

A Document Corpus Browser for In-Depth Reading

Eric Bier, Lance Good, Kris Popat

Palo Alto Research Center

3333 Coyote Hill Road

Palo Alto, CA 94304

650-812-4000

{bier, good, popat}@parc.com

Alan Newberger

UC Berkeley

387 Soda Hall

Berkeley, CA 94720

alann@eecs.berkeley.edu

ABSTRACT

Software tools, including Web browsers, e-books, electronic document formats, search engines, and digital libraries are changing the way that people read, making it easier for them to find and view documents. However, while these tools provide significant help with short-term reading projects involving small numbers of documents, they fall short of supporting readers engaged in longer-term reading projects, in which a topic is to be understood in-depth by reading many documents. Such readers need to find and manage many documents and citations, remember what they have read, and prioritize what to read next. In this paper, we describe three integrated software tools that facilitate in-depth reading. A first tool extracts citation information from documents. A second finds on-line documents from their citations. The last is a document corpus browser that uses a zoomable user interface to show a corpus at multiple granularities while supporting reading tasks that take days, weeks, or longer. We describe these tools and the design principles that motivated them.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces – *graphical user interfaces*. H.3.7 [Information Storage and Retrieval]: Digital Libraries – *user issues*. H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *search process*.

General Terms

Algorithms, Design, Human Factors.

Keywords

Computer-aided reading, visualization, bookplex, digital library, document management, spatial memory, zoomable user interface

1. INTRODUCTION

We are interested in the problem of creating software tools that aid people in long term and in-depth reading. Considered broadly, the tasks of in-depth reading include articulating a topic

or information need, deciding what documents to put on one's reading list, acquiring these documents in readable form, viewing the documents, adding marks and annotations to the documents, deciding which documents to read in what order, tracking which documents have been read and which are still to be read, expanding the list of documents on one's reading list, and iterating, until the reader has gained a desired level of understanding. This cycle of reading sub-tasks is illustrated in Figure 1.

The large ovals of Figure 1 show that a user is interacting both with a collected set of relevant documents and (via search engines, Web browsers, etc.) with a much larger set of documents that are available but not yet collected. The inside of the diagram shows in-depth reading as an iterative sensemaking task, in which the sub-tasks of searching, acquiring, managing, reading, and understanding documents occur repeatedly as the user makes sense of a topic or satisfies an information need.

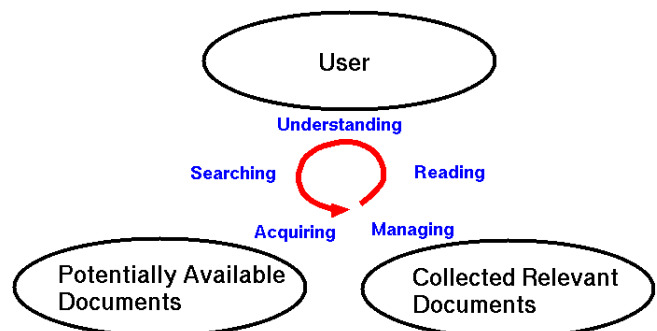


Figure 1. The iterative process of in-depth reading.

Recent software technologies have made some of the tasks of reading faster, easier, and more comprehensive. Search engines, digital libraries, and search result visualizations [1][5][7][15][18] have made it easier to determine what documents are available. Web browsers, e-books, and document formats such as HTML and Adobe's Portable Document Format have increased the speed at which users can acquire a document in readable form. Research technologies like XLibris [17] and commercial technologies like the Tablet PC have made it possible to annotate documents while reading them on-screen.

While these tools provide significant help with short-term reading projects involving small numbers of documents, they fall short of supporting readers who are engaged in longer-term reading, in which a topic is to be understood in-depth by reading many documents. This activity has been called *syntopical* reading [19].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

JCDL, June 7-11, 2004, Tucson, AZ, USA.

Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

Some difficulties encountered in dealing with the larger numbers of documents involved in syntopical reading include:

1. Acquiring documents from their citation information.
2. Managing collections of documents and their citations.
3. Finding important sub-collections of documents within a larger collection.
4. Deciding which documents to read first in a large collection.
5. Tracking which documents have been read.

Many existing systems have addressed these problems individually. However, we believe that a substantially better syntopical reading experience can be achieved by combining solutions to the entire set of problems. In this paper we describe a system called *Instant Bookplex* that provides tools to address each of these problems. The term *bookplex* was coined by Stuart Card and is described in a paper by Woodruff [20]. A bookplex is intended to suggest a collection of documents that behaves in some ways as a single “book”, but is in fact composed of many separate documents. The user of Instant Bookplex chooses one or more documents that are likely to be central to the topic of interest, or *seed documents*, and places them in a personal digital library known as UPLib [9]. By interacting with UPLib and other tools, the user adds additional documents to the digital library, including documents cited by the seed documents or cited by any other document in the library. At any time, the user can examine the contents of the library in a zoomable browser that displays information at different levels of granularity from the entire corpus, through sub-collections of documents, to pages of individual documents. Instant Bookplex uses colors, shapes, images, and text to show important document properties in ways appropriate to each level of zoom. The resulting user interface helps the user track the reading process, even if it takes weeks, months, or longer.

The rest of this paper is organized as follows. First, we describe related work. Next, we describe some of the design desiderata that motivated our in-depth reading tools. Third, we describe two tools that help a reader assemble a collection of documents on a topic, by automating some of the work of following citations. Fourth, we describe a third tool, a zoomable user interface (ZUI) [14] browser that helps the user become familiar with the documents in a particular bookplex. Fifth, we describe our implementation of all three tools. Sixth, we discuss some of the consequences of our design. Finally, we present our conclusions and plans for future work.

2. RELATED WORK

The system proposed here conceives of, and attempts to address, the problem of in-depth reading in its entirety from a user's perspective, and in so doing also proposes specific approaches to its subtasks. It can therefore be compared to other work at two distinct levels: in its support of the whole activity of in-depth reading, and in its approach to each sub-task.

We first consider support for in-depth reading as a whole. As we conceive of it here, in-depth reading fundamentally involves

reading from multiple sources, where the multiple sources are typically obtained by beginning with one or a few documents known to be highly relevant to a knowledge need, then growing a set of related documents from these. Thus described, the current de-facto standard system for supporting this activity in the electronic realm appears to be the World Wide Web itself, accessed through a browser, and considered here as comprising its available search engines, accessible on-line journals and publishers' portals, authors' homepages, and notably and importantly, the CiteSeer [5] system. Current practice around using the World Wide Web for in-depth reading is labor-intensive in ways that computers can and should specifically address – general clerical tasks, the recommendation and retrieval of related material, the managing of collected documents, and the moving of attention back and forth among the collected documents as they are read. While technologies such as CiteSeer help tremendously with some of these component tasks, integrated support for the activity as a whole is missing. The present work attempts to provide such integrated support, and does so largely by automating and improving upon the manual practice that has emerged around using the World Wide Web for in-depth reading, spanning all phases of the activity, and leveraging existing components when advantageous.

Our system design has also been influenced by studies of people engaged in multi-document reading tasks. The study by O'Hara et al. [12], for instance, reveals the extent to which readers switch their attention back and forth between multiple documents and return frequently to documents that have been read previously.

Other systems have been devised to address the reading task as a whole from a user's perspective, but often such systems focus on reading of a single document, or when applied to a collection, of one document at a time. The XLibris system [17] is particularly notable in that it considered the gap between the actual practice of reading and what was supported in then-current electronic technology, and attempted to bridge that gap in large part by making electronic reading more paper-like. In the terminology of Adler and Van Doren [19], it might be said that XLibris targeted the activity of *analytical* reading of individual documents, while the present system targets *syntopical* reading around a set of tightly related documents. The specifics of the functionality therefore differ. For instance, XLibris was never intended to provide the ability to browse and manage collected documents, nor provide an integrated means of finding and retrieving documents by following the links in a citation graph. Ultimately, XLibris-like functionality such as support for pen-based annotation can and should be available as a subset of a system for syntopical reading, because analytical reading is a component activity of syntopical reading. Currently, the functionalities provided by the proposed system are largely complementary to those of XLibris rather than a superset, but this describes merely the current state rather than a complementarity induced by principle, as will become evident in the Discussion section.

Many other systems and technologies have been proposed for effectively reading single works; these include various e-Book systems and electronic document formats with their associated viewers. The distinctions in functionality cited above between the proposed system and XLibris apply to many of these as well.

Prior work exists for each of the sub-tasks of in-depth reading; we now consider what we believe to be the most relevant for each. In

the finding and retrieving of specific documents related to a given one, the standard Web search engines and CiteSeer stand out. Indeed, some of the sub problems addressed in our system, such as the identification and parsing of bibliographic entries, have been also been addressed within CiteSeer. Here, in having likely duplicated some of CiteSeer's internal functionality, we achieve two ends: an improved and robust ability to find and retrieve documents on the web, *including* from CiteSeer itself, as well as the ability to support the integrated activities around linking, organizing, and browsing the collected documents. Other prior work on finding citations of relevant documents includes the multi-agent and rule-based graph-search approaches proposed by Han et al. [7]; see that paper for a summary of early prior work in this area. Another system for finding documents worth mentioning here is SenseMaker [1], which has several features in common with ours: the provision and abstraction of web-based and other remote services through a proxy mechanism, the ability to organize and manipulate retrieved references in clusters, and the ability to expand outwards around a set of references to obtain more, by automated query generation. The systems differ in the scope of task for which they are intended: SenseMaker is concerned primarily with obtaining relevant documents, which is one component of in-depth reading. For the representation, storage, and retrieval aspects of managing the collection of retrieved documents, as previously noted, we have built on the UpLib system [9], which was specifically designed to be extended. Many other systems have been proposed for these tasks; several are reviewed in the Uplib paper.

Within any reasonably large set of collected documents, sub-topics and other bases for sub-categorization and clustering will be evident. Often there is considerable advantage in organizing and accessing the collection accordingly. Several prior efforts have considered the problem of identifying and interfacing with sub-collections in an automated or semi-automated way. Nowell et al. [11] describe visualization strategies for collections based on laying out arrays of document icons using various graphical properties including layout and color; these characteristics are shared by the system proposed here. Rabuer and Müller-Kögler [15] propose using the document's inferred genre to determine sub-collection membership, based on structural features clustered by a self-organizing map. Shneiderman et al. [18] consider the organization and visualization of very large collections of documents by combining grid displays with hierarchical organization along axes with controls for zooming on the grid. Our system also allows zooming into document categories, but does so using smooth zooming and provides more informative document icons so that users can more easily form spatial memories of the document space.

Deciding on which papers to read first can be regarded largely as a recommendation problem (a refinement would be to compensate for interdependence and redundancies among those highly recommended). McNee et al. [10] propose applying a collaborative filtering technique to the citation graph among papers in a collection, making an analogy with social networks among people. Woodruff et al. [20] also consider the citation graph, and obtain recommendations by applying a spreading activation technique to simple quadratic forms involving the graph incidence matrix. In the present work, we assume that the documents have already been rated, either using an algorithm like the one reported in Woodruff et al. or by a user or users.

One approach to the sub-task of keeping track of what has already been read is suggested by Hill et al. [8], and another is suggested by Dumais et al. [4]. In the former work, an analogy is made to a physical artifact whose condition reflects the degree to which it has been worn down by use, and the wear pattern is depicted graphically in a part of the interface where that information is most helpful. In the latter work, a uniform search and indexing facility is provided for all documents of interest. The user's ability to recognize items previously read is facilitated by presenting graphically appropriate contextual cues to jog the user's memory. In the present system, some of the same effect is undoubtedly achieved by presenting a page thumbnail in the document icon, but our system goes further by explicitly encoding the degree to which the document has been read in the color of the icon, as will be discussed in subsequent sections.

3. DESIGN DESIDERATA

In this section, we describe first the design desiderata we apply to tools that find and organize documents. Next we describe design desiderata that we apply to tools for viewing and navigating the documents that have been found.

The process of gathering materials for in-depth reading is complicated by many practical issues. Some materials are available on-line, others on paper, and still others may be hard to find at all. The interests of the reader evolve as reading proceeds. The reader discovers new documents even before all of the initial documents have been read. Some materials are available for a fee and others are free. Documents vary widely in length and importance to the topic. Errors may be present in documents and their citations.

As a result, it is neither possible nor desirable to completely automate the processing of finding and organizing documents. Any software tools must be interactive, allowing the user to provide frequent guidance. For example, the user should be able to decide which version of a document to include, whether to scan in a version from paper, whether a given document should be included or not, whether the rating of a document should be changed to better reflect the user's interest, whether or not to buy an on-line copy of an expensive document, and whether or not to correct a spelling error in a citation.

As we designed Instant Bookplex, we tried to strike a balance between automating some of the process, while leaving the reader in control. We have adopted these desiderata for tools that help the user gather documents:

D1. The user must have ultimate control over which documents are or are not included in a bookplex.

D2. Any data derived from automatic processing of documents and citations must be able to be discarded or revised by the user.

In designing the browsing user interface of our system, we focused on two aspects of in-depth reading that distinguish it from other kinds of reading: the *greater number of documents* involved and the *longer time periods*. When many documents are involved, the reader has to be selective about which to read. Ideally, the selection process will take into account the entire collection and the history of the reading process. Furthermore, with many documents to read, the user cannot spend much time or

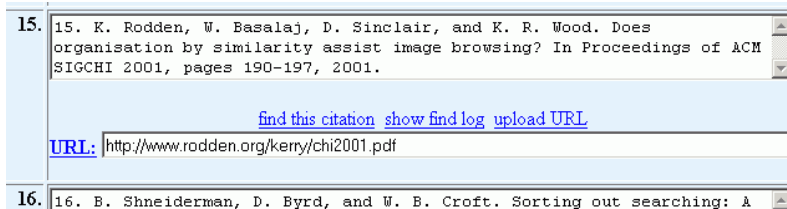


Figure 2. Web form for editing extracted citations.

thought navigating from one document to the next; transitions should be at least as quick and easy as they would be for someone reading a collection of paper documents arrayed on a desk or conference table. Likewise, it should be quick and easy to find a previously-viewed document. Finally, when making sense of a large collection, it may be necessary to divide the collection into groups, sub-groups, and so forth so that the reader consider the collection in chunks of manageable size.

Because the process proceeds over long time periods, the reader may not wish to rely on human memory alone to remember which documents have been read nor to remember where to find the highest quality document groups. On the other hand, the longer time period means that the reader may be willing to invest time to become familiar with a particular graphical presentation of the document collection if that presentation can be used repeatedly to aid the reading process. These considerations lead us to these additional desiderata:

D3. Visualizations of a bookplex should reveal information at several levels of granularity, from individual documents to all documents in the bookplex.

D4. Transitions from document to document or from sub-collection to sub-collection should require little time and little effort.

D5. Visualizations of a bookplex should be stable enough and easy enough to navigate that the user can return to previously-viewed documents and document sub-collections using spatial memory.

In the next two sections, we present our design and show how it follows these desiderata.

4. CREATING A BOOKPLEX

Having identified a topic or information need, the user identifies a small set of documents that are judged to be on-topic and places these documents into the UPLib personal digital library [9]. UPLib performs some initial processing on the documents, including computing a screen-resolution image and a smaller thumbnail image of each page of the document. UPLib also associates a collection of metadata fields with the new document and provides a Web interface for editing this metadata. Using this Web interface, the user can mark the document as a seed document, rank its importance, and indicate how much of the document the user has read.

Next, the user can ask UPLib to extract the citations from the document's references section. A data extraction algorithm runs that finds the references section and the individual citations in it. The extracted citations are then displayed for the user in a Web form that allows the user to add to, delete from, or change the

information produced by data extraction (in accordance with criterion D2). An example web form is shown in Figure 2.

In the web form, the user can read the citation list. If a particular citation looks promising as a possible addition to the bookplex, the user clicks the "find this citation" button. Using a combination of search engines and digital libraries UPLib tries to find a copy of the document. It places the best URL that it found in the URL area below the citation of interest. The user can visit the URL to see if it is the correct document and to evaluate its relevance. If the document is not correct, the user can search for a better one and place the better URL into the URL field (again in accordance with D2). If the document is correct and relevant the user may press the "upload URL" button to add the document to the bookplex (in accordance with D1).

By repeating this process on other citations in the seed documents and then on the citations in the cited documents, and so on, the user can build up a bookplex of documents that are part of a common citation graph. Note that this process is largely automatic, while allowing the user to override elements of the automatic process when necessary. As a result, an on-topic document collection can be created more quickly or a larger collection can be created in the same time than would be possible without this process.

For each document added, the user can rate the document and indicate how much of the document has so far been read. This information is used by the corpus visualization, which we describe next.

5. EXPLORING THE DOCUMENTS

A user who has assembled a collection of documents will then wish to view the collection and learn more about it. In particular, the user may want to identify the important authors, years, publications, and sub-topics of the collection. The user will also want to decide which documents to read and in what order. Finally, the user will want to read some of the documents.

In Instant Bookplex, both exploring the bookplex and reading individual documents are done through a ZUI, called the *corpus view*. Because the corpus view emphasizes different information depending on the level of zoom, we will organize our description of the corpus view by zoom level, describing first the most comprehensive view, next the most detailed view, and finally the intermediate views.

5.1 The Comprehensive View

The corpus view provides its most comprehensive picture of the bookplex when it is zoomed out to show all of the documents in the bookplex. An excerpt of a comprehensive view is shown in Figure 3 for a bookplex of about 50 documents. In this view, each document is represented by a small icon and a text fragment. The document icons are sorted into categories and laid out in two dimensions. Different colors indicate the rating of each document and how much of it has been read. If the document is available in the digital library, a thumbnail image of its first page is displayed as part of the document icon. Note that even in the comprehensive view enough information about the first author and date of the paper is given that it may be possible to identify it.

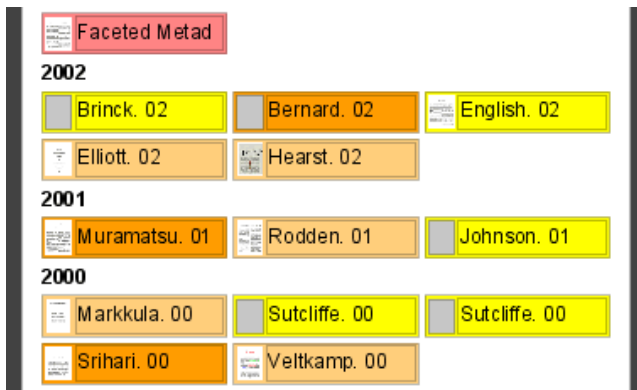


Figure 3. Excerpt from a comprehensive view of a bookplex.

Clicking on an object in this view zooms the view so that the selected object fills the corpus view window in the horizontal direction. One can click on the entire column (shown in white), on an entire citation, or on a thumbnail image of a document to zoom in to these objects. Clicking on the background zooms out to show the entire corpus. Moving the mousewheel allows zooming to arbitrary zoom levels. The resulting interface supports examining the corpus at arbitrary granularities as required by D3.

Wherever possible, the zooming operations use animation so the user can easily perceive the relative position of icons in the initial and final views. Together, the simple operations for zooming in and out and the animated operations support rapid transitions as required by D4.

5.2 Individual Documents - TATs

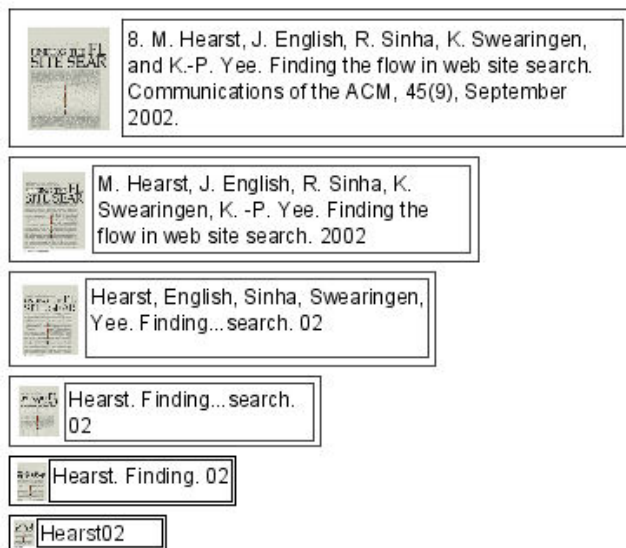


Figure 4. A TAT at different zoom levels.

At all zoom levels, individual documents are displayed as icons. Icons in the system are called TATs, containing Text and Thumbnail views of documents. The text portion of a TAT displays readable document metadata structured as a citation. The thumbnail provides a view of the first page of the document, which is displayed at low resolution when the TAT is small, for efficiency, and at full readable resolution when the user zooms in.

Both the text and thumbnail portions of a TAT are designed to operate well in a ZUI environment, as shown in Figure 4 and described further below.

When a TAT is viewed at different zoom scales, its area changes. In order to preserve readability over a wide range of scales, the text portion of a TAT utilizes semantic zooming. As the total area of a TAT reduces in size, the font size of its text is held constant and the text itself is pruned to fit the area. The TAT text is based on structured citation metadata, so the semantic zooming algorithm can choose from the fields of the citation which fields (or portions of fields) to display at each scale.

As shown in Figure 4, the maximum text supplied by the algorithm is a full citation including authors, title, publication name, volume, and date. When area is constrained, individual fields are either truncated or omitted to shorten the text. For example, authors are truncated first by including only last names, and then by showing first author only. Titles are first displayed in full, and are pruned by replacing certain less significant words with ellipsis. Currently, a simple significance estimation heuristic based upon word length is used. Year is pruned from a full four-digit representation to a two-digit one. Publication name is shortened by removing less informative words such as "Proceedings", "Conference", and "International" or omitted altogether. Volume, number, page numbers, and other fields are simply omitted when area is constrained. A full citation is semantically reduced in this fashion to a minimal representation indicating primary author last name and publication year.

The TAT text font size is maintained constant as zoom levels vary, and across all TATs in a corpus. The effect is a uniform readable text size over all TATs, as shown in Figure 3. However, when all the TATs in a corpus are at a scale sufficient to display full citation text, any further enforcement of a fixed font size would only result in increased white space within a TAT as the zoom scale increased. Therefore, at that point font size is allowed to vary and is free to scale with the zoom scale. This allows for a more natural look for TAT text portions when zoom scales are high.

The Thumbnail portion of each TAT captures an actual page of the represented document. Multiple resolutions are utilized so that the thumbnail is rendered rapidly at low zoom scales and during zooming, and can be rendered with detail at high zoom scales. At low zoom factors, the thumbnail helps the reader identify the document and its genre, as documents often contain distinctive layout factors on their title pages.

When zoomed in completely, users can read the document content from the thumbnail (which then becomes less of a "thumbnail" and more of a document viewer). In this case, the zoom factor is chosen carefully so that the pixels of the page image map one-to-one onto screen pixels, producing the highest image quality for reading. The page image itself was computed earlier by the UPLib system using careful image processing so as to produce text that is readable on screen. Interactors are provided for flipping through the pages of the documents backwards and forwards, so the user can read the entire document. When the user zooms out again, a thumbnail image of the *current page* is displayed, which is not necessarily the title page, thus reminding the user where reading left off. For example, in Figure 5 below, the document page image for "Untangling text data mining"

shows page 7 of the document instead of the title page. The next time the user zooms in to this document that same page will be displayed so reading can resume without further page flipping.

Other physical characteristics of a TAT are used to convey additional document and corpus information. In our current implementation, the background color of each TAT indicates either the rating of each associated document or (as in Figure 3) the number of links that must be followed in the citation graph to arrive at the nearest seed document. The saturation of the color shows whether the user has previously read that document or not.

5.3 Groups of TATs

TATs can be grouped spatially in order to reveal category structures in the bookplex. Some systems for organizing search results [11][18] organize documents into a grid structure in order to display two category axes simultaneously. One problem with this approach is that some grid cells are inevitably more sparsely filled than others, in which case significant amounts of screen space are wasted. We solve that problem in this system by using a layout idea borrowed from interactive document editors. Document icons are organized into virtual “paragraphs” that share a common width, but vary their height to accommodate a greater or lesser number of icons.

In Figure 3, the documents are sorted into categories based on date of publication. The algorithm treats TATs as “words” in “paragraphs” representing discrete axis units or categories, *e.g.* publication year, and are all organized onto pages that contain all TATs in a corpus. Each of these paragraphs is labeled with a section header. This layout provides a rich set of cues based on ordering and categorization to support the relocation of documents. That is, in addition to using information within a TAT to locate it, users can locate a TAT by its position in a page, its relative position in a category, or its proximity to other TATs.

This layout strategy is advantageous because it maintains its properties as a document corpus grows over time. Borrowing from the metaphor of a reflowable document that changes the wrapping of words but preserves their ordering, the layout can reflow TATs as new ones are added to the corpus, but general location on a page and within a category is largely preserved. Thus, like Data Mountain [16], our system takes advantage of spatial memory, but we allow some local movements of icons (within paragraphs) and some stretching of the visualizing in the y direction (as paragraphs grow taller) in order to allow for automatic adjustments to the layout as new documents are added. The resulting visualization provides a semi-stable workspace to support D5.

6. IMPLEMENTATION

This section describes the implementation of our three tools for in-depth reading. We describe the reference extraction tool first, then the document finding tool, and finally the corpus browser.

6.1 The Reference Extraction Tool

The reference extraction tool takes a document file and produces a list of the references that are cited in that document. It first decomposes the document format into a stand-off markup data structure containing raw text and formatting information. The tool then applies an analysis pipeline that segments the document into sections, locates the reference section if it exists, and extracts the

actual references. Finally, the tool stores the resulting references as metadata in UPLib associated with the citing document.

Documents submitted may be in either PostScript or PDF format. The tool uses a PDF parsing library based on the Multivalent system from UC Berkeley [11] that constructs an in-memory tree representation of the document (PostScript files are pre-converted to PDF), and converts that representation into a more convenient stand-off markup data structure. The stand-off markup provides access to the raw text of the document, and enables queries to retrieve annotations on spans of document text. Annotations generated by the tool include physical attributes of the document such as font type and size, paragraph breaks, page location, and line indentations.

The text and physical attributes of the document are used for further analysis to detect higher-level constructs including document sections and references as suggested by Giuffrida et. al [6]. Document sections are detected by finding section headers. First, all eligible section headers are identified, using criteria such as single-line paragraphs, differences in font size, and the existence of numerical prefixes. Candidate section headers are then screened to form a single, consistent set of section headers. Section breaks and headers are added to the stand-off markup data structure as annotations.

Once document sections are identified, the reference section is identified by a set of heuristics that look closely at the last section of the document, and for sections that are titled "References", using the annotations generated by the previous stages. The content of that section is then analyzed for references. A variety of mechanisms are used to identify individual citations in the section text, including identifying numbered lists, and analyzing the margins and indentations of the lines in the section for common indentation patterns.

The reference extractor is good enough to be useful, but imperfect. For example, we ran the extractor on a set of 30 PDF documents selected at random from the ACM Digital Library. For 60% of the documents the extractor finds the correct number of citations or is off by only one. For the remaining 40%, it makes larger errors. The extractor will become more accurate in the future. However, because the user can correct its output in UPLib's user interface, extraction errors only reduce the automation available to the user; the desired bookplex can still be created.

6.2 The Document Finding Tool

The document finding tool takes a citation as input and tries to find a copy of the cited document on the Internet.

One approach to solving this problem is to build robots that extract information from on-line document repositories, such as CiteSeer, the ACM Digital Library, or PubMed. We rejected this approach because each on-line repository has a limited offering of documents, and we wanted a tool that was flexible enough to find scholarly documents on virtually any topic.

Instead, we built a system that mimics strategies people use when finding documents on the Internet. In particular, this tool begins with the citation and parses it into data fields, such as author names, title, year of publication and so forth. Using these data fields, it constructs a set of queries on a general-purpose search



Figure 5. A group of TATs at an intermediate scale factor

engine. It then downloads the documents found by the search engine and examines each one to determine its actual title, authors, and page count. It compares this extracted information with the citation fields for each document found. Finally, it produces as output the URLs of any documents that match the desired citation to within a small error tolerance.

As described above, this tool can be run from a web form in the UPLib digital library, so the user can apply it interactively to any citation on the web form, or to all of them at once.

The process of constructing search engine queries begins by parsing the input citation into fields and extracting the author last names and the title. Common words are removed from the title using a stop word list. Then several queries are generated and sent to the search engine in decreasing order of specificity. The first query looks for those documents that contain all of the author last names, that contain all of the non-common title words, and that are in PDF format. If this query fails to find acceptable documents, then additional queries are sent that are less specific. For example, these queries may omit the author names (in case they were misspelled in the citation), omit the non-English words of the title (in case they resulted from OCR errors) and allow formats other than PDF, such as PostScript and HTML.

The top-rated documents found by the search engine are then downloaded. Their text is extracted, together with formatting information such as bounding boxes of words, typeface changes, font size changes, and page boundaries. Data extraction rules are then used to find the actual title and author names, if available.

The extracted title, author names, and page count is then compared to the input citation and the closeness of match is evaluated. For authors and title, the closeness of match is computed using a Levenshtein edit distance algorithm that has been modified to match the fields of the input citation to any substring of the corresponding field from the document being evaluated. Thus an author name can be matched no matter where it appears in the list of authors, and a title can be matched even if data extraction matches a region that includes more of the title page than just the title.

Like the reference extraction tool, the document finding tool is usable but imperfect. For example, when we run this tool on all 18 citations in the seed document of Figure 3, the document finding tool finds useful URLs for 28% of the citations. This tool will find a greater percentage of documents in the future. However, it will probably never be able to find as many of the documents as a skilled human user, who can try more search strategies, pay for material that is not free, provide passwords to members-only repositories, and scan in documents that are only available on paper. As a result, we will retain and extend facilities that allow the user to provide documents in addition to facilities that find them automatically.

6.3 Corpus Browser

In this section we report on three aspects of the corpus browser: its zoomable user interface, the semantic zooming technique used to compress citation text, and enforcement of a common font size.

6.3.1 Zoomable user interface

The corpus browser is implemented in Java on top of the Piccolo graphics toolkit from the University of Maryland [2]. Piccolo is a good match for this application, as it supports both the zooming needed to show the document corpus at different levels of detail and also the ability to display varying object representations, with different document page images and different citation text strings, at different zoom levels.

Navigation operations in the two-dimensional space of document icons may be triggered by positioning the cursor in the space and clicking the right mouse button. If a click is received on a document icon (TAT) inside a rectangle that contains citation text, on a document thumbnail, or on a page of TATs, the whole space is zoomed until the object that received the click is zoomed to nearly fill the width of the browser window, centered both horizontally and vertically in the window.

The change in position and size of the two dimensional space is animated, using a built-in slow-in and slow-out animation technique that starts slow, accelerates to a constant velocity in the middle of the motion, and then decelerates to a stop at the end. These types of viewpoint animations have been shown to help users build mental maps of spatial information [3].

6.3.2 Semantic zooming of citation text

Whenever a TAT is drawn on the screen, it decides what information to show based on the current scale factor. In our implementation, we distinguish between four ranges of scale factor: *full size*, *compression needed*, *small width*, and *small height*. In each range, a different computation is used to determine the appearance of the TAT text.

In the *full size* range, there is more than enough height and width in the text rectangle to fit the full citation text. As a result, an increased font size can be used when displaying the text. In the *compression needed* range, there is plenty of height in the text rectangle for one or more lines of text, and there is enough horizontal space for some version of the citation text, but there is not enough horizontal space for the full citation; in this case semantic zooming is used to compress the citation text. In the *small width* range, there is still enough height for one or more characters in a readable font, but there is not enough horizontal space to include even the shortest version of the citation that semantic zooming produces; in this case the shortest version of the citation is truncated as needed by removing its rightmost characters. Finally, in the *small height* range, the height of the text rectangle is too small to contain any text of the desired font size. In this case, we compute the text as in the small width range and then shrink the font size, even though this will make the text less readable.

In our current implementation, the citation text of each TAT has as many as 15 semantic zoom levels, each with a different number of text characters. With this many text lengths to choose from,

the semantic zooming algorithm can do a good job of finding an appropriate amount of citation text to display at each zoom factor. For example, Figure 5 shows a group of four TATs with different levels of semantic zoom. Some show the title in full; in others it is abbreviated. One shows all authors names in full, one shows last names only, and two show only the first author's last name.

The algorithm for generating the text variants could potentially be complicated. We wanted it to reflect as well as possible our intuitions about which parts of a citation are more important and which are less important to show at each zoom level. To keep the code simple and flexible, we decomposed the text variant generator into three component types. The first type of component generates one of the fields of the citation at a fixed number of lengths. For example, the AUTHORS component generates the authors field of a citation at six levels, call them A1-A6. A1 shows the first three letters of the first author's last name, A2 shows five letters, A3 seven letters, A4 all letters, A5 shows the last names of all authors, and A6 shows the full names of all authors.

The second component type is an array that indicates how each of the 15 semantic zoom levels differs from the previous level. For example, the first six elements of this array might be: A1, Y1, A2, A3, A4, and T1, where Y1 comes from the year of publication component, and T1 comes from the title component. This array would indicate that the shortest textual variant contains only A1. The next shortest contains A1 and Y1. The third text variant contains A2 and Y1 (since A2 replaces A1). Continuing in this way, the sixth variant contains A4, Y1, and T1, meaning that it will contain the 4th shortest string of author names, the shortest year of publication string, and the shortest title string.

The third component type indicates the order in which the text fields will be displayed. In our system, the author string precedes the title string (if any), which in turn precedes the year of publication string.

6.3.3 Enforcement of common font size

When implemented in the most straightforward way, the rules for determining the text appearance of TATs will result in different TATs displaying text in different font sizes. This follows because a scale factor that is large enough to show the entire citation text in one TAT, may force the citation text to be compressed in a different TAT (e.g., one with a longer citation). The resulting display appears inconsistent and draws the eye towards TATs with larger font sizes independent of their actual importance.

To solve this problem, the corpus view coordinates display across TATs, so that a single font size is used across the entire display. The coordination is performed once all of the TATs are known, by computing the minimum scale factor at which all TATs can display at least two text characters and the minimum scale factor at which all TATs can display their full citation text. Individual TATs then refrain from adopting a larger font until the scale factor is large enough to allow all TATs to use the same larger font.

7. DISCUSSION

Our goal in this work is to improve people's ability to do in-depth reading over what is possible with traditional processes both paper and electronic. The technologies we have developed

achieve this goal in two ways: by reducing the time needed to perform some of the steps in the process and by eliminating some steps altogether. In particular, the Instant Bookplex system reduces the time needed to:

- find a document once its citation has been located;
- locate the most important documents and document patches to read next;
- transfer attention from one document to another; and
- keep track of which documents have been read.

It also:

- eliminates the need to create a personal bibliography as a separate step.

In this section, we consider each of these five consequences of the design.

Finding documents from citations. People who read scholarly papers often encounter citations at the end of these papers that look interesting. When such citations are encountered the reader must make a difficult decision. Is it worth the time to actually find a copy of the paper? If the reader has found many interesting citations, which should be obtained first? Once some of the papers have been obtained, which should be the first to receive a more thorough reading? By providing automated tools to extract the reference list at the end of a document and then to locate on-line copies of the corresponding papers, Instant Bookplex reduces the time cost to the reader of obtaining cited works, and so makes it more likely that the reader will actually find the most relevant papers.

Which documents and document patches to read next. While on-line document indices like CiteSeer provide an excellent view of the documents in the immediate citation neighborhood of a given document, they do not provide a picture of the larger set of documents on a given topic. The reader must piece together this view manually by submitting multiple queries and combining the results. The decision of which documents to read first, if it is to be made properly, should be based on identifying the best documents that have been found across all queries.

By gathering all of the documents found on a topic into a single personal corpus, Instant Bookplex automates the work of combining results. By providing a zoomable corpus view that graphically distinguishes the highest rated documents, Instant Bookplex makes it possible to see at a glance which patches of documents contain the highly rated documents so that the reader can focus attention on these patches first.

Moving from document to document. Modern document tools have made it easier to view several documents in sequence. Click on a link in a Web browser and an application window opens to view a document. Click on another link and a second application window appears showing a second document. The reader can then move back and forth between the documents using window system operations. However, as the number of documents grows, this approach becomes unwieldy. It becomes hard to find a desired document among a long list of similar icons. Getting from one document to another may involve manipulating several different windows, including viewing applications and Web

browsers. The reading applications often do not remember which page was being read when the document was last opened, so the reader must spend time finding the page anew.

In the Instant Bookplex corpus viewer, moving from document to document is just a matter of zooming out and zooming in. Furthermore, with spatial cues, color cues, and semantically-zoomed text cues, the user can rapidly re-find documents that have been seen previously. Finally, the viewer leaves documents open to the last page that was viewed, so the reader can take up reading where he or she left off.

Which documents have been read. Anyone engaged in in-depth reading must keep track of which documents have been read (and to what extent) and which are yet to be read. This information may be kept in the reader's head, as annotations on a copy of each document, or as check marks on a personal bibliography, for example. However, none of these forms of the information are of much help to the reader in finding the most important unread documents. By using distinctive graphics (such as colors with a distinctive combination of hue and saturation), the Instant Bookplex corpus view makes it possible to see at a glance where to find the most important unread documents. Because this information is displayed as part of a layout that reveals other information about the documents as well, such as their categorization along one or more metadata axes, the reader can also identify important categories of highly ranked unread documents.

No need for a separate bibliography. Traditionally, readers performing in-depth reading construct a personal bibliography of the documents to be read. This bibliography is a document in its own right, containing both citation information about documents and other information that the reader thought to include, including notes about each document, a rating of its importance, information about how to find the document, and so forth. As more documents are found, keeping a personal bibliography up to date is a chore. Our work on Instant Bookplex suggests that personal bibliographies will become unnecessary. Citation information can be captured directly from the documents themselves. Notes, ratings, and information for finding the document can all be stored as metadata on a personal library object that represents each document. In essence a personal digital library can serve as a bibliography in addition to its other roles. The result is a savings of work for the reader and a bibliography that stays in synch with the contents of the document corpus, because the bibliography and the corpus are one and the same.

Not only does this synergy save work, but the function of both the bibliography and the digital library are enhanced. The digital library can display information that would ordinarily be in the bibliography (citations, ratings, information about what has and hasn't been read) to help the user navigate the corpus. Conversely, as a user reads through the documents in the digital library, the bibliography information is automatically updated to reflect the state of the user's reading and any ratings or notes that the user expresses while reading (e.g. by making annotations).

8. CONCLUSIONS

We have described the design for a document corpus browser that addresses some of the issues of supporting a reader during in-

depth reading tasks that involve viewing many documents over an extended period of time. By combining automatic citation-following with interaction, our system automates some of the tasks of building a topic-based document collection while leaving the user in control. By combining a ZUI, a document layout metaphor, and semantic text zooming, we have created a document corpus browser that allows documents to be seen in the context of related documents, allows rapid transition between documents, and takes advantage of human spatial memory to help the reader become familiar with a document corpus.

We have described how this design solves five problems associated with in-depth or syntopical reading that result from having to manage a large number of documents over an extended period of time. We have presented five design desiderata that express our desire to keep the user in control and to take advantage of human capabilities such as the ability to track motion and to form spatial memories. Finally, we have discussed the ways in which this system can save work during in-depth reading both by reducing the time needed for individual reading tasks and by eliminating the need for a separate bibliography.

9. FUTURE WORK

Our work so far has shown that practical tools can be built to help users assemble a document corpus based on a citation graph and track reading progress through the resulting corpus. However, we are planning improvements to all three of the components.

We plan to improve the algorithms used by the reference extraction tool for pulling out citations from technical papers. We will also augment the user interface of this tool, so that the user can more easily recover from common errors in data extraction. We plan to speed up the document finding tool by having it focus on the most promising URLs first. This tool will also be improved to give the user more information about the desired document, even if the document itself cannot be found. Finally, the corpus view will be extended to provide a more complete reading experience with support for annotation of documents, and highlighted full text search results across the entire corpus.

While our system is based on design criteria that make sense to us in the context of in-depth reading, we will test these techniques on people engaged in short and long term reading projects to find out how much value the techniques have in practice.

10. ACKNOWLEDGMENTS

We thank the Palo Alto Research Center for supporting this research, Bill Janssen for creating and improving the UPLib system, Ken Pier for implementing the citation-following Web form, and Paula Newman for creating and improving our citation parser. We thank the members of the User Interface Research team at PARC for helpful discussions of the bookplex concept that they pioneered.

11. REFERENCES

- [1] Michelle Q. Wang Baldonado and Terry Winograd. SenseMaker: An Information-Exploration Interface Supporting the Contextual Evolution of a User's Interests. *Proceedings of CHI '97*, 1997, pages 11-18.

- [2] Benjamin B. Bederson, Jesse Grosjean, and Jon Meyer. Toolkit Design for Interactive Structured Graphics. University of Maryland Computer Science Dept. Technical Report, CS-TR-4432, 2003. See <http://www.cs.umd.edu/Library/TRs/>
- [3] B.B. Bederson and A. Boltman. Does Animation Help Users Build Mental Maps of Spatial Information? *Proceedings of IEEE InfoVis '99*, 1999, pages 28-35.
- [4] S. T. Dumais, E. Cutrell, J. J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff I've Seen: A system for personal information retrieval and re-use. *Proceedings of SIGIR '03*, 2003, pages 72-79.
- [5] C.L. Giles, K. Bollacker, and S. Lawrence. CiteSeer: An Automatic Citation Indexing System. *Proceedings of the 3rd ACM Conference on Digital Libraries*, 1998, pages 89-98.
- [6] Giovanni Giuffrida, Eddie C. Shek, and Jihoon Yang. Knowledge-Based Metadata Extraction from PostScript Files. *Proceedings of the 5th ACM Conference on Digital Libraries*, 2000, pages 77-84.
- [7] Y. Han and L. Sterling. Agents for Citation Finding on the World Wide Web. *Proceedings of the Practical Application of Intelligent Agents and Multi-Agent Technology 1997*, 1997, pages 303-318.
- [8] W. C. Hill, J. D. Hollan, D. Wroblewski, and T. McCandless. Edit Wear and Read Wear. *Proceedings of ACM CHI '92*, 1992, pages 3-9.
- [9] William C. Janssen and Kris Popat. UpLib: a universal personal digital library system. *Proceedings of ACM Document Engineering '03*, 2003, pages 234-242.
- [10] S. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl. On the Recommending of Citations for Research Papers. *Proceedings of ACM Computer Supported Cooperative Work '02*, 2002, pages 116-125.
- [11] Lucy Terry Nowell, Robert K. France, Deborah Hix, Lenwood S. Heath, and Edward A. Fox. Visualizing Search Results: Some Alternatives to Query-Document Similarity. *Proceedings of SIGIR '96*, 1996, pages 67-75.
- [12] Kenton P. O'Hara, Alex Taylor, William Newman, and Abigail J. Sellen. Understanding the materiality of writing from multiple sources. *Int'l Journal of Human Computer Studies* 56(3), 2002, pages 269-305.
- [13] Thomas A. Phelps and Robert Wilensky. The Multivalent Browser: A Platform for New Ideas. *Proceedings of ACM Document Engineering '01*, 2001, pages 58-67.
- [14] Ken Perlin and David Fox. Pad: An Alternative Approach to the Computer Interface. *Proceedings of ACM SIGGRAPH '93*, 1993, pages 57-64.
- [15] Andreas Rauber and Alexander Müller-Kögler. Integrating Automatic Genre Analysis into Digital Libraries. In *Proceedings of the Joint Conference on Digital Libraries '01*, 2001, pages 1-10.
- [16] George G. Robertson, Mary Czerwinski, Kevin Larson, Daniel C. Robbins, David Thiel, and Maarten van Dantzich. Data Mountain: Using Spatial Memory for Document Management. *Proceedings of ACM UIST '98*, 1998, pages 153-162.
- [17] B. N. Schilit, G. Golovchinsky, and M. N. Price. Beyond paper: Supporting active reading with free form digital ink annotations. *Proceedings of ACM CHI '98 Conference*, 1998, pages 249-256.
- [18] Ben Shneiderman, David Feldman, Anne Rose, and Xavier Ferré Grau. Visualizing Digital Library Search Results with Categorical and Hierarchical Axes. *Proceedings of the 5th ACM Conference on Digital Libraries*, 2000, pages 57-66.
- [19] Charles Van Doren and Mortimer J. Adler. *How to Read a Book*. Touchstone Books; Revised edition (August 15, 1972).
- [20] Allison Woodruff, Rich Gossweiler, James Pitkow, Ed H. Chi, Stuart K. Card. Enhancing a Digital Book with a Reading Recommender. *Proceedings of ACM CHI '00*, 2000, pages 153-160.