On-the-fly Hyperlink Creation for Page Images

Eytan Adar and Jeremy Hylton
Library 2000
M.I.T. Laboratory for Computer Science
545 Technology Sq., Cambridge MA 02141
{eytan, jerhy}@lcs.mit.edu

ABSTRACT

Hypertext is an appealing interface for digital libraries, but using existing paper documents to build such a library poses several challenges. We describe a system for creating hypertext links on the fly in a library composed of bitmapped images of paper documents and text derived from those images by optical-character recognition.

We present two simple ideas: text-image maps coordinate text and image representations of a document, and our probabilistic search heuristics generate hypertext links from the text of citations. Using the World-Wide Web, we built an interface that lets readers move from a bibliography entry to the cited document with a mouse click. Similarly, readers can click on entries in the table of contents and move directly to them.

INTRODUCTION

This paper describes an ongoing research effort to support the use of bitmapped images as the primary storage and presentation format of future digital libraries. Using images simplifies the task of creating computerized libraries of already-printed texts, but poses other complications. In particular, many tasks we expect to perform with a digital library require that the computer be able to manipulate the text of the document.

The use of bitmapped images also poses problems for one of the most basic uses of a digital library: following a reference from one document to another. A user is browsing a document — looking at images of the document pages— and discovers a reference to another work that is potentially of interest. The user highlights the reference with his or her mouse and clicks a button; the document that was referenced appears in a new window on the user's screen.

The preceding vision of a digital library — a collection of page images with a hypertext-like interface — is due to Saltzer [8]. Our work examines some of the issues involved in implementing this vision.

The use of images as the primary data format gives rise to a common question. What is the advantage of images over Postscript or SGML or some other format? We see two primary motivations for page images. First, images are a good format for long-term, archival storage for three primary reasons:

- They are good at capturing information that a text-only representation would miss, like typographic information and the placement of graphics.
- Bitmaps are also an exceedingly simple representation that will probably be understood long after Postscript is superseded by a new language.
- An uncompressed bitmap is more resistant to errors introduced by the kinds of failures, particularly media failures, that one expects in an archival system.

The second reason we choose images is that we have focused on the task of creating a digital library to complement existing libraries — and that means creating electronic versions of the millions of already-existing paper documents. The Library 2000 project plans to scan in all of MIT's computer science technical reports.

Changes in disk, network, and display technology are rapidly making it feasible to build a large library of digital images, but the problem we address is the software support that will be needed to build hypertext documents with images. First, we describe text-image maps that allow users to treat images the same way they would treat text when they highlight regions and select references. Second, we use those text-image maps to generate hypertext links between documents on-the-fly.

Several other digital library projects have used images to display documents, including CORE [2], Dienst [6], RightPages [3], and TULIP [7]. Our work is different primarily because it focuses on a closer integration between text and image than has been reported before; we link text and image at the word level instead of the paragraph level. Also, much of the published work on CORE and RightPages discusses image processing and recognition, which we do not deal with.

TEXT-IMAGE MAPS

We manage documents in two distinct formats, bitmapped images of individual pages and a text version of those pages that includes information about the position and appearance of the text on the page. Almost any operation that user would perform — highlighting a section of text or moving from the table of contents to a particular chapter — makes use of both representations of a document.

When a user sees an interesting reference and wants to get a copy, he clicks on it; the system will receive the coordinates of the user's mouse click. Before the system can attempt to locate the document the user wants, it must use the image coordinates to locate the actual text that the user clicked on. Once this text is located, the system can use the search heuristics we describe to find the document cited.

The text-image map is nothing more than a list of words, and the coordinates of their location on the page. Given a pair of x,y coordinates, it requires a simple lookup to determine what word (if any) is at that location; given a word, the coordinates of its location on the page can be looked up.

Building Text-Image Maps

We build the text-image maps using optical character recognition (OCR) software that stores both the text it recreates from an image and information about how the text looked, most importantly where the text was. Though all OCR software makes use of location information during the OCR process, ScanWorX from Xerox was the only program we identified that exported detailed position information. ScanWorX uses the XDOC data format [9] to describe page layout and other information about the structure of text and graphics.

The text-image maps we currently use are generated by a Perl script that parses the XDOC source. XDOC describes pages as a series of lines; for each line, the format provides the location of the line on the page, typographic information about the text, the actual words, and the location of spaces between the words. Based on the line's location, the height of the font, and the width of the inter-word spaces, we can reconstruct the coordinates of a rectangle that bounds the word. The text-image map is a simple listing of each word along with its bounding rectangle. On an RS/6000 PowerStation 250, it takes between a half second and a second to generate a single image map using a Perl interpreter.

Using Text-Image Maps

We have implemented a simple demonstration of the utility of text-image maps using a collection of documents from Cornell University that are available in both image and XDOC format[1].

Our system allows users to view images a page at a time using a World-Wide Web browser. We use the imagemap interface to allow users to click on a single point on the image. The service locates the single word the user clicked on and the text of the entire line clicked on. The demonstration returns the text that the user clicked on and a hypertext link that will cause the entire line of text to be used as input to our search heuristics.

The position information provided by a text-image map (or by the raw XDOC files) can also be used to identify particular features of a document and build links to them. For example, the page number can often be located on a page by looking for a number with little or no text nearby that falls at the top or bottom of the page. The extra position information also distinguishes a heading labeled "Bibliography" from an occurrence of the word "Bibliography" in the text of a document.

Our system is somewhat primitive, limited primarily by its use of the World-Wide Web as an interface. The Web interface is limiting because the browser only allows users to send the coordinates of a single mouse-click in an HTTP request; thus we can identify only the exact word and line the user clicks on, which does not always provide enough information to perform the catalog lookup. In an earlier paper [4], Hylton recommended several changes to the Web that would allow it to handle image-based documents better.

THE HEURISTIC

After extracting the (partial) bibliography string from the image we parse search requests into a machine digestible format. The parsing of natural language bibliographies has proven is a difficult task, which can be broken down into three parts: document sources, clients, and library servers.

The difficulty with document sources is that there is no standard for bibliography creation which is strictly adhered to. Different style guides recommend many different formats for bibliographies. Even within the same style guide, the citation is different for books, journals, and video tapes.

Human entry is also a problem. Even with strictly defined rules for bibliography formation, humans will take certain liberties in the creation of the text. For example, the name "Guy Lewis Steele Jr." can be written at least seven ways, from Guy Steele to G. L. Steele Jr.. The problem is compounded by the fact that the order of first and last names varies between bibliography types.

Human entry is also very error-prone. Although most published documents have been checked for accurate bibliographies, typographical errors still abound, especially in older documents. An incorrect date could result in the inability to find the document in the library database. While an obvious misspelling may be easy to catch, the appearance of 1982 instead of 1983 is more subtle. It is also possible that the difference is not actually a mistake. A paper may have been accepted for publication in 1982, but not published until 1983. Depending on when the reference was created, the bibliography will reflect this fact.

The intermediate step between image and text is also riddled with problems. Although optical character recognition technology is improving, it is not yet perfect, and most packages lack useful features such as word to location mapping. There are still a number of cases in which OCR fails to properly read a document, and typographical mistakes result. Just as with human typographical errors, it is not an easy task to counter this problem when parsing the bibliography information.

Our search heuristic functions less than optimally when parsing partial bibliographies. The limited Web interface makes this problem more pronounced.

On the server side we again encounter the same two problems as the document source. Library databases are stored as MARC records, RFC-1357, WAIS, and many other formats that are library specific.

---

[1] A demonstration can be found at <URL: http://ltt-www.lcs.mit.edu/ltt-www/Public/work.html>.

Additionally, the amount of information represented in the database is varied. For example in the RFC-1357 format [1], the only three mandatory fields are BIB-VERSION, ID, and ENTRY date. With only the bibliography information in hand, it would be impossible to ascertain if the item exists in a library whose database contains only these three fields.

Why Certain Heuristics Fail

The first instinct, to merely send the extracted text string into the library search interface, met with failure. Even after removing all punctuation marks and initials, the information supplied by the bibliography string was usually not found in the library. When looking for "word1 and word2 and word3 and ..." the slightest incongruity will cause failure. That is, if the bibliography says "University of Illinois" and the library database says "Univ. of Illinois," the search will not yield any entries.

As an experiment a heuristic was created that attempted to extract the author's name and the date from the bibliography string, because the author's name was generally unique and the date would limited the number of results when the author's name was common. For example, looking up the name "Smith" in the local reading room catalog (which contains on the order of 20,000 records) reveals 127 entries. Looking up "Smith and Mar," narrows the returned records to 10, and looking for "Smith and Mar and 1988," brings the number of records to 2. Because of the arbitrary use of abbreviations, searching the catalog for "Smith and March" yields no results.

To deal with the date problem, a database was added to the heuristic which contained possible abbreviations for month names. For September the entries were, "Sept Sep September 9." The heuristic would then iteratively search the library for the author's name, and each of the possible abbreviations for the month element in the bibliography.

The name and date approach failed not because of the complex way in which dates were extracted, but rather the way in which the author's name was determined. It was based on the assumption that the first element in the bibliography string was the author's name. Even if it was not the author's name, chances were that it was part of the title, and so was unique in that way. However, there is no guarantee that the user in clicking on the imagemap was able to select a complete bibliography entry. The line received may only contain the title, leading us to the following problem: no date can be extracted, and the first word can be a commonly used word, e.g. "the." Experiments showed that the name and date extraction heuristic performed poorly, and the idea was discarded.

We also explored using WAIS[5], or similar relevance ranking search methods. In WAIS, entries are returned and are ranked based on the number of times the search words occur in each entry. However, in order to use this scoring method it would be necessary to convert the catalog of the library to WAIS, not a minor task for any library, and not one which we attempted. The number of services that would break with a change to the library interface is too large for such a change to even be considered. However, the general idea of relevance searching is very powerful, and is similar to the heuristic we use.

A Heuristic that Works

To deal with the wide range of possibilities for bibliographic string input the heuristic needs to be randomized. Our heuristic works in the following way:

1. All stop words are removed from the bibliographic string, abbreviations and punctuation marks are deleted. These include words such as "of, the, and." In addition, words such as "computer" in a computer science library resulted in a large number of database hits. The useful feature of a stop word list is that it acts to focus the result set. Our particular service uses a stop word list that is optimized for a computer science library, and contains over 60 words. The list will change by necessity for different types of libraries. For example, we might wish to make "DNA" a stop word for a biology library.

2. From the resulting cleaned bibliography string the heuristic attempts to create at most 12, three-word sets. The words are selected randomly.

3. Each of the three-word sets is sent to the library search interface. The library database returns a list of identifiers which are unique pointers to database entries. The identifiers are "scored" by the number of times they are returned. For example: if an identifier is returned by the server as a result of ten searches, it has a score of ten.

We chose a 12/3 system for the following reasons: If the number of word sets is decreased from twelve, the scoring of the identifiers does not allow for a definite answer. However, increasing the number beyond twelve slows the search process to a speed unacceptable to users. Additionally, we do not want to choose less than three words for each set because the number of results would be high. However, we don't want to choose a number much greater then three because the logical "and" operation when used with a large number of words in a search tends to fail.

4. Identifiers are sorted by their score, top scoring identifiers first. The identifiers are handed to the library server, and the returned database entries are displayed to the user.

This heuristic has proven to be remarkably successful for a number of reasons: it is fast, it does not require any knowledge of bibliography formats, and it deals well with bibliography strings with errors.

As an example of error handling performance, the original bibliography string: "d. knuth, the art of computer programming volume 2 seminumerical algorithms 2nd edition addison-wesley 1981," was mutated to include two errors (seminumrical and algothims). Running the second (erroneous) string through the heuristic 50 times yielded the following results. The percent of tests in which the correct library record was returned as the top scoring result was

57.3. The percent of times the correct record was returned at all was 99.8. Probabilistically, it is possible that the entry is in fact in the library database but the heuristic is unable to find it because it has chosen the wrong random words.

Probably the best way to categorize the heuristic is as a "best-effort." Statistically, the results are very optimistic. However, a user may find the randomness to be discomforting. That is, running the same bibliographic string through the heuristic may yield two different results. It is therefore necessary to attach a disclaimer, that if a item is not found by the heuristic, it does not mean it is not in the library catalog.

DIRECTIONS FOR FUTURE WORK

The initial work we have done with text-image maps suggests they could be used for many tasks. A simple extension would allow users to specify a rectangular region instead of a single point, by selecting two corners in the standard click-and-drag style. This would give users greater precision in specifying text for lookup. The interface should also allow users to highlight words in an image the same way they would highlight sections of text in an editor like Emacs or Microsoft Word — by dragging the mouse over sections of an image and highlighting the text a word at a time.

In the current demonstration, we needed to identify a correspondence between image numbers, which typically start with a cover page image, and page numbers, which often begin after the table of contents or acknowledgments. It is not unusual for a document to start with 10 unnumbered pages, so page 1 is actually image 11. Our simple search heuristic identified page number as a number with no other text on the same line that fell at either the top or bottom of the page.

From the heuristic viewpoint there is also room for improvement. The 12/3 combination of sets and words was seemed to yield the best results. By adjusting the numbers of sets and words we determined the most useful balance between accuracy and speed. However, the tests were done on a rather stable group of bibliography strings. We know from the experience of pulling text from images that the assumption of receiving useful information from the reference is a false one.

It is probably here where we can most improve the heuristic. No matter what the size of the bibliography string, the heuristic will still use twelve word sets, three words each. It may be of benefit both in speed and accuracy to use a variance system. By varying the number of word sets and words(within some limits) depending on the number words in the bibliography string, it may be possible to attain this improvement.

The second improvement can be obtained on a higher level. By improving the quality and quantity of material retrieved from the images the heuristic will greatly benefit. In its current state a user can click on the wrong line of text in the image and get the line "2nd edition, Addison-Wesley 1981." The heuristic works best when it is given "quality" information. So a string such as, "D. Knuth, The Art of Computer Programming volume 2 Seminumerical," will work far better then the line described above.

REFERENCES

1. Cohen, Danny. A Format for E-mailing Bibliographic Records. Internet RFC 1357, July 1992.

2. Egan, Dennis E. et al. Hypertext for the Electronic Library? CORE Sample Results. In *Proceedings of ACM Hypertext '91,* 1994, pp. 299-312.

3. Hoffman, Melia M. et al. The RightPages Service: An Image-Based Electronic Library. *Journal of the American Society for Information Science.* 44,8 (Sept. 1993), 446-452.

4. Hylton, Jeremy. Text-Image Maps for Document Delivery on the Web. Workshop on HTML+ at the First International Conference on the World-Wide Web. CERN, Geneva, May 1994.

5. Kahle, Brewster. Wide Area Information Server Concepts. Version 4, Draft. Thinking Machines.

6. Lagoze, Carl and James R. Davis. Dienst: An Architecture for Distributed Document Libraries. *Communications of the ACM.* 38, 4 (April 1995), 47.

7. Mostert, Paul. TULIP: Specification of the Data Structure and of the Delivery Methods. Amsterdam, Elsevier, 1994.

8. Saltzer, Jerome H., Technology, Networks, and the Library of the Year 2000, in *Future Tendencies in Computer Science, Control, and Applied Mathematics, Lecture Notes in Computer Science 653,* edited by A. Bensoussan and J.-P. Verjus, Springer-Verlag, New York, 1992, pp. 51-67.

9. XDOC Data Format: Technical Specification. Xerox Imaging Systems part no. 00-07571-00.

ACKNOWLEDGMENTS

The authors would like to thank Mitchell Charity and Jerry Saltzer for their advice and comments on the text-image map, the citation look-up system, and this paper. The text-image map was designed and implemented by Hylton; the citation look-up system was designed and implemented by Adar.

This work was supported in part by the IBM Corporation, in part by the Digital Equipment Corporation, and in part by the Corporation for National Research Initiatives, using funds from the Advanced Research Projects Agency of the United States Department of Defense under grant MDA972-92-J1029.