Working Paper Series
ISSN 1170-487X

Building a Public Digital Library Based on Full-Text Retrieval

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

Working Paper 95/24

August 1995

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Building a Public Digital Library Based on Full-Text Retrieval

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

Phone (+64) 7 838–4246, fax (+64) 7 838–4155

Abstract *Digital libraries are expensive to create and maintain, and generally restricted to a particular corporation or group of paying subscribers. While many indexes to the World Wide Web are freely available, the quality of what is indexed is extremely uneven. The digital analog of a public library—a reliable, quality, community service—has yet to appear. This paper demonstrates the feasibility of a cost-effective collection of high-quality public-domain information, available free over the Internet.*

One obstacle to the creation of a digital library is the difficulty of providing formal cataloguing information. Without a title, author and subject database it seems hard to offer the searching facilities normally available in physical libraries. Full-text retrieval provides a way of approximating these services without a concomitant investment of resources. A second is the problem of finding a suitable corpus of material. Computer science research reports form the focus of our prototype implementation. These constitute a large body of high-quality public-domain documents. Given such a corpus, a third issue becomes the question of obtaining both plain text for indexing, and page images for readability. Typesetting formats such as PostScript provide some of the benefits of libraries scanned from paper documents—such as page-based indexing and viewing—without the physical demands and error-prone nature of scanning and optical character recognition. However, until recently the difficulty of extracting text from PostScript seems to have encouraged indexing on plain-text abstracts or bibliographic information provided by authors. We have developed a new technique that overcomes the problem.

This paper describes the architecture, the indexing, collection and maintenance processes, and the retrieval interface, to a prototype public digital library.

Keywords *Digital libraries, information retrieval, full-text retrieval, distributed databases, data compression, user interfaces*

Building a Public Digital Library Based on Full-Text Retrieval

Creating a publicly-available digital library presents some interesting challenges, particularly if the library is intended for serious professional work rather than casual browsing. First, the raw material: the collection must comprise text that has been placed in the public domain. Second, the selection of material: although a huge amount of public-domain text is present on the Internet, its quality is extremely uneven and only a tiny fraction is appropriate for inclusion in a library. Third, the format of material: it is helpful if the collection resembles the traditional form of typeset pages rather than raw electronic text. Fourth, cataloguing: appropriate bibliographic information in a usable format may be difficult to find and onerous to provide manually. Fifth, information retrieval: a uniform, easy-to-use, publicly-accessible interface is necessary.

This paper describes a prototype digital library that addresses these issues. In order to focus the collection so that the library can become a serious research tool, we have chosen the domain of computer science technical reports. Much high-quality information already exists in digital form and is freely accessible on the Internet, and because the time value of information is high, computer science already relies more than most on pre-publication in the form of technical reports. Although others have built collections of computer science technical reports (see sidebar), the present project is unique 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 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, the scheme is extremely economical in local disk resources.

The New Zealand Digital Library for Computer Science Research currently provides access to 11,000 research documents on 170 archives worldwide (310,000 pages, 130 million words). This corresponds to 6.6 Gb of PostScript files containing technical reports in the computer science area; some relevant statistics are given in the upper block of Table 1. The library went online in May 1995 with an initial collection of approximately 2000 documents. Although until recently it has been publicized only within New Zealand universities, the library has been accessed from over 200 sites worldwide. The current implementation is clearly both useful and seeing use, even as a prototype. User feedback is overwhelmingly favorable.

INDEXING AND RETRIEVAL

The key innovation in our work is that it dispenses with the traditional library catalogue, which relies on information about each item in the collection being supplied along with the item itself. If information providers must supply such information, the scope of the collection is thereby restricted. The alternative, cataloguing by library staff, means that the collection is limited by human resources. It was resolved that although the collection would be formed entirely from publicly-available repositories of text, the system should not require any effort on the part of participating repositories; indeed there should be no need for these information providers to be aware of their inclusion in the index. No special software, archive organizations, or file formats are required of the providers. The only information used for cataloguing is derived from the documents themselves.

Full-text index

These considerations led to the use of a full-text index instead of a library catalog as the main retrieval mechanism. The index makes the entire text of all documents available for retrieval, rather than some much more restricted list of keywords as is the case in many computer-based text retrieval systems. From the point of view of building the system, this has the advantage that nothing more is needed to construct the index than the plain text of the documents in the library: there is no requirement for traditional bibliographic database information such as author, title, publisher, and so on. From the point of view of the user, it provides a powerful tool for searching for information: we consider below whether this can supplant the more traditional forms of library access—by author, title, date, subject, and so on.

The search engine for the library is the public-domain system MG. Tailored for highly efficient storage of full-text databases, MG can pack an index to a large collection of text into only 5% of the size of the original text [1]. Further, it responds rapidly to queries: experiments with the 750,000 document TREC collection produce ranked output for queries of forty to fifty terms within three to five seconds.

Types of search

MG supports the usual Boolean and ranked keyword searches over the full text of the document. It allows other common types of search, such as within-page searching or term truncation, only if the database is specifically constructed for them. To offer the facilities that library users need, the system creates several different indexes to the document collection. Different kinds of search are implemented as follows.

Author/title. In the vast majority of reports, the first page gives bibliographic information such as title and author. By limiting attention to this page, the user can approximate a search based on such information. For example, an initial page search for documents authored by *Knuth* will not retrieve documents that merely cite his work.

Publication date. Most reports also include publication date on the initial page, and the same technique serves for publication date searches. Alternatively, this facility could be simulated by permitting the user to search on the date in which a technical report was entered into its repository, although such a strategy is likely to produce uneven results because many reports are placed in repositories long after they were originally produced.

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 a search for *information retrieval*, many technical reports will contain these terms somewhere in the document, perhaps widely separated. The greater the physical distance between the two words, the less likely that the document is about the subject *information retrieval*. A limited kind of proximity searching is supported by requiring the query terms to appear on the same physical page of the document. 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 users to decide whether or not 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 helps to avoid artificial distinctions between capitalized and non-capitalized forms of the same word (such as *Information* and *information*) or between different forms of a word (*retrieval, retrieve, retrieving*). These truncated terms of the root word should be sought automatically, without forcing the user to list all alternatives.

Indexes

To support the kinds of search identified above, there are four separate indexes to the collection:

- a document-level index, with stemming and case-folding;
- a page-level index, with stemming and case-folding;
- an index of first pages only, without stemming and case-folding;
- a document-level index, without stemming and case-folding.

For an ordinary topic search, a user chooses the first index, or, for a more focused query, the second one. To search by author, title, or date, the third index is used; while the fourth is appropriate to seek cited authors or papers.

Query interface

The World Wide Web is the access medium for the digital library. Figure 1 shows the query page and a typical response. Because of the Web's 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, but can be increased 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.

It is clearly necessary to share the query engine between users. MG was not designed to deal with simultaneous multiuser access, and while it is possible to launch a process for each query, the startup overhead makes this very inefficient. Consequently we have arranged for access to one persistent copy of the query engine to be shared between incoming queries. A copy runs continually for each of the four collections, and additional software provides exclusive access for the duration of each query.

BUILDING THE COLLECTION

The lowest common denominator for representing information in conventional libraries is paper, and many digital library efforts involve scanned paper documents (e.g. [2], [3]). In the world of electronic information, PostScript, rather than plain, unformatted text, is the closest analog to paper as a document storage medium, and its page-based nature turns out to be helpful in structuring the collection. In order to build the index, it is necessary to be able to extract plain, unformatted, text automatically from the documents. In fact, the system design accommodates not just PostScript but any format from which paginated ASCII text can be extracted—for example DVI or RTF files. Even document images can be accommodated by OCR-ing them for indexing purposes: the inevitable recognition errors will reduce the quality of the index, but this can be ameliorated by using ranked queries containing redundant terms. However, from an initial investigation it appeared that PostScript files are almost universal in computer science technical report archives, and the system currently deals only with this format.

Archives of technical reports can be located through several lists maintained on the Internet, and recursively descending the directory hierarchy looking for (possibly compressed) PostScript files. Each file is downloaded, along with its size and date, and the appropriate information is extracted.

Information stored centrally

With an Internet-based digital library, a crucial question is how much information to store centrally. It was decided that the library would comprise an index and search engine, and the documents themselves would remain in their original repositories. Periodically, all repositories are scanned to refresh the index, adding any new documents and removing references to those that have been deleted. There is no intention to provide an archiving service that will retain documents that are removed from their original sites.

In addition to the index, a facsimile image of each document's first page or two is retained locally so that users can read the title and abstract, and sample the look and feel of the original. The plain, unformatted, text must be extracted from the documents to build the index, and it proves expedient to retain a full copy of this text locally as well. This is useful in its own right for browsing: users can examine the text without going to the trouble of downloading PostScript. Moreover, the text of the collection provides an excellent foundation for bibliometric research. 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 a facsimile image of the first page.

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 2 illustrates some of these problems.

The solution is to prepend a PostScript prologue that redefines the operators responsible for placing text on the page, and execute the resulting file using 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 variety of PostScript files it can handle. In fact, *ps2ascii* failed on 75% of 144 reports chosen at random each from a different site, while another convertor, that used in the Harvest system [4], failed on 40%.

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 nevertheless seem to work well in practice. This process not only extracts the body of the report, but also any text in figures and tables.

First-page facsimile

The facsimile image of the first page 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 the facsimile is stored 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 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 with some content is imaged, the information in a page is estimated 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 10 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 in practice.

Collection maintenance

All that is necessary to maintain the library's collection is to ensure that the documents indexed continue to exist. This can be checked by periodically examining the technical report repositories for changes and updating the collection accordingly. Considerable benefit is obtained by not relying on any cataloguing information stored with the repository itself: for example, frequent maintenance problems in technical report servers are caused by changes in the bibliography file format, and inconsistencies in the information, in the repositories that they index [5].

One source of growth for the collection is when new documents in known repositories are located during a routine examination of currently indexed sites. New sites can be detected by various means: monitoring standard Internet lists for new additions; manually scanning the newsgroups that announce them; and encouraging users to email suggestions to a central coordinator.

GATHERING THE INFORMATION

One of the motivations for a New Zealand digital library is efficient use of the expensive transPacific Internet link. Computer scientists can search for and preview technical reports locally before downloading the full document file, thus encouraging 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 across the Pacific would negate any cost benefits the project might offer.

Consequently a distributed scheme is used to create the index. In the first stage, a computer in North America downloads each technical report, extracts the facsimile images and raw text, sends them to New Zealand, and deletes the report. This process is illustrated in Figure 4.

This operation significantly reduces 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 Pacific link. In actuality, only 700Mb of uncompressed text, which reduces to 230 Mb when compressed, need be transmitted, 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.

SIZE OF THE LIBRARY

The lower block of Table 1 summarizes the size of the document collection and the storage requirements for the MG indexes that comprise the library. As discussed above, four separate indexes into the document collection are maintained. The total space they occupy is 140 Mb, a mere 2% of the total size of the PostScript files that are indexed. The text of the technical reports occupies 230 Mb when compressed by MG. Finally, the first-page facsimiles occupy an additional 280 Mb, giving a total disk storage requirement of 650 Mb, 10% of the total PostScript size.

What are the limits to the size of a library of this nature, using current technology? All sizes involved increase linearly with the volume of text, and a factor of ten growth would only involve a few Gb of disk storage. Retrieval time is virtually independent of database size: it takes two disk seeks per query term and two per document retrieved. The database inversion process is more likely to be a limiting factor. Moffat and Bell [6] report an improved algorithm that is faster and requires less main memory than MG: extrapolating from their experiments with a 2 Gb text collection they estimate that 5 Gb of raw text can be inverted in twelve hours with only 40 Mb of main memory. This corresponds to growth of the digital library by a factor of seven. It provides an indication of what is certainly possible, although there is no reason to believe that it represents a limit. The resulting collection would index 80,000 documents comprising two million pages or 45 Gb of PostScript, and the size of the centralized information, including plain text, indexes, and first-page facsimiles, would amount to just 4.5 Gb.

Nevertheless, an all-inclusive information collection policy is basically unscalable and will become infeasible as the number and size of technical report repositories grows. There are several possibilities for culling the collection when that becomes necessary. One is to monitor participating users' access, and note the repositories that receive the least use. This regulates the growth of the collection by the size, diversity, and level of activity of the user population rather than by the growth of the bibliographic universe. Another is to use the characteristics of the documents themselves to determine what to cull from the collection. For example, citation studies could identify documents that do not appear to contribute to the literature.

FUTURE DIRECTIONS

The operation of the digital library hinges on the idea of extracting information automatically from PostScript. Presently, words and page boundaries are extracted, and the process is entirely algorithmic. The collection would be enhanced if reliable ways could be devised for determining bibliographic details of the reports so that a proper catalogue could be built, rather than relying on their appearance on the first page (which may be a cover page). Moreover, the text collection provides an ideal basis for bibliographic studies in computer science. If it were possible to locate references sections within documents, a separate database could be constructed that would be extremely useful for citation analysis. Hence we need to consider how to extend the extraction process by incorporating heuristics to identify some document structure.

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 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. The same goes for Greek letters. The text often looks messy because mathematical formulae and diagrams come out as strange characters; its appearance would be improved by a mechanism to detect these and replace them by some standard place-holder. A significant improvement would be to detect paragraph boundaries so that the text could be re-paragraphed automatically 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 on the printer). We have found no reliable way of telling that they are backwards; again, a heuristic solution is needed.

Users find it hard 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 [7] provide a way to visualize 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. 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 multipart 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.

CONCLUSION

Public digital libraries can be constructed on the Internet in any area for which repositories of suitable text can be located. Extracting all index information from the documents themselves is feasible if full-text indexing is used, and virtually eliminates the manual cataloguing effort involved in creating and maintaining the library. The material in such a library can be distributed globally. The central database in our prototype library, which comprises a full-text index, facsimile images of the first page or two of each document, and the plain text of all documents, represents only 10% of the collection size.

Digital libraries represent one way of dealing with the new reality of Internet publishing. Making a minimum of assumptions, our scheme imposes structure on a fundamentally anarchic, uncatalogued, system, giving information consumers a tool to find what they need.

ACKNOWLEDGEMENTS

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, and the Computer Science Department at Calgary generously served as our North American site.

REFERENCES

1. Witten, I.H., Moffat, A., and Bell, T.C. (1994) *Managing Gigabytes: Compressing and indexing documents and images*, Van Nostrand Reinhold, New York.
2. Crocca, W.T. and Anderson, W.L. (1995) "Delivering Technology for digital libraries: experiences as vendors," *Proc. Digital Libraries '95*, pp. 1–8, Austin, Texas, June.
3. Van House, N.A. (1995) "User needs assessment and evaluation for the UC Berkeley electronic environmental library project: a preliminary report," *Proc. Digital Libraries '95*, pp. 71–76, Austin, Texas, June.
4. Bowman, C.M., Danzig, P.B., Manber, U., and Schwartz, M.F. (1994) "Scalable Internet resource discovery: Research problems and approaches," *Communications of the ACM 37*(8), pp. 98–107.
5. Van Heyningen, M. (1994) "The Unified Computer Science Technical Report Index: Lessons in indexing diverse resources," *Proc. Second International WWW Conference*, Chicago.
6. Moffat, A. and Bell, T.A.H. (1995) "*In situ* generation of compressed inverted files," *J American Society for Information Science 46*(7): 537–550; August.
7. Hearst, M.A. (1995) "TileBars: visualization of term distribution information in full text information access," *Proc. CHI'95*, pp. 56–66; May.

INTERNET TECHNICAL REPORT SEARCHING SYSTEMS (SIDEBAR)

Several technical report searching and indexing systems are summarized in the Table. 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 180 sites and 11,000 documents.

Name	Domain	Reference	Sites	Documents	Indexed on
e-print archive	Physics	[1]		–	user-supplied title, author, abstract
NTRS	NASA	[2]		2 000 000	abstracts
UCSTRI	CS	[3]	180	11 000	file of abstracts at ftp site
DIENST	CS	[4]	15	–	bibliographic information provided
WATERS	CS	[5]	15	2 000	citation and possibly abstract
HARVEST	CS	[6]	300	37 000	limited info from several file types
NZDL	CS	this paper	170	11 000	full text from PostScript

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

REFERENCES

1. Ginsparg, P. (1994) First steps towards electronic research communication, *Computers in Physics 8*(4), pp. 390–401.
2. Nelson, M.L., Gottlich, G.L., and Bianco, D.J. (1994) "World Wide Web implementation of the Langley Technical Report Server," *NASA Technical Memorandum 109162*, Langley Research Center, Hampton, Virginia.
3. Van Heyningen, M. (1994) "The Unified Computer Science Technical Report Index: Lessons in indexing diverse resources," *Proc. Second International WWW Conference*, Chicago.
4. Davis, J. & Lagoze, C. (1994) "A protocol and server for a distributed digital technical report library," Technical Report 94-1418, Computer Science Department, Cornell University.
5. Maly, K., Fox, E.A., French, J.C., and Selman, A.L. (1994) "Wide area technical report server," Technical Report, Dept. of Computer Science, Old Dominion University.
6. Bowman, C.M., Danzig, P.B., Manber, U., and Schwartz, M.F. (1994) "Scalable Internet resource discovery: Research problems and approaches," *Communications of the ACM 37*(8), pp. 98–107.

WEB SEARCH ENGINES (SIDEBAR)

The World Wide Web is a structure of interlinked pages containing text, pictures, and pointers to other pages. The pages are formatted in HTML, a simple hypertext markup language, although links can point to files of any format. Search engines, often dubbed "spiders," traverse the Web in search of text, following the links that pages contain. The HTML representation makes it trivial to extract the text and links from the pages. GOPHER is a precursor to the Web which is based on a hierarchical structure of plain menus.

Several Web search engines are summarized in the Table. LYCOS permits a full-text search on the contents of Web pages and some of the files that it points to (although most non-HTML file types, including PostScript files, are excluded.) The text searched is restricted to the first few lines of each page, and certain high-information-content keywords that appear later in the page. LYCOS has encountered 5.6 million pointers to Web pages, but 75% of these are unexplored. At the current rate of 15,000 documents per day it will take nearly nine months to explore the rest, by which time the Web will have grown considerably. INFOSEEK and the WEBCRAWLER are similar systems.

JUMPSTATION provides three separate indexes to HTML pages: one for titles, a second for headings, and the third for the most frequent words in the text. OPENTEXT, a recent addition to the field, allows full text search of nearly one million pages, including ranked, boolean and proximity searches. Searches can be confined to summaries, titles, first headings, links, and body text. OPENTEXT indexes up to 50,000 pages per day and seems poised to usurp LYCOS as the server of choice.

Some more restricted services are also available. ALIWEB indexes services based on specially formatted submissions from service providers. VERONICA claims virtually complete coverage of the documents pointed to by the GOPHER menu structure—although no attempt is made to index the document text.

Name	Domain	Location	Documents	Indexed on
LYCOS	WWW	`http://lycos.cs.cmu.edu/`	1 200 000	partial full text
OPENTEXT	WWW	`http://opentext.uunet.ca:8080/omw.html`	1 000 000	full text
INFOSEEK	WWW	`http://www.infoseek.com/`	400 000	partial full text
WEBCRAWLER	WWW	`http://webcrawler.com/`	150 000	full text
ALIWEB	WWW	`http://web.nexor.co.uk/public/aliweb/aliweb.html`	8 000	abstracts
JUMPSTATION	WWW	`http://js.stir.ac.uk/jsbin/jsii`	–	keywords, headings
VERONICA	GOPHER	`gopher://veronica.scs.unr.edu/11/veronica`	15 000 000	menu items

Web search engines

Source documents	total	per report
Sites	170	
Documents	11 000	
	130 million words	12 000 words
	310 000 pages	30 pages
Original PostScript	6.6 Gb	600 kb
compressed with gzip	*2.3 Gb*	*200 kb*

Extraction process

Text	700 Mb	65 kb
compressed with gzip	*230 Mb*	*20 kb*
First page facsimiles	25 000 pages	2.2 pages
	280 Mb	25 kb

Indexed collection

Text			230 Mb
Index	*Indexed by*	*Stemming/ case folding*	
	report	yes	30 Mb
	report	no	40 Mb
	page	yes	65 Mb
	first page only	no	5 Mb
First-page images			280 Mb
Total			650 Mb

Table 1 Current size of the digital library

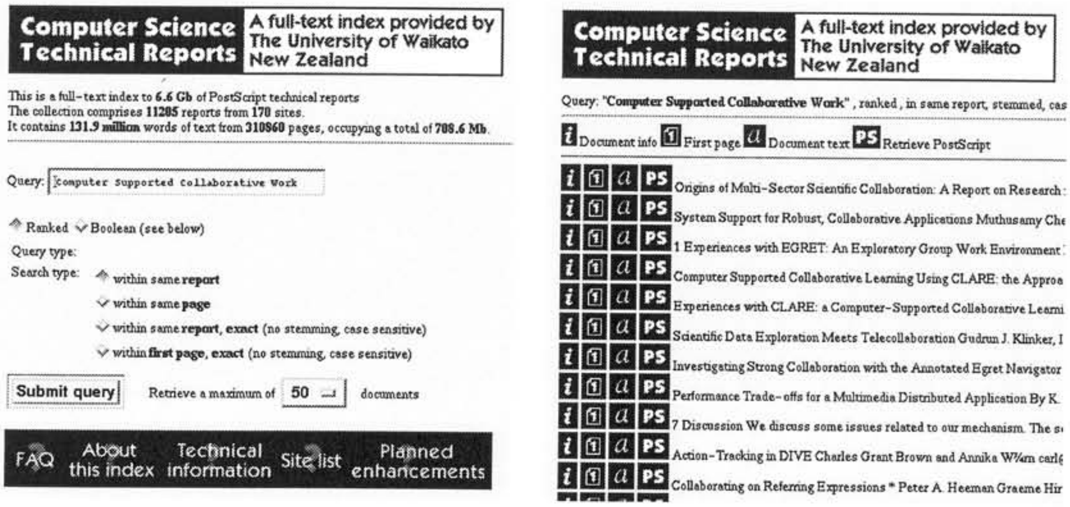

Figure 1 Interface to the digital library: the query page and a typical response

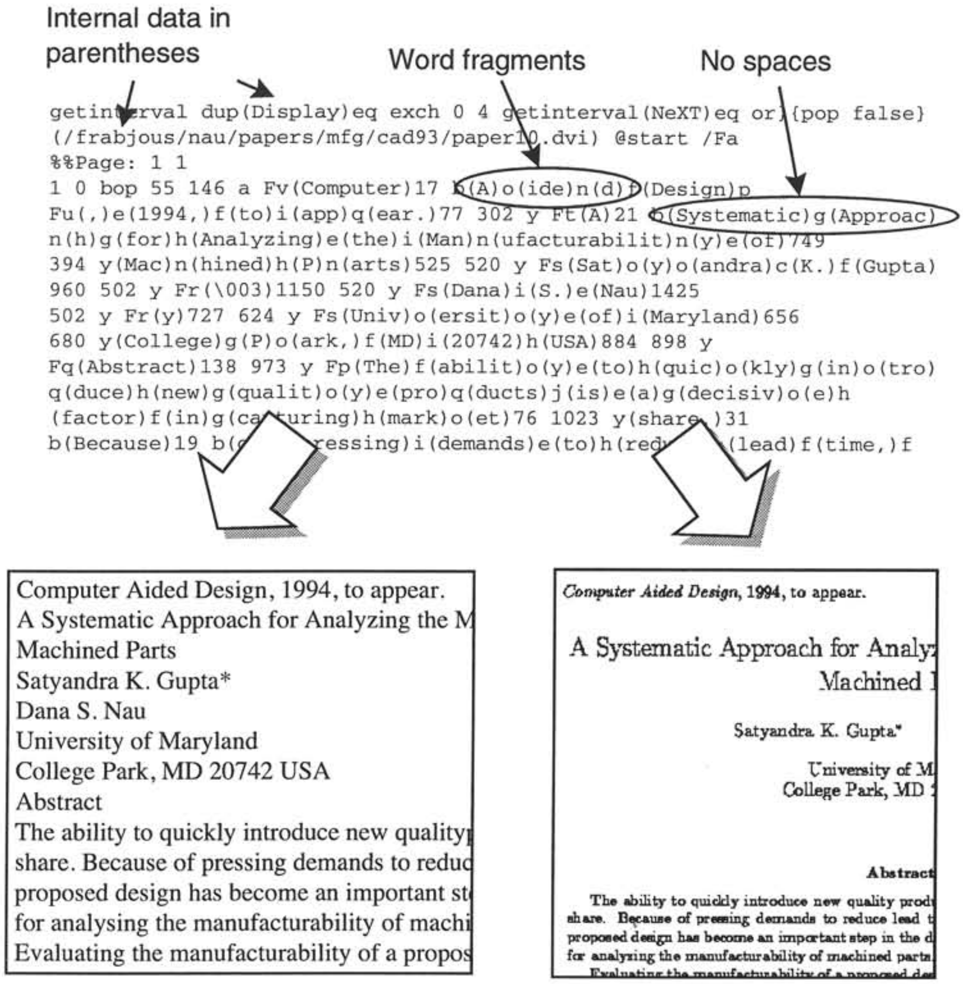

Figure 2 Conversion from PostScript: a PostScript file, the text extracted from it, and a facsimile image

`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` *with the original operator on the stack*
`/ashow { /ashow simple } def`	
`/awidthshow { /awidthshow simple } def`	
`/widthshow { /widthshow simple } def`	
`/kshow { /kshow simple } def`	
`/showpage { extract_file (\nNZDL_NEW_PAGE\n) writestring } def`	*record the page boundaries*

Figure 3 The prologue used to extract text from PostScript documents

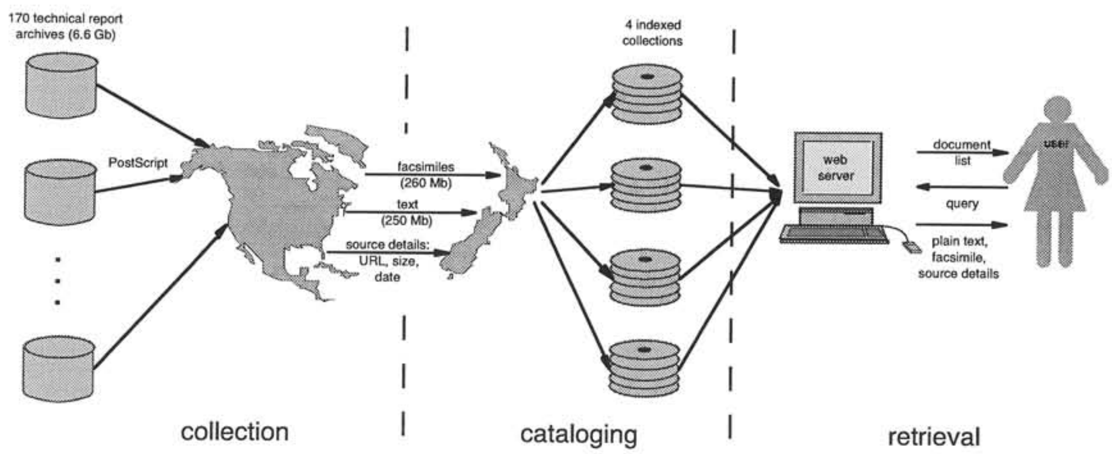

Figure 4 Architecture of the digital library