta-search engines will provide even better results than those achieved by ProFusion or MetaCrawler. The meta-search engines must contend with a vastly heterogeneous environment. NCSTRL is a tightly controlled environment with a relatively singular purpose compared to the World Wide Web. In essence it is a homogeneous environment. Dead links and redundancy will be ignored as each document only appears once in the library and the sites are individually maintained. Also, a ranking scheme based on estimated ranking will be far more accurate than meta-search engines for the web for several reasons. Queries will not be hampered by discrepancies in

natural language because of NCSTRL's focused environment. Each site within NCSTRL returns data in exactly the same format. This is one of the biggest problems that meta-search engines must contend with.

A selective search algorithm will be much more difficult to base on preexisting search engines. Presently, SavvySearch performs a selective search in a distributed environment, but it uses domain analysis rather than a specific knowledge of the contents of each service[7]. Unfortunately, this does not lend itself to NCSTRL which has a single domain of computer science technical reports. What this algorithm will most likely require is a pre-knowledge of the sites to be searched. A data structure stored locally could store information about the contents of each site. From this information the search engine could then determine which sites could then be eliminated from a distributed search based on the particular needs of a query.

This aspect of my thesis overlaps with Ben Young's thesis concerning the selective dissemination of information in NCSTRL. An efficient selective search algorithm and a SDI for NCSTRL will more than likely use a similar data structure for selection. Information stored in this data structure would be such things as what authors are at which sites and keywords for both title and abstract. Authors has a much smaller domain than other field in a search on NCSTRL so it will be tackled first. Selection for other fields can then be based on the selection process for authors. Ben Young is currently analyzing the distribution of authors and keywords across the site of the whole library. Both the SDI's and the selective search algorithm's design and implementation will depend on the Mr. Young's initial results.

I will be pursuing this thesis under the technical advisement of Jim French. This topic falls into the realm of the Information Retrieval Group of which Jim French is the head. On top of weekly meetings with the IR group Jim French will advise me individually as to the direction and feasibility of my thesis. Also, a portion of my thesis will be partially based on the work done by Ben Young for his thesis which he is researching concurrently.

Schedule

March 13 - May 31

Acquire http and dienst server software

Set up a testbed

Develop proficiency in PERL

Determine format of information returned by each site

Determine if information is sufficient for results merging

If not determine what fields must be added

Develop ranking scheme based on info returned by each site

Implement merging algorithm in PERL

Test the merging algorithm

Iterate if algorithm insufficient

August 25 - September 15

At this time Ben Young should have completed his analysis of keyword

distribution across the library

Develop data structure to store distribution of authors across NCSTRL

Implement data structure and algorithm for selective searching

Test selective searching based on author

September 15 - September 29

Develop and implement selective searching for title and abstract

Test selective searching for title and abstract

October 1

Thesis introduction

October 6 - November 15

Finish testing changes to search engine

Possibly submit findings to NCSTRL for consideration

Technical report

Expected Outcomes

I fully anticipate developing a selective searching algorithm and a results merging scheme that could possibly be used with NCSTRL. Whether or not my findings are submitted to NCSTRL will be based on the quality of those findings to be determined by myself and my technical advisor, Jim French. It is left to the discretion of the administrators of NCSTRL to accept my changes to the current search engine. NCSTRL will eventually need to adopt strategies similar to those that I am researching for my thesis out of necessity.

Appendix A - Budget and Equipment

My expected costs for this thesis are zero. I will primarily be using preexisting equipment and software. The http and dienst software are free of charge and all development tools are provided by the UVA Computer Science department. As a student pursuing an undergraduate degree in Computer Science, I am performing the work related to this thesis pro bono.

Appendix B - Annotated Bibliography

1. Networked Computer Science Technical Report home page. [Http://www.ncstrl.org/](http://www.ncstrl.org/). Maintained at Cornell University.
This is the source for much of my information pertaining to the inner workings of NCSTRL.
2. Davis, James. Sept '95. "Creating a Networked Computer Science Technical Report Library." *D-Lib Magazine*. Obtainable through NCSTRL.
This article contributed even further information on how NCSTRL was developed.
3. Schatz, Bruce R. Jan '97. "Information Retrieval in Digital Libraries: Bringing Search to the Net." *Science*. 275:327-334
This article was the source for general knowledge in information retrieval.
4. Salton, Gerard. Aug '91. "Developments in Automatic Text Retrieval." *Science*. 253:974-980.
This article was also the source for general knowledge in information retrieval.
5. Gauch, Susan. Wang, Guijin. "Information Fusion with ProFusion." *Proc. Of WebNet '96: The First World Conference of the Web Society*. October 1996.
This paper outlined the idea of results merging in the heterogeneous World Wide Web and discussed the initial results of ProFusion. Results merging in NCSTRL should work in much the same way but should be more accurate do to NCSTRL's homogeneous nature.
6. Gauch, Susan. Wang, Guijun. Gomez, Mario. "ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines." *Journal of Universal Computer Science*.
This paper further discussed the ProFusion engine and the results obtained.
7. Selburg, Erik. Etzioni, Oren. "Multi-Service Search and Comparison Using the MetaCrawler." *Proc. Of the 1995 World Wide Web Conference*. October 9, 1995.
This paper discussed the problems associated with searching in a heterogeneous environment and their approach (MetaCrawler) to solving them.

Appendix C - Personal Biography

I am a fourth year undergraduate student in the School of Engineering at the University of Virginia. I am currently pursuing a degree in Computer Science and to that end I am proposing this thesis, Distributed Selection and Result Merging in the Networked Computer Science Technical Report Library. I have taken the relevant courses of Databases, Data Structures, and Networks. Although I have no previous experience in PERL, I have had no problem learning other programming languages especially as similar as PERL is to languages I already have experience with.

Appendix D - Preliminary Outline of Technical Report

- I. Title Page
- II. Table of Contents
- III. Introduction
 - A. What is NCSTRL
 - B. Why this was necessary
 - C. Literature review
 - D. What was done
 - E. Overview of Results
- IV. Results Merging
 - A. Relevant research
 - B. Strategies considered
 - C. Strategy developed and implemented
 - D. Test done and findings
- V. Distributed Searching
 - A. Relevant research
 - B. Strategies considered
 - C. Strategy developed and implemented
 - D. Test done and findings
- VI. Conclusion
 - A. Summary
 - B. Interpretations
 - C. Recommendations
- VII. Appendixes
 - A. Auxiliary and supplementary materials