Building a Digital Library for Computer Science Research: Technical Issues

Ian H. Witten, Craig G. Nevill-Manning and Sally Jo Cunningham

Department of Computer Science, University of Waikato,
Hamilton, New Zealand.

{ihw, cgn, sallyjo}@cs.waikato.ac.nz

Abstract

Technical reports are available electronically at hundreds of internet sites around the world. A major impediment to their utility in computer science research is the difficulty in locating reports that are relevant to a particular area. We describe the implementation of a digital library for computer science technical reports that indexes every word in each report, covers a majority of computer science technical report archives, and supports a variety of search types despite the fact that documents are not formally cataloged. We discuss in detail techniques for constructing the digital library that minimize Internet and local storage costs.

Keywords Digital libraries, information retrieval, data compression, user interfaces

1 Introduction

The New Zealand Digital Library for Computer Science Research is a pilot project designed to explore the potential of network-based digital libraries.[1] Currently, it provides access to 11,200 research documents worldwide (310,000 pages, 130 million words). Computer science is unique in that much high-quality information already exists in digital form and is freely accessible on the Internet. Because the time value of information is high, the field already relies more than most on pre-publication in the form of technical reports.

The present project is unusual in several respects. First, it provides a full-text index of the entire contents of each document, whereas other schemes index on user-supplied document descriptions, abstracts, or other document surrogates. Second, it makes a minimum of assumptions about conventions adopted by document repositories. Other schemes rely for their information on the index file that is present by convention in most ftp directories of technical reports, or on other information provided explicitly for indexing purposes. Third, close attention is paid to the interface and to the real needs of library users. Fourth, our work directly addresses the problem of building the library in a geographically remote location with high Internet costs—an environment in which the benefits of networked library technology are especially striking. Finally, our scheme is extremely economical in local disk resources.

To place the library in the context of similar efforts, Table 1 summarises existing technical report searching and indexing systems. The physics e-print archive, operating since 1991, has already supplanted journals and pre-print mailings as the primary information dissemination point for several areas of physics. Documents are submitted by e-mail in TeX format, along with bibliographic information which is used for indexing. NASA has a large index to publications since 1962, but most references are to documents that are only available in physical form. Computer Science is well endowed with indexes. The Harvest system contains a large number of documents, many of which are web pages rather than technical reports. The documents are indexed on limited information extracted from the original files. The DIENST system, a project involving 14 universities, is based on scanned images: indexing is performed on text produced by OCR. WATERS includes technical reports from 14 universities and provides bibliographic searches based on information provided by participating departments. UCSTRI allows searching of files of abstracts, which often appear in technical report ftp archives. While this information is limited in scope, the collection includes 185 sites and 11,000 documents.

This paper describes the technical issues that arose when constructing the New Zealand Digital Library. Witten *et al.* (1995) give a more general description of the project, and also discuss the support provided for bibliometric/scientometric studies of the computing literature and of library usage patterns. The next section reviews the system's overall structure. The following sections examine in some detail the three phases of library operation: collection, cataloguing, and retrieval of information. Then we discuss some planned enhancements that will improve access to the collection. Finally, we summarize the present state of the library and draw some conclusions.

---

1 http://www.cs.waikato.ac.nz/~nzdl

Proceedings of the 19th Australasian Computer Science Conference, Melbourne, Australia, January 31–February 2 1996.

Name	Domain	Reference	Sites	Docs	Indexed on
e-print archive	Physics	Ginsparg, 1994		–	user-supplied title, author, abstract
NTRS	NASA	Nelson, 1994		2 million	abstracts
UCSTRI	CS	van Heyningen, 1994	185	11 094	file of abstracts at ftp site
DIENST	CS	Davis, 1994	14	–	bibliographic information provided
WATERS	CS	Maly, 1994	16	–	citation and possibly abstract
HARVEST	CS	Bowman, 1994	297	37 476	limited info from several file types
NZDL	CS	this paper	170	11 205	full text from PostScript

Table 1: Technical report indexes on the Internet (– indicates that the figure is unavailable)

2 Structure of the Digital Library

The structure of the library can be viewed in terms of three processes that are fundamental to any library operation: collecting, cataloguing, and retrieval. Figure 1 shows the architecture that has emerged.

The collection phase is dominated by:

- the need to provide access to as many technical reports as possible, regardless of how or where they are stored;
- the cost of Internet transfers to New Zealand.

It was resolved that the system should not require any effort on the part of participating technical report repositories, and indeed these information providers would not generally be aware of their inclusion in our index. No special software, archive organizations, or file formats are required of the providers. The system design allows for any format from which ASCII text can be extracted, for example .dvi files, rtf files, document images (for which OCR would be used to extract the text), or PostScript. From an initial investigation it appeared that the use of PostScript files was pervasive in computer science technical report archives, so the system currently deals only with this format.

To reduce Internet costs and local storage requirements, it was decided that the New Zealand site would hold only an index and search engine, the documents themselves remaining in their original repositories. It also proved expedient to retain a full copy of the text (the ASCII text, not the PostScript) of the document locally: this forms the basis for the index and is useful in its own right for browsing. Moreover, the text of the collection provides an excellent foundation for bibliometric research. Finally, a facsimile image of each document's first page or two is retained locally so that users can sample the correctly formatted original.

The cataloguing phase is hamstrung by the fact that we expect no information to be supplied by participating repositories apart from the actual document text itself. None of the standard bibliographic information—author, title, date, abstract, etc.—is available. Full-text search of the contents of the documents seems to be the only viable access mechanism. Consequently an early decision was made to index the entire contents of the documents, rather than restrict access to file information or title/author/abstract summaries.

We decided to use a standard public-domain indexing and search engine, MG (Witten *et al.*, 1994). In order to limit the scope of the project, we resolved not to modify the MG system at all. To provide the kinds of search that we feel library users need, we were forced to index multiple versions of the document collection—a somewhat ironic development considering the attention that MG pays to compression, but nevertheless a pragmatic decision that has allowed us to get the library off the ground very quickly.

The retrieval phase is driven by an overriding desire to make the system widely available over the Internet. This forced us to abandon the standard X interface to MG and build one based on the World Wide Web. While considerably less functionality can be supported, universality of access is paramount in this application.

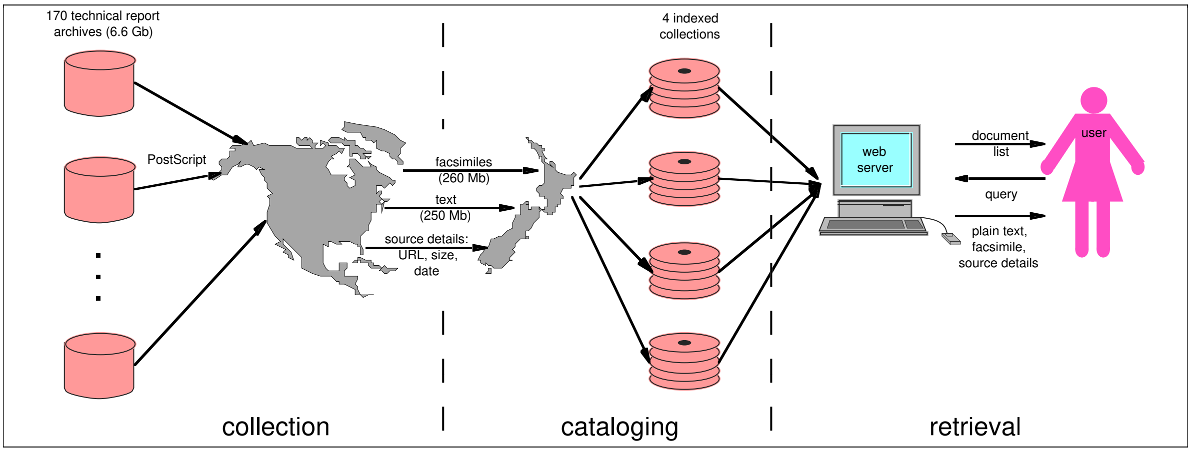

Figure 1: Architecture of the digital library

Source documents

Sites	170	
Reports	11 205	66 reports per site
Original PostScript	6.6 Gb	590 Kb per report
compressed with gzip	2.3 Gb	205 Kb per report

Extraction process	*total*	*per report*
Text	708 Mb	63 kb
	132 million words	12 000 words
	310 671 pages	28 pages
compressed with gzip	227 Mb	20 kb
First page facsimiles	24 656 pages	2.2 pages
	276 Mb	24 kb
Times (elapsed)		
download compressed PostScript	40 hours	13 seconds
extract text	40 hours	13 seconds
render first-page image	62 hours	20 seconds
transmit text and image to NZ	25 hours	8 seconds

Indexed collection

Text			228 Mb
Index	*Indexed by*	*Stemming and case folding*	
	report	yes	29 Mb
	report	no	39 Mb
	page	yes	66 Mb
	first page only	no	4 Mb
First-page images			276 Mb
Total			642 Mb

Table 2: Current size of the digital library

3 Collecting: Gathering the Information

The lowest common denominator for representing information in conventional libraries is paper, and many digital library efforts involve scanned paper documents (e.g. Crocca & Anderson, 1995, Van House, 1995, Davis & Lagoze, 1994). At the outset we resolved to avoid any scanning and base the digital library on machine-readable text. In the world of electronic information, PostScript—rather than plain, unformatted text—seems to be the closest analog to paper as a document storage medium. The current size of the library is summarised in the top part of Table 2. It provides access to 6.6 Gb of PostScript files containing technical reports in the computer science area.

One of the motivations for a New Zealand digital library is efficient use of the expensive Internet link to North America. Allowing computer scientists to seek technical reports and preview them locally before downloading the full PostScript file encourages exploration without concern for network charges. However, to build the full-text index it is necessary to examine the contents of each report. Transmitting all of them to New Zealand for indexing would cost many thousands of dollars, and negate any cost benefits the project might offer.

For this reason, we use a distributed scheme to create the index. In the first stage, a computer in North America (provided by the Department of Computer Science at Calgary in Canada) downloads each technical report, extracts the facsimile images of the first page or two and the full text of the entire document, sends them to New Zealand, and deletes the report. This process is shown at the left of Figure 1. Figure 2 illustrates (at the top) an original document in PostScript form, and (at the bottom) the two files extracted from it: on the left the full ASCII text, and on the right the facsimile image of the first page.

This operation, which is done at night to reduce its impact on network and machine performance in North America, significantly reduces the transmission cost on the Pacific link. The 6.6 Gb of uncompressed PostScript currently indexed by the library is actually stored in repositories in compressed form, occupying approximately 2.3 Gb. In this form, each report takes about 13 seconds to download to the North American site, whereas it would take have taken about 36 seconds to download to New Zealand because of the low bandwidth of the current Pacific link. In actuality, only 708 Mb of uncompressed text, which reduces to 227 Mb when compressed, needs to be transmitted,

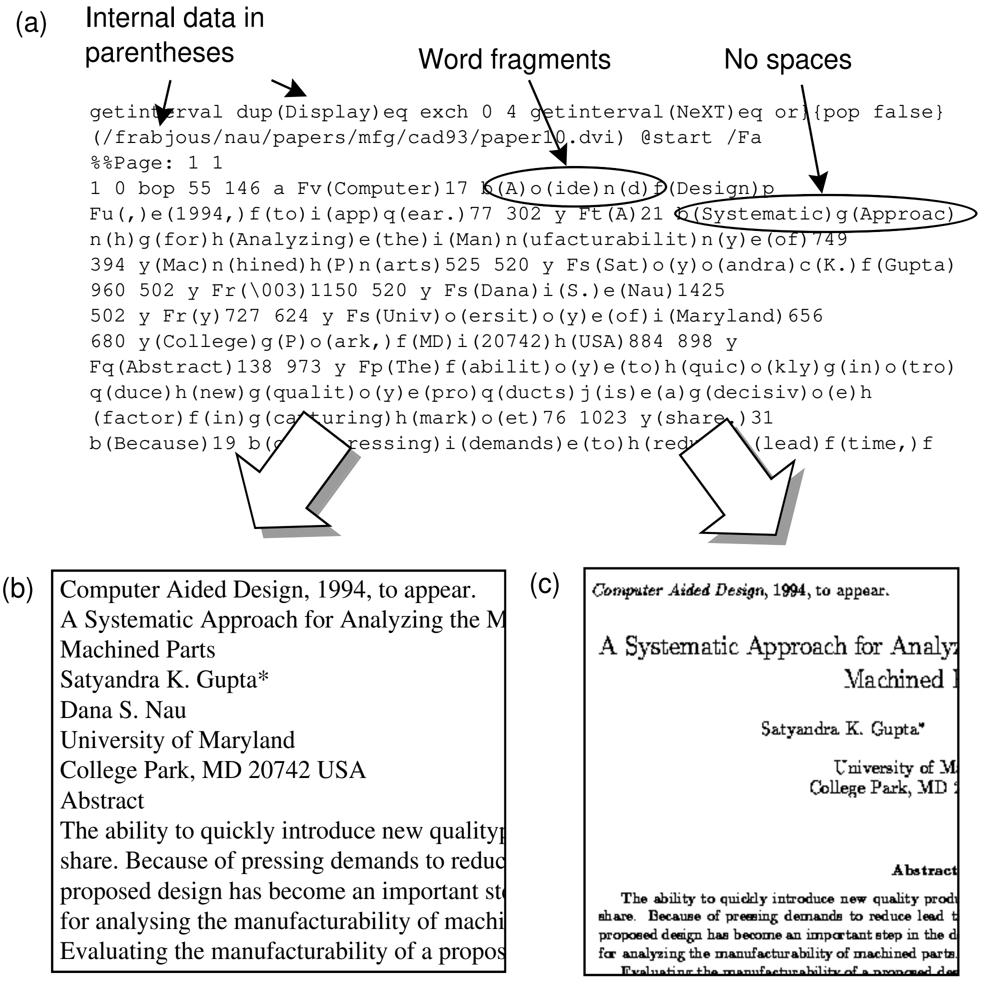

Figure 2: Conversion from PostScript: a PostScript file, the raw text extracted from it, and a facsimile page

which takes about 4 seconds per report. Added to this is the cost of transmitting facsimile images: 280 Mb in total, or another 4 seconds per report. The centre block of Table 2 summarises these figures.

To find archives of technical reports, we use several lists maintained on the Internet, including Blythe (1995) and Harris (1995), and recursively descend the directory hierarchy looking for PostScript files. Each file is downloaded, along with its size and date, and the text and facsimile images are extracted.

3.1 Text extraction

While the words of a technical report usually appear as plain text within a PostScript file, they are thoroughly intermixed with PostScript language commands and internal data. Words appear within parentheses, but so does internal information such as font names and error messages. Spaces are not explicit, but are coded implicitly in terms of the placement of words on the page. Finally, whole words are not always bracketed together: to give greater control over spacing, letters and word fragments are often placed individually. For these reasons, text cannot be extracted reliably by syntactic analysis of the PostScript file. Figure 2a illustrates some of these problems: commands and internal data in parentheses, just like document text; the word *Aided* broken into three fragments; and the lack of spaces between words.

Our solution is to prepend a PostScript prologue that redefines the operators responsible for placing text on the page, and process the resulting file by a PostScript interpreter. The redefined operators write the text fragments to a file, taking note of their position on the page in order to insert spaces and new lines where appropriate. This technique is based on part of the GhostScript distribution (the *ps2ascii* program), but is simpler and far more robust in terms of the number of different PostScript files it can handle (in fact, *ps2ascii* failed on 90 out of a sample of 143 reports that we downloaded).

Figure 3 shows the complete PostScript prologue used for this process. It involves an absolute specification of the minimum inter-word spacing (currently set to 5 points) and inter-line spacing (currently 10 points)—crude heuristics which could probably be improved significantly but seem to work well in practice. This process not only extracts the body of the report, but also any text in figures and tables. It takes 13 elapsed seconds per report (on a Sun SparcStation 10).

```
100 dict begin                                     new dictionary for operator redefinitions
/extract_file outfile (w) file def                 open the output file
/extract_x 0 def /extract_y 0 def                  the last location of the current point

/simple {
     currentpoint                                  put the current point on the stack
     extract_y sub abs line_space gt               if the current point has moved up or down,
     { extract_file (\n) writestring } if               output a new line
     extract_x sub space_width gt                  if the current point has moved right,
     { extract_file ( )  writestring } if               output a space
     extract_file 2 index writestring              write the string to the file
     systemdict begin cvx exec end                 show the text to update current point
     currentpoint /extract_y exch def              remember the current point for next time
     /extract_x exch def
} def

/show       { /show       simple } def             for each variation of show, call simple
/ashow      { /ashow      simple } def                  with the original operator on the stack
/awidthshow { /awidthshow simple } def
/widthshow  { /widthshow  simple } def
/kshow      { /kshow      simple } def

/showpage   { extract_file (\nNZDL_NEW_PAGE\n)     record the page boundaries
writestring } def
```

Figure 3: The prologue used to extract text from PostScript documents

3.2 First-page facsimile

The facsimile image of the first page(s) shows the user the beginning of the document as it actually appears, complementing the raw text of the full paper. This provides an analogous service to a library supplying users with fax copies of the first page or two of a document.

One approach is to store the PostScript version of the first page. However, PostScript documents invariably contain a lengthy prologue that defines fonts and special functions. This prologue is essential to interpret the first page, and it turns out that in most cases the size of the file required to reproduce the first page is nearly as large as the entire document. Consequently we decided to store the facsimile as a screen-resolution bitmap, so that the file size is bounded by the size of a standard page. The facsimile images are produced by a PostScript interpreter as a 75 dpi bitmap, and are saved as graphics files in GIF format. The average size of a one-page facsimile is 11 kb, and its production takes about 20 seconds.

Unfortunately, the first page of many technical reports shows little more than an institutional logo along with the title and authors' names. The second page may also give little information. In order to ensure that a page containing some content is imaged, we estimate the content of a page by the size of the GIF file that represents it. Because these images are compressed, pages with large areas of white space and big letters compress well, to a few kb, while pages full of actual text occupy over ten kb. The extraction system renders pages in sequence until the sum of the file sizes exceeds 10 kb. This simple heuristic seems to work well.

4 Cataloging: Building the Database

The search engine for our digital library is the public-domain system MG. Tailored for highly efficient storage of text, MG can store a full-text index to a large collection of text in only 5% of the size of the original text (Witten *et al.*, 1994). Further, it provides exceptional response time in processing queries: experiments with the 750,000 document TREC collection produce ranked output for queries of forty to fifty terms within three to five seconds.

4.1 Types of search

MG supports the usual Boolean and ranked keyword searches over the full text of the document. Since the library has no access to formal cataloging information for the technical reports, it cannot provide descriptive field searching (such as author, title, and publication date). Moreover, MG does not support other common types of search, such as within-page searching or term truncation, unless the database is specifically constructed for them. We implement several different kinds of search as follows.

Author/title. In the vast majority of reports, the first page gives bibliographic information such as title, author, author's institution, etc. By limiting attention to this first page, the user can approximate a search based on this type of information. For example, an initial page search for documents authored by *Knuth* will not retrieve documents that merely cite his work. To support such searches, we maintain a separate index on the first page of each of the documents in the collection.

Publication date. Again, since most reports list the date of publication on the initial page of the document, a publication date search can be approximated by a first-page-only search. The MG search syntax does not support date range searching directly (e.g. *1992–1994*), but ranges can be specified by OR-ing the desired dates (*1992 OR 1993 OR 1994*).

We plan to simulate publication date search by permitting the user to search on the date in which a technical report was entered into its repository. Several repositories are digitizing their older paper reports, and so this type of search is likely to produce uneven results, because the timestamp only records the date that the report was stored and not the date on which it was originally produced. However, we expect searches on file storage dates to become more accurate as the repositories catch up on their retrospective conversion.

Page searching. The digital library stores the full text of technical reports, and supports searching over the complete document text. This is very useful in performing very general searches with high recall, that is, ones that retrieve a high proportion of the relevant documents in the collection. However, a large number of irrelevant documents ("false drops") can be expected as well. For example, in searching for *information retrieval* we could expect that many technical reports would contain both of those terms somewhere in the document, perhaps widely separated. The greater the physical distance between the two words, the less likely that it is about the subject *information retrieval*.

We support proximity searching—permitting the user to narrow the acceptable distance between terms—by an option requiring the query terms to appear on the same physical page of the document. Since MG does not store the location of each term within a document, we process these queries against an additional index which treats each page of each document as a separate item. Phrase searching is implemented within MG by post-processing query results; a string search for phrases can be performed on documents returned by any query.

Case folding/truncation/exact match. Retrieval systems commonly allow the user to decide whether or not the query terms should be exactly matched in the document returned. Exact matches are useful, if not crucial, in constructing some types of search: for example, in locating the author *Gray* rather than the color *gray*, or for finding documents about *NeXT* computers or the *SMART* system. On the other hand, case folding is required in order to avoid artificial distinctions between capitalized and non-capitalized forms of the same word (such as *Information* and *information*). Similarly, it is sometimes useful to retrieve all possible forms of a given word (*retrieval*, *retrieve*, *retrieving*). These truncated terms of the root word should be sought automatically, without forcing the user to list all alternatives.

To permit exact match searching, we index the document collection as is. We also maintain a separate index of the collection with case folding and term stemming enforced.

4.2 Size of collection and indexes

The lower block of Table 2 summarizes the size of the document collection and the storage requirements for the MG indexes that comprise the library. As discussed above, we maintain four separate indexes on the document collection: indexing by individual technical reports, both with and without case folding and stemming; indexing by individual pages of reports; and indexing only the first pages of reports. While these separate indexes obviously increase the storage requirements of our system, they also support a wider range of search options for users. The total space occupied by the four indexes is 138 Mb, a mere 2% of the total size of the PostScript files that are indexed. The text of the technical reports occupies 228 Mb when compressed by MG, giving a total of 366 Mb for the text and index.[2] Finally, the first-page facsimiles (centre block of Table 2) occupy an additional 276 Mb, giving a total disk storage requirement of 640 Mb.

4.3 Collection maintenance

All that is necessary to maintain the library's collection is to ensure that the documents indexed actually continue to exist. This can be checked by periodically examining the technical report repositories for changes and updating the collection accordingly.

UCSTRI, another technical report server that stores only the document index and pointers to the documents themselves, reports frequent maintenance problems caused by changes in the technical report repositories that it indexes—for example, files being renamed or removed from the collection (Van Heyningen, 1994). However, it builds its index from summary files that are present by convention in most ftp directories of technical reports, and many problems inevitably arise from new or altered formats for the document list on which UCSTRI relies for its bibliographic information.

One source of growth for the collection is when new documents in known repositories are located and indexed during a routine examination of currently indexed sites for the purpose of collection maintenance. Additionally, new sites can be detected by various means: monitoring lists such as those compiled by Blythe (1995) and Harris (1995) for new additions; manually scanning the newsgroups that announce new technical report lists; and encouraging users to email suggested new sites to a central coordinator.

However, an all-inclusive information collection policy is basically unscalable and will become

---

2 Honesty compels us to confess that in the current implementation, the compressed text is in fact duplicated for each of the four indexes. A small change to the system will remove this redundancy.

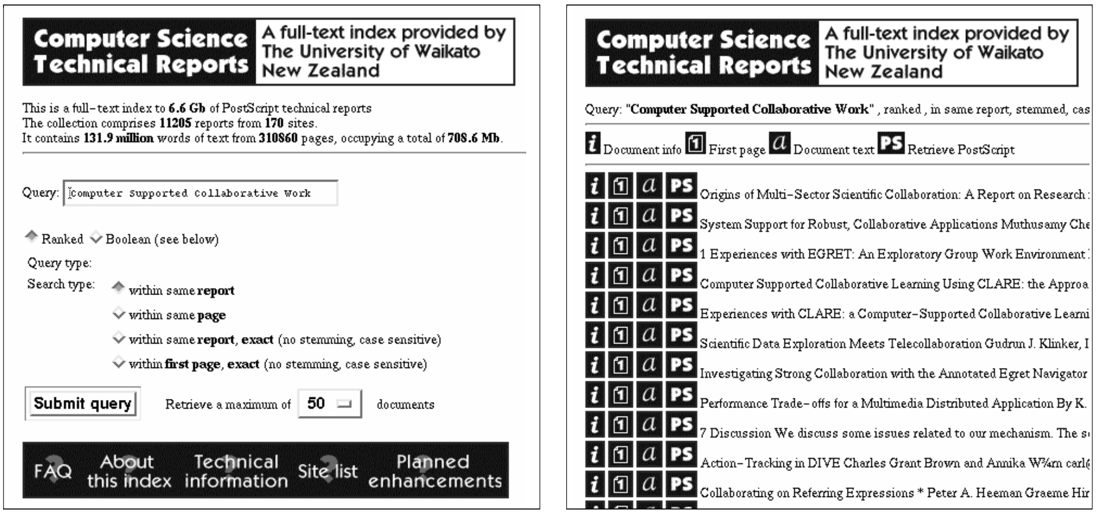

Figure 4: Interface to the digital library: the query page and a typical response

infeasible as the number and size of technical report repositories grows. We are examining several techniques for culling the collection when that becomes necessary. One possibility is to monitor every participating user's access to technical reports, and note the sites that see the least use. These sites are prime candidates for removal when the size of the collection becomes unmanageable. This means that the rate of growth of the collection, and hence the resources it consumes, is governed by the size, diversity, and level of activity of the user population rather than by rate of growth of the bibliographic universe.

Another technique is to use the characteristics of the documents themselves to determine what to delete from the collection. Technical reports typically are most useful shortly after their publication: in computer science in particular, they tend to receive half of their citations within two years, and the rate of citation falls off rapidly as time passes (Cunningham, 1995). Given this usage pattern, one might consider eliminating the more elderly reports, as they are more likely to be obsolete.

5 Retrieving: The Query Interface

The World Wide Web is the the distribution medium for the digital library. Figure 4 shows the query page and a typical response.

The MG system already includes an X-windows interface, called *xmg*, which allows document collections to be queried interactively, and it is instructive to review the changes that were necessary to recast this as a WWW service. *Xmg* provides a main window that contains query options, the query itself, the document list resulting from the query, and one of the returned documents. Moreover, it provides multiple windows within which returned documents can be browsed and compared.

The http protocol forces a more spartan interface. Because of its page-oriented nature, the document list, query options, and returned documents are placed on separate pages. The maximum number of matches returned for a ranked query is a potential transmission bottleneck, for it is easy to issue a query that returns many thousands of documents. This is initially set to 40, which gives good performance over the Internet, and can be raised to 200 (but no higher) by the user. Because WWW transactions are stateless and the user is unknown to the server, it is impossible to provide services such as remembering queries, displaying a history, or saving profiles without requiring users to identify themselves explicitly.

Another problem arises from the need to share the query engine between users. MG was not designed to deal with simultaneous multi-user access, and while it is possible to run a separate copy of the process for each query, the startup overhead makes this very inefficient. We therefore run one copy of *mgquery* continually for each of the four collections, and additional software provides exclusive access to the query process for the duration of a query.

6 Planned Enhancements

The library is an early prototype and several simple enhancements are planned.

It is hard to expect the user to choose between document- and page-level retrieval because it is not clear what the basis of the choice should be. It may be that the page-level index suffices for all retrieval, and the document-level one should be abandoned. Perhaps the problem can be solved by a more sophisticated information display mechanism: for example, TileBars (Hearst, 1995) are a convenient way of visualising the distribution of query terms throughout a document that should be easy to integrate into our system. A number of other searching mechanisms would be very useful. Mixed Boolean and ranked queries would allow users greater control over the documents that are retrieved. Searching by report date was discussed above but is not actually implemented. Browsing by location needs some human pre-processing to identify machine locations by institution rather than by network address. Even simpler, and also useful, would be to allow users to look at the directory that a particular report comes from, because sometimes multi-part documents are stored as separate files. Finally, co-location queries allow users to examine which words occur together; this is useful for textual analysis purposes but would have to be implemented by a post-retrieval scan.

The collection would be enhanced considerably if reliable heuristics could be devised for determining bibliographic details (title, author, date, institution) of the reports so that a proper catalogue could be built. Moreover, the text collection provides an ideal basis for bibliographic studies in computer science. If it were possible to devise heuristics to locate the references section within a document, a separate references database could be constructed that would be extremely useful for citation analysis.

At a lower level, the PostScript conversion process could be improved in many ways. The use of absolute constants for the minimum inter-word and inter-line spacing is clearly a potential source of problems (although in fact we have rarely noticed them in practice). Ligatures like *ff*, *fi*, *fl*, *ffi*, and *ffl* are represented as single PostScript characters, and different word-processing programs use different conventions for coding them; we do not yet have anything like a complete list. The same goes for Greek letters. The text often looks messy because items like mathematical formulae come out as strange characters; the result would be improved if there were a mechanism to detect this and replace formulae by some standard text like *<MATH>*. A significant improvement would be to apply heuristics to detect paragraph boundaries so that the text could re-paragraphed automatically by the WWW server to fit the user's current window size. Finally, PostScript documents are sometimes stored in reverse page order (so that they come out the right way round in the printer). We have found no reliable way of telling that they are backwards; again, a heuristic solution seems inevitable.

7 Status

The library went online in May, 1995, with an initial collection of approximately 2000 documents. Although it has been publicized only within the seven New Zealand universities, the library has been accessed from 40 sites—28 of them overseas. The current implementation is clearly both useful and seeing use, even as a prototype.

8 Conclusion

A prototype digital library has been constructed which provides access to 11,200 documents worldwide in computer science. Unique in its full-text index to all documents, it has already demonstrated potential as a research tool. The material in the library is distributed globally: the current collection amounts to 6.6 Gb of PostScript files. The volume of information stored centrally is 640 Mb, 10% of the collection size. Of this, approximately one-fifth (2% of collection size) comprises a full-text index, two-fifths (4%) stores facsimile images of the first page or two of each report, and two-fifths (4%) records the plain ASCII text of all reports. Because reports were downloaded and processed by a North American site, only the images and plain text had to be transmitted across the Pacific.

The concept is by no means limited to PostScript files. Any file format will do so long as it is possible to extract the plain text from it for indexing purposes. For example, many sites are scanning old reports and storing them as page images; these can be accommodated within the library by OCR-ing them for indexing purposes. The inevitable OCR errors will reduce the quality of the index, but this can be ameliorated by using ranked queries containing redundant terms.

This library provides a facility for accessing the "grey literature" contained in technical reports in the field of computer science, a field which—because the time value of information is high—relies more than most on pre-publication in the form of reports. Because of the extremely rapid rate of change in the Internet, the scheme is designed to perform a useful job in the short term, as well as lay the foundation for full-text access to substantial document collections in the future. It represents an innovative investigation into how digital libraries might benefit small, geographically isolated communities, and counter the diseconomies of scale from which they suffer in a world of exponentially growing information. The system can provide excellent response because the index is kept locally. This encourages browsing, and protects users from variable network loading and remote machine downtime. The system also provides a source of detailed statistics on information retrieval and usage by a small research community, and a platform for research on the characteristics of the computing literature as a whole.

The project shows one way of dealing with the new reality of Internet publishing. Making a minimum of assumptions, our digital library imposes structure on a fundamentally anarchic, uncatalogued, system, giving information consumers a tool to find the information they need.

Acknowledgments

We would like to acknowledge the support of the New Zealand Lottery Grants Board for this project. Thanks to Mike Vallabh and Brent Summers for technical assistance, and to Tim Bell and Mark Apperley for encouragement and advice. Bruce McKenzie provided the prototype WWW interface to MG.

References

[1] J. Blythe. On-line CS Tech reports. http://www.cs.cmu.edu:8001/afs/cs.cmu.edu/user/ jblythe/Mosaic/cs-reports.html.

[2] C.M. Bowman, P. Danzig, U. Manber and M.F. Schwartz. Scalable Internet resource discovery: Research problems and approaches. *Communications of the ACM 37*(8), pages 98–107, 1994.

[3] W.T. Crocca and W.L. Anderson. Delivering Technology for digital libraries: experiences as vendors. *Proc. Digital Libraries '95*, pages 1–8, 1995.

[4] S.J. Cunningham, N. Empson and R. Kamau. Bibliomania: what can we learn from the research literature?*Proc. New Zealand Computer Society Conference*, 1995.

[5] J. Davis and C. Lagoze. A protocol and server for a distributed digital technical report library. Technical Report 94-1418, Computer Science Department, Cornell University, 1994.

[6] P. Ginsparg. First steps towards electronic research communication, *Computers in Physics 8*(4), p. 390–401, 1994.

[7] R. Harris. Computer Science Technical Reports Archive Sites. ftp://rdt.monash.edu.au/pub/techreports/sites/sites-list-data.

[8] M.A. Hearst. TileBars: visualization of term distribution information in full text information access. *Proc. CHI'95*, pages 56–66; May, 1995.

[9] K. Maly, E.A. Fox, J.C. French and A.L. Selman. Wide area technical report server. Technical Report , Dept. of Computer Science, Old Dominion University, 1994.

[10] M.L. Nelson, G.L. Gottlich and D.J. Bianco. World Wide Web implementation of the Langley Technical Report Server. *NASA Technical Memorandum 109162*, Langley Research Center, Hampton, Virginia, 1994.

[11] N.A. Van House. User needs assessment and evaluation for the UC Berkeley electronic environmental library project: a preliminary report. *Proc. Digital Libraries '95*, pages 71–76, 1995.

[12] M. VanHeyningen. The Unified Computer Science Technical Report Index: Lessons in indexing diverse resources. *Proc. Second International WWW Conference*, Chicago, 1994.

[13] I.H. Witten, A. Moffat and T.C. Bell. *Managing Gigabytes: Compressing and indexing documents and images*. Van Nostrand Reinhold, New York, 1994.

[14] I.H. Witten, S.J. Cunningham, M. Vallabh and T.C. Bell. A New Zealand digital library for computer science research. *Proc. Digital Libraries '95*, 25–30, Austin, Texas, June, 1995.