DESIDOC Bulletin of Information Technology, Vol. 17, No. 6, November 1997, pp. 33-44

A Compression-Based Digital Library

Alistair Moffat* and Ian H Witten**

Abstract

The explosive growth in digital information places an ever- increasing burden on the library mechanisms used to store and access it. A wide range of technologies must be developed to manage this flood of data. For example, we need ways of using storage effectively and economically, mechanisms for indexing, and methods for fast content-based querying. In this article we consider these three issues from an implementation point of view, showing how appropriate data compression techniques can be harnessed to provide improved solutions in all three areas. To set a context for the development, we discuss the New Zealand Digital Library initiative, a Web-based tool that provides many different services, the largest of which indexes in excess of 40,000 technical documents drawn from sources around the world.

Keywords - digital library, text compression, indexing, query mechanisms, coding.

1. INTRODUCTION

The explosive growth in digital information places an ever-increasing burden on the mechanisms used to store and access it. The rapid evolution of the internet into its current form is but one example of this phenomenon: within every large organisation a similar growth of online data is taking place. Traditional paper- based schemes for organising the storage of textual documents so that they can be located and accessed later are breaking down, and computer-based solutions are being designed to replace them.

A wide range of technologies must be employed to manage this flood of data, including (but not restricted to) mechanisms for providing economical use of storage, mechanisms for indexing, and mechanisms for providing fast content-based querying. A digital library system must be capable of storing the volumes of data in question; it must have a viable indexing technique for recording, for each document so stored, what concepts it describes; and it must be able to use the index to find information in the document collection within reasonable amounts of time and space. The data stored by an organisation is its most valuable resource, and any inability to locate relevant information within a large collection of documents will severely hamper its effectiveness.

In this article we consider the three issues of storage, indexing, and retrieval from an implementation point of view, showing how data compression techniques can be employed to increase both space and time efficiency. We discuss how to compress documents in a

*Department of Computer Science, University of Melbourne, Parkville, Australia 3052, Fax: +613 93481184. Internet: alistair@cs.mu.oz.au
**Department of Computer Science, University of Waikato, New Zealand. Fax: +6478384155. Internet: ihw@waikato.ac.nz.

collection; show how the use of compression can make indexes for multi-gigabyte databases manageable and also aid in their construction; and finally survey techniques for supporting content-based queries on very large retrieval systems.

To set a context for the development we also discuss the New Zealand Digital Library, a Web-based tool that provides many distinct services, the largest of which indexes in excess of 40,000 technical documents drawn from sources around the world. It uses as its underlying search engine the public domain MG software system, a collection of programs that through the use of compression provide economical storage and indexing for large collections of documents, as well as fast index construction and query processing.

2. COMPRESSING THE STORED DOCUMENTS

Compression technology has come a long way since the early work of Shannon, Fano, and Huffman[1]. A compression mechanism be it for text, images, video, or sound is now thought of as two cooperating processes. Coding, the simpler of the two, takes a symbol from a defined alphabet, together with a specified set of symbol probabilities, and represents it by a codeword over some channel alphabet usually binary. Shannon showed that a symbol whose probability of appearance is estimated as P_s should be assigned a codeword exactly $-\log_2 p_s$ bits long. Coding is the process of calculating such codewords.

The other, more challenging, task is that of modelling. The modelling component assigns a probability to each possible next symbol, and provided an effective coder is being used – the more accurate the prediction, the better compression will be. Current high performance compression schemes make use of contexts of several consecutive preceding characters, so that, for example, following the characters *qui* the character z might be given a relatively high probability, certainly much higher than it would be based solely upon its overall frequency of occurrence.

Although in compression one normally seeks the best possible storage efficiency, for document databases and digital libraries other constraints come into play. One important requirement is that documents be individually accessible. This means that adaptive models that adjust their statistical parameters as the input stream proceeds cannot be employed, since in an adaptive model the probabilities, and possibly the set of contexts used, are determined as a function of all of the previous data until that point, and the compressed representation must be decompressed in a sequential manner from the beginning. A second requirement is that whereas encoding is not so critical, decoding must be fast, because retrieval systems must normally be capable of supporting many simultaneous users, all making independent requests. The compression mechanisms employed in a digital library must also be capable of handling a wide range of data types.

The MG software system[5] provides tools for compressing plain text documents, bi-level images, scanned textual images, and grey-scale images. In the remainder of this section we concentrate on the mechanism used for text compression, with only passing reference to the other compression modes. Detailed descriptions of the methods used for compressing the other types of data may be found elsewhere.[11]

2.1 Static versus Adaptive

Adaptive modelling has drawbacks for document databases, as we noted above. Instead, MG makes use of a semi-static arrangement in which the collection of documents is scanned in a preliminary pass to determine the symbol probabilities, and then coded in a second pass. This makes the set of codes fixed and the same for each document, allowing short documents to be represented economically without the 'learning' overhead that results from the need for the symbol probability estimates to converge from initial approximations.

Hence, the compression effectiveness is independent of the length of the document. This is important in a document database,

where each unit of access might be as short as a dozen words a brief catalogue entry describing a title and an author might be all that is available online, for example. An alternative would be to adapt the model within each document, possibly starting each document with some good approximation of the model parameters gleaned by an initial analysis of the collection as a whole. Although the local adaptivity of such a mechanism might give better compression within long documents, fully adaptive compression is costly in terms of resources, and not recommended. On the other hand, partially adaptive compression is useful when collections are to be dynamic, and this point is considered further below.

Given a bit address for the start of a document, a semi-static model allows the documents to be decoded independently of each other. The absence of adaptation in a semi-static model also means that decoding is faster than in an adaptive model, a further major advantage.

2.2 Word-Based Modelling

The model employed in the MG system is a word-based one. The source documents assumed here to be in raw text form, but also possibly word-processing documents in proprietary formats are parsed into a sequence of words and non-words that strictly alternate. Each of the words is then coded relative to a lexicon of words and their observed frequencies, and each nonword is similarly coded relative to a lexicon of non-words. The number of words that appear in a collection of text can be large, and as a rule of thumb novel words appear at a rate of about one per one thousand word appearances, even after several gigabytes of text have already been considered. (Most of the new symbols are proper names, acronyms, and spelling mistakes; nevertheless, they must be handled correctly by the compressor) Despite the large number of distinct words, there are many words that appear frequently, and if these words are assigned short codes then substantial compression results. For typical text the average space required for each word code is about 10-12 bits, and a further 13 bits on average are required for each non-word code. Thus 11-15 bits suffice to represent perhaps 56 characters on average, a compressed size of about 25-40% of the original. While not the best possible compression, these rates are comparable to those attained by string-based parsing approaches, and have the useful side effect of making a lexicon of all words available.

2.3 Fast Decoding

The coding mechanism used in MG is minimum redundancy coding[6], also sometimes known as Huffman coding. Discrete bit codewords are assigned, with the length of each codeword inversely related to the observed frequency of the word in the collection. Frequent words—*the* is usually the most frequent in a collection of English text get codes that are four or five bits long, while *hapax legomena,* words that appear only once, get codes as long as 30 bits or more, depending upon the total number of words in the collection.

The lengths of the codewords are calculated using a carefully organised implementation of Huffman's famous algorithm.[11] The codewords themselves are then assigned in a controlled pattern that results in the codewords being lexicographically ordered when the symbols are frequency ordered. This so called canonical arrangement allows extremely fast decoding, little more than a machine level integer comparison per bit decoded. Moreover, because the tokens being decoded are words strings of four or five characters—each decoded symbol produces several bytes of output. The final result is surprisingly fast execution.

Compression is normally thought of as being a technology that trades computation time for storage space, and this is still true on the encoding side. But when accessing a small a much larger collection there is no need to trade off: speed increases with compression. The smaller size of the compressed collection makes disk seek times smaller, and the smaller size of the compressed documents reduces the disk transfer time for each record. These effects are sufficiently great that the savings include ample time for accessed records to be decoded. That is, compression with an appropriate choice of mechanism, of course can actually reduce the

elapsed needed to access documents out of a large collection.[12]

2.4 Collection Extension

One operation that is required in a document database that is not germane to most other compression applications is the need to allow for the growth of the collection as new documents are appended. In the semi-static word-based model described above new documents can be compressed provided that they contain only words (and non-words) already present in the lexicon. Novel symbols, however, pose a problem, since no codewords are assigned to them. (It is clearly absurd to assign in advance a codeword to every possible sequence of say eight characters—the result would, of necessity, look little different from the ASCII already used to represent the text.) One way of handling novel symbols is to assign an *escape* code at the time the original code is devised that allows access to a secondary character-based model. When novel symbols appear they can be coded using the character level code, and so the document collection can expand beyond its original size. More complex arrangements, in which the new words are then included into the lexicon with non-optimal codes, are also possible. Using these techniques, collections can be expanded by factors of 100 or more with negligible degradation in compression[8].

2.5 Other Data Types

Other types of data are also stored in digital libraries: scanned of archival material, greyscale or colour images, video clips, sound bites, high-fidelity music, and so on.

It is important that these media also be stored economically, since they tend to be space intensive than even large quantities of text. A wide range of suitable compression mechanisms have been developed. They lie beyond the scope of the present article; interested readers will find relevant material in the book by Jayant (1997).[4]

3. INDEX COMPRESSION

An index to a large document collection, necessary for any comprehensive document retrieval system, also has the potential to consume large amounts of space. This section provides an overview of the indexing mechanism used in the MG system.

3.1 Inverted File Indexing

The most efficient indexing mechanism for large document collections is the *inverted file*[2,13] for a discussion of alternative mechanisms). An inverted file index consists of a lexicon that contains information about each of the terms or concepts that are being indexed, and a set of inverted lists, one for each of those terms. Each inverted list records identification numbers for all documents in the collection that are 'about' that term or concept.

The concepts can be assigned to documents manually, or, as is now usual case for electronic text, automatically. The simplest automatic assignment for text is to index every word that appears, possibly after the application of transformations such as case-folding, and stemming to root form by the removal of suffixes. In some situations one might also filter out common terms using a *stop list*, and not index them. This does reduce the size of the index but has the unfortunate side-effect of pre-effect of pre-empting queries on stopped terms that might, under some circumstances, be perfectly valid. Because of this desire for universality, we suggest that all words and numbers should be indexed.

3.2 Representing the Index

Uncompressed, this index might be large. A four-byte document number occupies about the same space as the word it represents, and even allowing for the fact that on average each word in a document appears (say) twice, the index will still occupy an overhead of around 50% of the space of the original source text. To reduce this space requirement and achieve compression, each inverted list can be sorted by document number and represented as a sequence of differences between successive document numbers. This also speeds query processing, as is described below. Representing the differences with a Golomb code gives an average cost, on typical data, of about 5-6 bits per document number[11]. To this must be added

the cost of storing the *within document frequency*, denoted for $f_{d,t}$ the frequency of term t in document d, the value of which is required for evaluating ranked queries (described in Section 4). That component can be stored in 1-2 bits per pointer on average, for a total indexing cost of around one byte per pointer. Supposing as before that on average each word that appears in a document does so twice, and that each word in a document accounts for about 5 bytes (including whitespace), the total cost of the compressed index is about 10% of the size of the source text. These estimates are supported by experimental results over a wide range of large document collections. [11]

3.3 Word-Level Indexing

For some applications it is desirable to know not only whether or not a word appears in a document, but also exactly where. For example, we might wish to support queries that require that two particular words be adjacent, or within some specified distance of each other. This can be achieved with a word-level index. Adding a list of word positions to each document pointer is a straight forward application of the techniques used at the document level. It does, however, add significantly to the size of the index. The savings due to repetition of words within a document are lost, and several bits of locational information per word appearance must be added. In combination these two effects take the size of a compressed word-position index back up to the 25% mark. Indeed, unless the majority of queries are going to involve adjacency or proximity, the best compromise might be to use a document-level index and handle the specialised query modes with a post-retrieval scan.

3.4 Index Construction

Compression has a useful role to play during the process of index *construction*. The volume of data involved means that conventional cross reference generation techniques cannot be used, and instead a range of methods have been developed that use compressed representations throughout to save space. These methods allow indexes for multi-gigabyte collections to be constructed in a few hours on typical workstations or personal computers.

4. QUERYING THE COLLECTION

The purpose of the index is, of course, to support content-based queries. Boolean queries—terms connected with the Boolean operators AND, OR, and NOT—are easily handled using an inverted file. To process such a query the inverted lists for the terms are fetched, decoded, and then intersected for AND, merged for OR, and complemented for NOT. In traditional retrieval systems Boolean queries were the standard paradigm for information searching. They do, however, have the drawback of requiring that users master the intricacies of a query language.

4.1 Ranked Queries

Instead, most modern systems make use of *ranked* queries, in which a list of query terms is specified using a natural language approach, and a similarity heuristic used to estimate the likelihood for each document in the collection that it is relevant to the query. One heuristic is the *cosine* measure[10], so named because the value calculated is the cosine of the angle in n-dimensional term space between a document vector and a query vector, each of which contains a weight w for each of the n terms present in the collection. One possible specification of the cosine measure is the value $S_{d,q}$ given by

$$w_{d,t} = \log_2 (1+f_{d,t})$$

$$w_{q,t} = \log_2 [1 + (\frac{N}{f_t})]$$

$$W_d = [\sum_{t=1}^{n} w_{d,t}^2]^{1/2}$$

$$W_q = [\sum_{t=1}^{n} w_{q,t}^2]^{1/2}$$

$$S_{d,q} = \sum_{t=1}^{n} \left[\frac{(w_{d,t})}{W_d} \cdot \frac{(w_{q,t})}{W_q} \right]$$

in which there are N documents and n distinct terms, weights $w_{d,t}$ and $w_{q,t}$ are taken to

be zero for terms t not present in document *d* and query *q* respectively, and where term *t* appears in f_t of the documents and $f_{d,t}$ times in document *d*.

Cosine similarity values can be computed by processing inverted lists, and require only slightly more time than a disjunctive Boolean query on the same set of terms.[7] Although ranked queries tend to take longer to execute than Boolean ones, this is only because users feel more comfortable typing lengthier and thus more specific queries.

4.2 Assessing a Querying Mechanism

The effectiveness of querying mechanisms is hard to assess. Not only do users have differing requirements and expectations, but the data itself can influence the efficacy of the information search. For example, most users seeking an entry point to the literature using a World Wide Web search engine will be pleased if they get a single hit within the top ten documents returned, and may not look any further down the ranking than that first screen.

On the other hand, a lawyer searching for a legal precedent may explore a long way down the ranking, and be pleased not so much when a hit occurs but when a large number of suggestions have been considered and no hits located. In terms of the traditional measures of *precision* (the fraction of returned documents that are relevant) and *recall* (the fraction of relevant documents that are returned), the first of these two searchers is interested in moderately high precision at low recall levels, while the second cares only about obtaining high recall.

Calculating recall for non-trivial experimental databases is expensive, since it is necessary to check every document in the collection manually against each of the test queries. Assessing precision at a given level of returned documents is easier, but still a considerable undertaking. One of the most important outcomes of TREC, a large-scale international project involving many prominent research groups[3], is a large set of queries and relevance judgements against a particular multi-gigabyte document collection. These form an invaluable resource, and *considerable improvement* in retrieval technology, as measured by recall and precision averages, has been reported at TREC conferences over a five-year period.

4.3 Other Searching Modes

There are other information searching methods that might be used in the digital library. As in the conventional library, browsing is an important means of information retrieval. In a library of physical documents it is not at all uncommon to use the catalogue (which is, of course, the index for the collection) to locate an entry point into the collection, and continue the search by browsing documents on neighbouring shelves. In a physical library this clustering of related work is a one-dimensional projection of the hierarchical classification scheme used to organise the library. For example, in both the Dewey Decimal and Library of Congress classification systems work that is spatially close to a document is likely to be nearby in a content sense as well. In digital libraries the clustering mechanism is no longer constrained to be one-dimensional, and multiple adjacencies can be supported.

The success of the World Wide Web is a good demonstration of the power of this approach. Documents contain as many links to related work as their authors wish to provide, and the human browser is usually supplied with a brief context that allows them to decide, for each such link, whether it is likely to provide additional information relevant to their current information need. To find information, users call on one or more indexing and search engines to obtain a small set of starting points, and then browse from there. World Wide Web searching is a good example of an application in which an overall recall precision average as a measure of retrieval effectiveness is less important than getting at least one hit in the top ten. The user cares little that all of the pages they eventually access are somewhere in the listing returned from the search engine, if they can all be found linked from a single page near the top of the ranking anyway.

5. THE NEW ZEALAND DIGITAL LIBRARY

MG indexing and search engine described in the above sections is used as the kernel of the New Zealand Digital Library project, a publicly accessible system on the World Wide Web (at *http://www.nzdl.org*) that provides full-text indexes to several substantial collections of information. This section describes some of the facilities developed in that project.

5.1 Computer Science Technical Reports

The flagship collection is a library of computer science technical reports, which currently provides a full-text index to 40,000 reports one million pages, containing 400 million words collected from over 300 sites around the world. It involves nearly 2.5 Gigabytes of text, extracted automatically from 35 Gigabytes of source information by a system developed specifically for this task.[9] A sub-collection has been set up containing technical reports gathered from sites in Germany, and the New Zealand Digital Library software has been transferred to the University of Bonn which offers this collection on an experimental basis as part of the German digital libraries program.

The current interface to the Library allows documents to be located by either Boolean or ranked search. The latter performs an OR query on the specified terms and ranks the documents retrieved by relevance according to the cosine rule described in Section 4. In either case, queries are answered quickly, and the plain text of each matching document is made available for browsing almost instantly.

The Computer Science Technical Report collection is the most mature, and organisationally the most complex, collection offered by the New Zealand Digital Library to date. Intended as a serious research resource, it

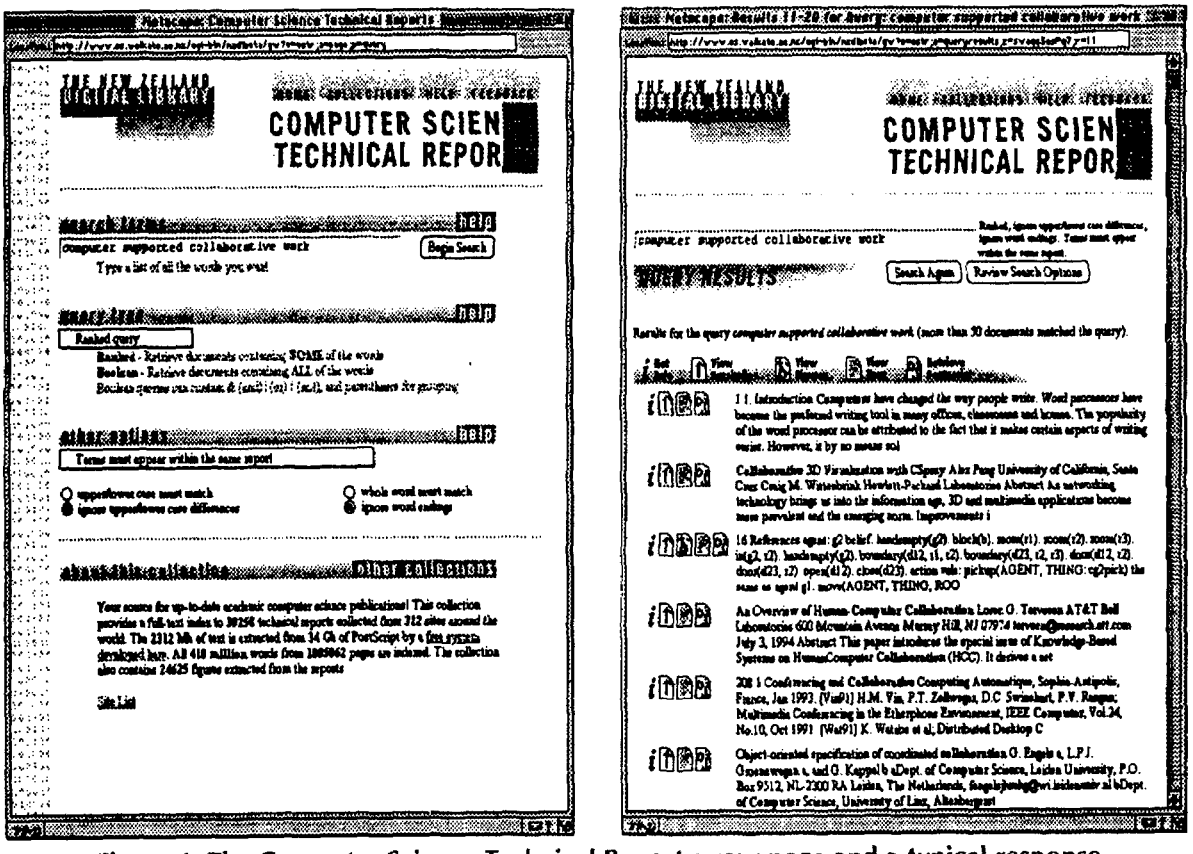

Figure 1: The Computer Science Technical Report query page and a typical response

is used regularly by computer scientists world wide. Figure 1 shows the query page on the left and a typical response on the right. The searching options, which are entered on the query page, include:

- Choice of ranked or Boolean queries;
- The ability to stem terms, and/or make them case-insensitive, both on a whole-query basis and a term-by-term basis;
- Phrase searching;
- Choice of searching at the document or the page level; and
- Searching first pages only.

In order to bound the resources consumed by each query, a maximum of 50 documents are returned. The 'query response' page on the right of Figure 1 contains information about ten of them, including the first few words of the document and four or five icons that serve as buttons. From these the user can quickly retrieve the:

- URL of the original document, with its size, creation date, and download date;
- Original document itself, downloaded from that URL;
- Facsimile image of the first page or two of the original document;
- Text extracted from the original document, crudely formatted; and
- Figures extracted from it (if they are in encapsulated PostScript).

Other collections are organised in a similar manner, although since they are not usually extracted from PostScript, they are somewhat simpler: the text is the original document and there is no need to provide access to it separately, nor to facsimile images of the first pages.

5.2 Other Collections

Computer science is unique in that a vast amount of high-quality information already exists in digital form and is freely accessible on the Internet in the form of technical reports. In order to demonstrate that digital libraries can benefit diverse groups of users, other collections of publicly-available information have been constructed.

- *The Computists Communique*. An online AI research news magazines, operating since 1991, this includes grant and funding opportunities, industry news, Internet and Web news, online resources, research discussion lists, news and software offers, software development resources, and career or entrepreneurial tips.
- *FAQ Archive*. Frequently Asked Questions are an Internet phenomenon that arose as a way of reducing the number of newcomers asking the same questions in USENET discussion groups. They have developed into a corpus of questions and answers on a huge variety of topics. This collection can be searched for terms appearing within the same Frequently Asked Questions list, under the same subject heading, or within the same paragraph.
- *The HCI Bibliography*. This is a free access online bibliographic database on Human Computer Interaction, created in a project whose goal is to put an electronic bibliography for most of HCI on the screens of all researchers, developers, educators and students in the field. With this collection, either reference entries alone, or reference entries and abstracts, can be searched.
- *Indigenous People*. This collection contains information on indigenous people found around the world. The material emanates from the Fourth World Documentation Project, which was started in 1992 to collect and distribute information of importance for indigenous people. The documents include essays, position papers, resolutions, organisational information, treaties, UN documents, speeches and declarations on social, political, strategic, economic and human rights issues. This project has become a primary information resource for universities, state and federal agencies, Indian and tribal councilmen.
- *Oxford Text Archive*. The public-domain texts in the Oxford Text Archive comprise 53 literary works ranging from Wuthering Heights to *The Red Badge of Courage*. This archive is provided by Oxford University Computing Services,

which aims to serve the interests of the academic community by providing archival and dissemination facilities for electronic texts at low cost.

- *Project Gutenberg Collection.* This consists of five hundred public-domain books: *The King James Bible;* classic literary works such as *Moby Dick, Alice in Wonderland,* and *Paradise Lost;* and reference books like *Webster's Dictionary,* the periodic table, and *The Hacker's Dictionary.* They are collected as part of the Gutenberg Project whose goal is to encourage the creation and distribution of electronic text. The texts are entered by volunteers who are encouraged to choose whatever books they like.
- *TidBITS* is a weekly electronic publication that covers news and views relating to the Macintosh computer, with a focus on Internet related topics. News items are independent of each other, and so it is possible to search individual items, individual paragraphs, or titles of news items.

The major difference between these collections is the source and format of information, the updating policy, the granularity with which searching can be done (document, page, paragraph, and so on), the different kinds of index that need to be provided (titles, references, abstracts), and the structure and format in which the output is to be displayed.

For example, Figure 2 shows, on the left, the query response page for the query *Perlman* made to the HCI Bibliography collection. Complete references are provided, and where more information is available—as in all but the third entry—a link is created to a fuller record.

At the right of Figure 2 are two sample records, which contain abstracts along with the reference themselves. The *detach* button at the top is used to detach the page so that it is retained while a second document is shown.

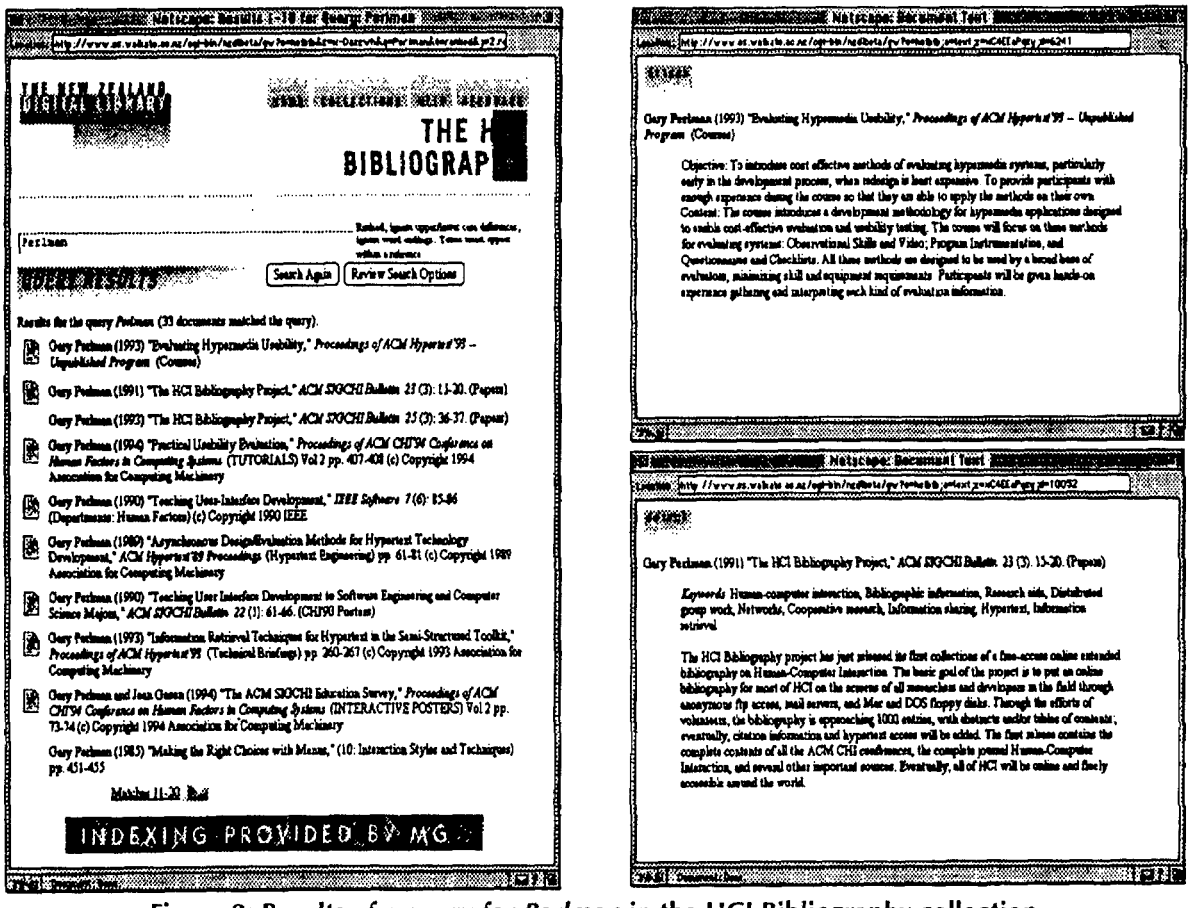

Figure 2: Results of a query for *Perlman* in the HCI Bibliography collection

This is a typical example of how the structure of the information displayed needs to be tailored specifically for that kind of collection. Another example, which occurs in some of the other collections, is the provision of a summary of the contents of the collection for example, a clickable list of titles or even a browsing structure to allow non-searching access to the collection.

5.3 Collection Development

Work is underway on a number of other collections that have been made available on a private basis until they are approved for public release. Of particular interest are those collections that create novel problems for full-text retrieval, and which introduce new issues that have not yet been addressed in the project. Here are some examples of collections that are being constructed.

- *The Karlsruhe Bibliographies.* This is a large collection of bibliographies in the field of computer science, totalling about 700,000 bibliography entries. It will be made similar to the current HCI Bibliography collection, except that users may be offered a choice of reference format so that they can cut and paste references into documents they are working on.
- *The Humanity Development Library.* This is a collection put together by the Global Help Project to provide ideas, experiences and solutions to workers in developing countries. The Humanity Library is the fruit of a massive humanitarian and development information transfer project to developing countries. Forty organisations are collating useful publications and resources to help reduce poverty, increase human potential, and provide education to all. The goal is that by mid 1998, any person in a developing country with access to the Internet, or to a PC with CD-ROM drive, will have direct low-cost access to a complete library of 3,000 books containing humanitarian aid information.
- *City Council Minutes.* In an effort to increase public awareness and participation in local government, a local city council is planning to make available on the Web records of council meetings, along with related discussion papers. Having these in searchable form will greatly increase their accessibility and usefulness. For example, one can search just motions, or paragraphs, or whole sets of minutes, to find information relevant to particular decisions, people, or places.
- *Library Catalogue.* A large library catalogue containing about 10 million records is being converted into a full-text-searchable collection. While conventional fielded search is appropriate when seeking an item that one knows is there and has a good deal of information about, full-text search of library catalogue entries is probably more convenient for non-professional users, in situations where one is not sure what one is looking for, and for casual browsing. We are adding author, title, and keyword fields, so that if necessary the user can disambiguate query terms by restricting them to one of those fields. Human factor experiments are planned to determine the relative efficiencies of field-based and full-text searching for various kinds of information retrieval tasks.

Each of these pose interesting problems. For example, the left part of Figure 3 shows the result of a query for marriage in the Humanity Development Library (the repetitions in the list are caused by repeated sections in the original document collection), and on the right is the fourth document, Closing the Gap Between Supply and Demand: Fertility Decline in Egypt. Because this collection is structured as a hierarchical system of document emanating from many different sources, it is rather hard to orient oneself in the collection. Consequently a navigational aid is presented at the top of the document returned (on the right) showing the section it resides in, along with the sibling subsections, preceded by the titles of hierarchically enclosing sections. This is generated automatically from the document structure, and by clicking on the folder and document icons the user can browse the collection very conveniently.

5.4 Usage, Distribution, and Collection Creation

The initial digital library prototype, the New Zealand Digital Library for Computer

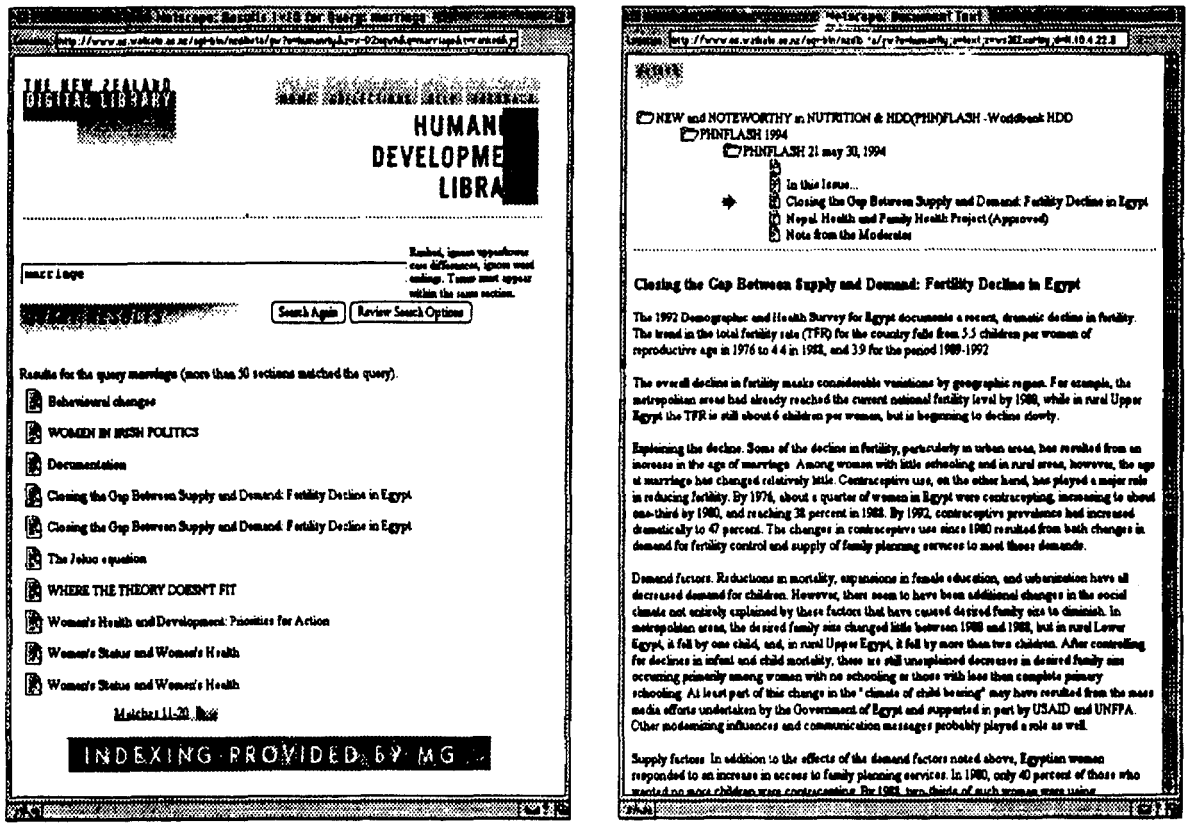

Figure 3: Results of a query for *marriage* in the Humanity Development Library

Science has already gained widespread acceptance both nationally and internationally. Although its existence has not been widely advertised, since it became available two years ago 2,700 queries have been made from 300 sites nationally, and 42,000 from about 80 countries internationally. These figures exclude the 7,000 odd queries from Waikato University. Although this level of usage is not high as compared to many Internet information services it is aimed at a rather specialised market. The service will be advertised more extensively once a North American mirror site has been set up.

The eventual goal is to produce an easy-to-use digital library system that runs at information providers' own sites. The only such site so far established is at the University (Bonn, to which an initial version of the software has been transferred. However, the system has been ported recently to the Linux operating system and it runs effectively on a modern PC configuration that is powerful (100 MHz processor, 128 Megabyte memory, 4 Gigabyte disk) but nevertheless fairly inexpensive.

Moreover, the distribution of subsets of the New Zealand Digital Library collections on CD-ROM is also being investigated. Coupled with a local Web server, this provides a powerful means whereby users without convenient access to the Web, or for whom access is unreliable or expensive, can still benefit from just the same interface and facilities.

Currently it takes a skilled person about a day or two to build each collection and integrate it into the Digital Library framework, excluding any time that is necessary together the information in the first place. We aim eventually to make it easy for users to create their own collections. However, a remarkably wide range of system requirements have been encountered. Each of the collections already created posed new problems: the information comes in different formats, the requirements of the interface are different, the indexing needs are different, the maintenance and update regime is different. Given this rate at which new problems must be solved, it seems likely that the number of systems built must triple before a

state is reached where it is possible to be reasonably confident that new collections can be accommodated within the existing framework. When that point is reached, an end user interface and tool for collection building will be tackled.

6. SUMMARY

We have described the various components that make up a digital library, covering not only the indexing and retrieval engine but also more general issues of what information might be stored and how it might be presented. We have focussed very much on two tools that have been developed during the life of this project: the MG indexing and retrieval engine; and the New Zealand Digital Library tools and facilities that make use of that engine. We hope that others will appreciate the functionality and versatility of these tools, and consider using them for their own purposes.

7. ACKNOWLEDGEMENTS

This work was supported by the Australian Research Council, the New Zealand Foundation for Research, Science and Technology, and the Natural Sciences and Engineering Research Council of Canada. We also thank the numerous people who have worked with us on the projects described here, in particular Tim Bell (University of Canterbury), Craig Nevill-Manning (Stanford University) and Justin Zobel (RMIT).

8. REFERENCES

1. Bell, TC, Cleary, JG, and Witten, IH. Text compression. Prentice-Hall, Englewood Cliffs, New Jersey. 1990.
2. Frakes, WB, and Baeza-Yates, R (Eds). Information retrieval: Data structures and algorithms. Prentice-Hall, New Jersey. 1992.
3. Harman, D. Overview of the Second Text Retrieval Conference (TREC-2). *Information Processing & Management*, 1995, **31**(3), 271-89.
4. Jayant, N (Ed). Signal compression: Coding of speech, audio, text, images and video. World Scientific, Singapore. 1997.
5. MG-software: MG public domain software for indexing and retrieving text, including tools for compressing text, bi-level images, gray scale images, and textual images. 1995. Available from ftp://munnari.oz.au/pub/mg and from http://www.mds.rmit.edu.au/ mg/.
6. Moffat, A, and Iurpin, A. On the implementation of minimum-redundancy prefix codes. IEEE Transactions on Communications, Oct. (To appear). Preliminary version in Proc. IEEE Data Compression Conference, Snowbird, Utah, March 1996, 1997. 170-179.
7. Moffat, A, and Zobel, J. Self-indexing inverted files for fast text retrieval. *ACM Transactions on Information Systems*, 1996, **14**(4), 349-379.
8. Moffat, A, Zobel, J, and Sharman, N. Text compression for dynamic document databases. *IEEE Transactions on Knowledge and Data Engineering*, 1997, **9**(2), 302-13.
9. Nevill-Manning, CG, Reed, T and Witten, IH. Extracting text from PostScript. *Software-Practice and Experience*. (To appear). 1997.
10. Salton, G, and McGill, MJ. Introduction to modern information retrieval. McGraw Hill. New York, 1994.
11. Witten, IH, Moffat, A, and Bell TC. Managing gigabytes: Compressing and indexing documents and images. Van Nostrand Reinhold, New York, 1994.
12. Zobel, J, and Moffat, A. Adding compression to a full-text retrieval system. *Software–Practice and Experience*, 1995, **25**(8), 891- 903.
13. Zobel, J, Moffat, A, and Ramamohana Rao, K. Inverted ales versus signature files for text indexing. *Submitted*. Preliminary version available as TR-95-5, Collaborative Information Technology Research Institute, Departments of Computer Science, RMIT and The University of Melbourne, Australia, 1995-96.