BiblioText Version 5.0
A Hypertext Browser for Bibliographic Data and Notes

Michael L. Van De Vanter

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, California 94720

November 1990

Report No. UCB/CSD 90/603

Research sponsored in part by the Defense Advanced Research Projects Agency (DoD), monitored by Space and Naval Warfare Systems Command under Contracts N00039-84-C-0089 and N00039-88-C-0292, by IBM under IBM Research Contract No. 564516, by a gift from Apple Computer, Inc., and by the State of California MICRO Fellowship Program.

Abstract

BiblioText is a window- and mouse-based program for browsing bibliographic data and related notes. It is designed for a personal working environment characterized by indexed bibliographic data in *bib*, *refer*, or *tib* format, along with online documents containing imprecise citations that point into the database. BiblioText assists the writing of papers with functions like interactive keyword search and automatic generation of imprecise citations; it also supports related tasks like tracking notes and annotations in bibliographic data, building reading lists and annotated bibliographies, and managing personal libraries. In an appropriately configured data environment, BiblioText functions as a hypertext browser, traversing imprecise citations (in the manner of *bib*) and named links in the bibliographic database (a BiblioText extension).

This document supersedes an earlier version dated October 1988 and reflects changes to BiblioText since that time. Notable changes include a recent port to the X window system, using the XView toolkit, and a complete reorganization of the user interface.

Contents

List of Figures

1 Introduction

BiblioText is an interactive, window- and mouse-based program for browsing collections of notes, papers, and bibliographic reference data. It is a byproduct of ongoing research on the design of personal information management tools and on issues in user interface design. This research is part of the PIPER projects at UC Berkeley. BiblioText runs on Sun Workstations[1] under UNIX[2] and OpenWindows 2.0. Although BiblioText is integrated with OpenWindows[3], it also runs with other X11-based window managers, *twm* for example.

BiblioText, in cooperation with other UNIX tools, can help with the following tasks that arise in many working environments:

- Keeping track of personal notes on material already read;
- Building annotated bibliographies;
- Writing papers, including retrieval of relevant reading notes, location of important references, and automatic insertion of citation strings;
- Keeping track of reading lists, including the management of "virtual piles" of papers for later reading;
- Managing personal libraries; and
- Sharing bibliographic references and notes among colleagues.

With appropriately configured data, BiblioText can be used as a hypertext browser. It differs from other hypertext systems (see Conklin's survey [4]) in several important respects.

1. BiblioText is *compatible* with data used by other UNIX tools. It operates on ordinary text files and three kinds of bibliographic databases.
2. BiblioText is *specialized* for a particular application area (keeping notes, tracking bibliographic references, writing papers, and the like), and for particular suites of text processing tools (for example your favorite text editor and *bib*/*troff* or *tib*/TEX).
3. BiblioText is *read-only*. It complements other tools for managing your data; it does not replace them.

This document introduces BiblioText version 5.0 and supersedes an earlier version dated October 1988 [17]. Notable changes since the earlier report include conversion to the X

[1] Sun Workstation and OpenWindows are a registered trademarks of Sun Microsystems, Inc.

[2] UNIX is a registered trademark of AT&T Bell Laboratories in the USA and other countries.

[3] BiblioText's user interface works best and has the most functionality with the OpenWindows window manager. For example files can be loaded into BiblioText using the "drag and drop" operation on a file icon from the OpenWindows *filemgr*. BiblioText can be registered with the OpenWindows *binder* as the default viewing mechanism for files of bibliographic data. Finally, BiblioText is integrated with the OpenWindows help system.

window system, a complete reorganization of the user interface, functional enhancements, and the removal of some implementation restrictions.

This document describes how BiblioText can fit into your working environment and gives you enough information to get started with it. Section 2 "Why BiblioText?" begins with some background and a preview of what BiblioText can do for you. The remaining sections explain how to make BiblioText work (Sections 3 through 5) and what to do with it (Section 6), as well as some reflection on BiblioText's past and future (Section 7). For a quick introduction to BiblioText read Sections 2, 6, and 7.

BiblioText is only as useful as your collected data; Section 3 "How to Organize Your Data for BiblioText" introduces basic organizational techniques. The later discussion in Section 6 suggests how to use these techniques for maximum advantage.

You will need to configure the program to reflect your particular way of organizing data; Section 4 "How to Configure BiblioText for Your Data" contains the details.

Finally, Section 5 "How BiblioText Works" describes BiblioText's functional characteristics in some detail.

For completeness, this document mentions minor points about BiblioText and its use. These notes appear in a smaller typefont, as in the following. Feel free to skip these during a first reading.

BiblioText's *man* page summarizes much of the material presented in Sections 4 and 5.

2 Why BiblioText?

BiblioText was designed for a specific niche in personal working environments; it complements existing tools and existing data in support of certain kinds of work. This section supplies a bit of background on BiblioText's intended environment, and discusses the approach taken in its design.

Information-Based Working Environment

BiblioText was designed for an information-based working environment, where people read documents, write documents, and (whether they realize it or not) build complex webs of documents, notes, and bibliographic reference data. Some of these documents are in online computer files. Other documents are not online, but might be annotated, tracked, and cited using online files. In this environment, people spend their time on tasks such as the following:

- *Reading Documents*. Some are physical documents, others arrive electronically. Some documents must first be located physically, based perhaps on a reference in another document or on the recommendation of a colleague.

- *Taking Notes*. Note taking is often an important part of reading. Some notes become part of the document (as with marginal notes in a book), while some reside elsewhere

(in a notebook, for example). When document and notes are separate, it is sometimes necessary to locate one, given the other.

- *Building Piles of Documents.* There are often more documents to read than time permits. The standard solution (and an effective one in many situations [11]) is to stack them in *piles*. Piles, sorted perhaps by subject or project, are visual reminders and provide a convenient access mechanism for possible later reading.

- *Writing Papers.* Authors, as they work, may refer to documents they have read earlier, although it often isn't clear in advance which documents will be needed. If separate notes are available, those may be needed too. An important part of this task is determining which documents and notes are relevant at any given moment. Bibliographic descriptions of some documents may be copied into the paper's reference list and citations inserted appropriately.

- *Managing a Library.* People own personal copies of documents, although many remain unread. These include books, journals, conference proceedings, photocopies, course notes, and numerous loose ends. Finding specific documents can be difficult, especially when they might reside in various piles or may have been lent to colleagues.

- *Sharing Information.* Groups of colleagues usually circulate copies of documents among themselves, and they may also share online information in some way.

Stepping back from this description for a moment, the information structure in which this working environment is embedded resembles *hypertext*. Documents cite other documents, documents refer to notes, notes refer to documents, lists (the online equivalent of piles) refer to documents, and colleagues refer to documents in communication with one another.

Automated Tools

Computer based tools for personal information management are becoming more common. These usually include, among others, document processing tools such as text editors (like *vi* [8] and Emacs [15]), document typesetting programs (like *troff* [13] and TEX [9]), bibliographic database preprocessors for documents (like *refer* [10], *bib* [3], and *tib* [1]), and file systems that underly everything (like the one in UNIX).

With some ingenuity and initiative, these tools can be brought to bear on the tasks described above. They form a loose confederation, however, and getting them to work in concert can present unpleasant technical obstacles. The problem is that tools are generally built around their functionality, what they *do*, as opposed to *how* they might be used. This is the advantage of the toolkit approach to automated tools, but it puts the burden on the individual to create a convenient personal working environment that will earn its keep (that is, be worth the trouble of putting it together).

Even more troublesome is the observation that, for some of the tasks described above, the trouble of maintaining the information online is simply too great to justify use by one

person. Sharing, on the other hand, requires that personal working environments not be completely incompatible with one another.

Workstations

Workstation technology can help. It makes available more computing power, supports better user interfaces, and enables better integration among different tools.

Tool integration in practice, however, is too often limited to cut-and-paste. This mechanism permits text selected from one application to be inserted into another, but even this seldom works as generally as one would like.

Tools are being developed that take better advantage of workstation technology, for example hypertext systems like NoteCards [7] and Hypercard [6]. These advanced systems tend to be monolithic, however, and are often detached completely from existing working environments. This presents the potential user with a dilemma.

- Monolithic systems seldom work with existing data (documents, bibliographic databases, etc.).
- Monolithic systems may not support all the functionality to which users have become accustomed. This is especially true for users who add personal components to groups of batch tools.
- Monolithic systems tend to make data accessible only through the workstation interface, excluding access with standard terminals over dialup lines, for example.

Where BiblioText Fits

BiblioText is designed for information-based working environments. Potential users already have data and personal tools built around suites of batch programs for this kind of management (in particular, with data suitable for *troff*, *bib*, TEX, *tib*, and the like), and they want to take advantage of workstation technology for enhanced access to that data.

BiblioText is a browser, a tool that permits inspection of data, without requiring that its users understand the structure of the data at every turn. BiblioText is a workstation program, using bitmapped graphics and the mouse to support a quick and convenient user interface. BiblioText is compatible with existing tools, allowing navigation through data maintained with other programs. Although BiblioText is *general*, in the sense that it operates over data shared with many other tools, it is also *specialized* for the kinds of tasks described above. In particular, BiblioText is designed to bind together this working environment, and to make explicit and accessible those hypertext-like aspects of a personal working environment that are not visible through existing, individual tools.

3 How to Organize Your Data for BiblioText

BiblioText is a browser for data shared with other tools. This section describes the data domain over which BiblioText can operate and discusses basic organizational techniques for

```
%A Douglas C. Engelbart
%A William K. English
%T A Research Center for Augmenting Human Intellect
%J FJCC
%P 395-410
%D 1968
```

Figure 1: A raw ref. describing a conference paper

data that make BiblioText useful.

Like the tools it complements, BiblioText permits many degrees of freedom in your data organization. You will want to configure your environment to suit your personal needs. The discussion here emphasizes how to make the tools work. Section 6 "How to Use BiblioText" complements this discussion by suggesting how you might adapt these organizational techniques to your particular needs.

BiblioText operates over a data domain that contains two types of objects: bibliographic databases and online documents. This section discusses the the two data areas in turn, with a final note about sharing.

3.1 Bibliographic Databases

The backbone of BiblioText's data environment is a collection of bibliographic databases, which you manage using any of several suites of tools designed for just this purpose. BiblioText is compatible with database formats used by *refer* [10], *bib* [3], and *tib* [1]. For simplicity, this discussion presumes the use of *bib*, unless specifically mentioned otherwise.

Raw Refs.

In every database format the basic unit of data is the *bibliographic reference* (shortened in BiblioText terminology to *ref.*). A ref. is just a description, in bibliographic format, of something else. The something else being described could be a physical document (such as a book), part of physical document (such as an article from conference proceedings), another online document, or anything else that you want to manage, track, cite, or locate.

Using a text editor, you add refs. to your database in *raw* form, as groups of lines separated by blank lines. For example Figure 1 shows a raw ref. in *bib* format that describes a paper delivered at a conference. Each line of the raw ref. contains a *field*, identified by a label (sometimes called a *key*) like `%A` and `%T`. See the relevant documentation for a more detailed description of the various raw formats; all are similar. The conventional file name extension for raw refs. in *bib* format is `.ref`. A typical bibliographic database includes many files of raw refs., both to avoid unmanageably long files and to add useful structure to the database.

An important aspect of the raw ref. representation is that all fields need not be recognized by all tools. Each bibliographic format defines a standard group of fields, but other tools are free to use undefined fields with the confidence that existing tools will simply ignore data in such fields. BiblioText takes advantage of this freedom and allows you to specify that certain fields in your data be recognized as *links* between refs. and other refs. or between refs. and files (more on links later).

Abbreviations

Bibliographic databases in *bib* and *tib* format make use of macro expansion (or abbreviation), although the two systems define expansion a bit differently. This mechanism can buy a certain amount of compactness and uniformity. In the above example "`FJCC`" is just an abbreviation for "`Fall Joint Computer Conference`".

A typical bibliographic database may include several files of definitions for this kind of abbreviation. The exact format is beyond the scope of this introduction; consult the relevant database documentation for details. It is possible, and sometimes useful, to define your own additional abbreviations.

The document preprocessors for *bib* and *tib* support varying reference formatting styles that, among other things, include alternate sets of definitions for abbreviations. By convention, some sets of definitions are short (or terse), some are full, and some are used for both. An terse definition for "`FJCC`" in the above example might be "`Fall Joint Comp. Conf.`". BiblioText mirrors this usage with the Display Verbose mode for viewing formatted refs. When you create your own definitions, you may specify that they be used for verbose viewing only, for terse (non-verbose) viewing only, or for both.

Indexing

Most tools that read raw refs. (including document preprocessors and BiblioText) require the presence of *indexes*. Each is an inverted keyword index to one more more files of raw refs. [10]. It is usually necessary to rebuild an index when any of its files of raw refs. changes.

The programs for building indexes are *invert* (for *bib* databases), *indxbib* (for *refer* databases), and *tibdex* (for *tib* databases). Most allow you to select which fields to use for indexing; consult the appropriate documentation for details.

> You may want to exclude from indexing fields like `%V` and `%N` (periodical volume and number), `%P` (page number), and fields with extended annotations. It is an especially good idea to exclude from indexing any fields that you are using as links (see discussion of links below).

The conventional (and default) file name for an index is simply `INDEX`. A common organization for raw ref. files has them grouped into directories, with a single index for each directory. A particularly helpful technique for keeping an index current is to use *make* [5] with a makefile fragment something like the one appearing in Figure 2.

```
all:    INDEX

INDEX:  *.ref
        invert *.ref
```

Figure 2: A *make* fragment for maintaining an index

```
%A Douglas C. Engelbart
%A William K. English
%T A Research Center for Augmenting Human Intellect
%J FJCC
%P 395-410
%D 1968
%K hypertext mouse chord keyboard
```

Figure 3: A raw ref. with additional keywords

Additional Keywords

Bibliographic formats usually reserve the %K field for *additional keywords*, words added to a raw ref. only for indexing and keyword search. One common use for additional keywords is to associate a ref. with a subject, when the subject is not evident from the title. For example, Figure 3 shows how additional keywords might be added to the earlier example.

Annotation

An essential part of any personal information management is *annotation*. BiblioText supports two annotation mechanisms for bibliographic data: inline annotations (discussed here) and links (discussed below).

Most bibliographic database formats reserve the %X field for *inline annotations*, even though none of the standard tools support this field directly. Inline annotations are short bits of text, included with the raw ref. file, in which you can add anything you wish. For example Figure 4 shows how a short annotation might be added to the earlier example.

BiblioText explicitly supports the %X field. The contents of %X fields appear in BiblioText's viewer when Display Verbose is on, but are elided when it is off. Inline annotations are most convenient for short notes, containing one to twenty lines of text.

Links

BiblioText provides a powerful annotation mechanism as an extension to the basic bibliographic data format. You may place any number of *links* in each raw ref., based on field

```
%A Douglas C. Engelbart
%A William K. English
%T A Research Center for Augmenting Human Intellect
%J FJCC
%P 395-410
%D 1968
%K hypertext mouse chord keyboard
%X seminal paper, accompanied by astounding demo
```

Figure 4: A raw ref. with a short annotation

```
%A Douglas C. Engelbart
%A William K. English
%T A Research Center for Augmenting Human Intellect
%J FJCC
%P 395-410
%D 1968
%K hypertext mouse chord keyboard
%X seminal paper, accompanied by astounding demo
%Y <conceptual framework paper> engelbart conceptual framework 1963
%Z <notes> note/Engelbart68
```

Figure 5: A raw ref. with two links

labels you have defined for this purpose. A link points to another ref. or to an external file (outside of the bibliographic database proper). Links of the former type are called *cite links*, those of the latter are called *file links*. BiblioText supports rapid traversal of these links during browsing sessions.

For example, you might have specified in your environment that `%Y` fields contain cite links and that `%Z` fields contain file links. Figure 5 shows how you might further annotate the example ref. with a link of each kind.

A link field of either type must contain a name followed by a value that describes the target of the link. The name of the link may be a single word, a string surrounded by double quotes, or a string surrounded by angle brackets "<" and ">". The value of a link is the rest of the field. BiblioText displays the names of links in a format something like the following (Section 5.3 describes BiblioText's display in more detail):

<conceptual framework paper><notes>

A single ref. may contain any number of links of either kind, but they should be named uniquely to avoid confusion during browsing.

The value of a cite link is a collection of keywords that is presumed to identify another ref. uniquely (much like an imprecise citation in your *bib* documents). These links might point to alternate versions of a paper, a collection in which the ref. was reprinted, or whatever else you choose.

> It is an especially good idea to exclude from indexing any fields you are using as links. Otherwise, cite links would be inherently self referential.

The value of a file link is simply the name of a file. If the file name does not begin with a "/" BiblioText interprets the file name relative to the directory in which the raw ref. resides. These links might point to files in which you have written extended notes about the ref. (cf. "Inline Annotations" above), to an online copy of the abstract, or to the document described by the ref., if it happens to be online.

However you choose to use file links, the files to which they point are generally just instances of *documents*, the subject of the next section.

3.2 Documents

From the point of view of BiblioText a document is any text file that is not part of your bibliographic database proper. You may organize them in any way. In many cases they will be written for typesetting with *bib*/*troff* (or *tib*/TEX). Some might be papers you are writing or have written, others might be reviews of documents you have read, still others might be fragmentary working notes.

Citations

The one structural feature of documents relevant to BiblioText is the presence of *imprecise citations*, as defined by the document preprocessor *bib* (or *tib* or *refer* respectively). An imprecise citation is a specially delimited string of keywords that uniquely identifies a particular ref. from your database. For example, a document might cite the example ref. as follows:

> ...demonstrated first by Doug Engelbart during his memorable 1968 appearance at the Fall Joint Computer Conference [.engelbart english 1968.].

This citation allows your document preprocessor to create a suitable citation at this point in the text and to add the ref. to the list of references at the end of the document.

During a session with BiblioText, you can treat an imprecise citation as just another kind of *cite link*, pointing from a document to a ref. (cf. cite links that point from a ref. to another ref., discussed previously under "Links"). When you use your editor to examine the text file containing this document, you can select the text of the citation and then can use BiblioText (treating the selection as a set of keywords) to locate the ref. it identifies.

Hidden Citations

Although most citations in your documents will be of the ordinary sort, as demonstrated above, there are some useful variations. In particular, you can add citations that are *hidden*, either from the document typesetter, or from both the bibliographic preprocessor and typesetter.

Most document typesetters allow embedded *comments*, text in the source file that does not appear in the typeset result. One might insert an imprecise citation in such a comment; in *troff* this would appear as the line

```
.\"  [.engelbart english 1968.]
```

This citation *is* visible to the bibliographic preprocessor (*bib*), so that the cited ref. would appear in the paper's reference list. However, the citation itself would not appear in the typeset document. While browsing the source file containing a hidden citation, you can select the text of the hidden citation and use BiblioText to locate the cited ref.

It is also possible to hide a citation from both typesetter and bibliographic preprocessor, simply by avoiding the special delimiters that mark normal imprecise citations. One way to do this in TEX might be with the line

```
%  Review [engelbart english 1968] on this point.
```

Here no mention of the cited ref. would appear in the typeset document, but the comment would be visible to the reader of the source text, and the link would be suggested by the square brackets. As before, while browsing this file you can select the text of the citation and use BiblioText to locate the cited ref.

Section 6 "How to Use BiblioText" contains a more thorough discussion of how hidden citations might be useful.

3.3 Sharing

Although many BiblioText users will only have one bibliographic database (typically a directory of raw ref. files and a single index), there is much to be gained by sharing databases with colleagues.

Sharing Personal Databases

BiblioText can operate over multiple bibliographic databases; all that is required (other than the appropriate configuration) is read-access to the files. Database searches proceed sequentially through all the databases you have active at the time. This can be an effective way to share information among a group of people with overlapping interests. There is no risk to personal data, since BiblioText does not modify data in the bibliographic databases it reads.

You may prefer to keep some data private, for example certain files of notes pointed to by file links. You can limit access to some data using file permissions, without denying access to the bibliographic database in general.

A weakness of the *bib* data model becomes clear when databases are shared. A user who wishes to write notes about a reference mentioned in a colleague's database must make a personal copy of the ref. data so that the notes (inline or as a file link) can be added. There are now two copies of the ref. data for the same document, but BiblioText cannot discover their common ancestry.

Multiple Indexing

Some groups maintain among their communal resources shared bibliographic reference data, maintained in special directories with their own indexes. These might include, among others, tables of contents for journals and conference proceedings of common interest. You can effectively append shared databases to your own by adding these databases to your list for browsing (and document preprocessing).

You may wish to be more selective as you include communal data in your personal environment. For example, in your personal directory of reference data you can create file system links to selected files of shared reference data; the linked files then appear in your private database as well as in the shared one. BiblioText, when browsing, filters out this redundancy by refusing to add to the collection any reference that is already present, independently of how it was indexed.

Importing Reference Data

You can extend the scope of your bibliographic databases by importing reference data from external sources. One source would be various kinds of document distribution lists (departmental technical reports, for example), delivered online and converted automatically to *bib* format. Another possibility is the capture (and conversion) of bibliographic data resulting from queries to large online systems, such as library catalogues or commercial bibliographic services. As an example of the latter, the BibIX system (a variant of *bib* developed for a medical community [14]) includes tools for converting the results of online MEDLINE searches into *bib* format.

4 How to Configure BiblioText for Your Data

You will need to configure BiblioText to operate in your personal data environment. Although most configuration information is optional, you must specify at least one bibliographic database for keyword search in BiblioText to operate at all.

When you invoke BiblioText you may supply command line arguments[4] recognized by all XView clients, for example

```
bibliotext -Wp 200 100
```

[4] You may optionally supply the name of a file in addition to other command arguments; this has the effect of running BiblioText's Load File command on the file.

```
BiblioText.Db1.Label:          personal
BiblioText.Db1.FileName:       /users/mlvdv/bib/INDEX
BiblioText.Db1.Select:         True
BiblioText.Db4.Label:          proceedings
BiblioText.Db4.FileName:       /users/group/bib/proceedings/INDEX
BiblioText.Db4.Select:         True
BiblioText.Db12.Label:         eduardo
BiblioText.Db12.FileName:      /users/eduardo/bib/INDEX
BiblioText.Db12.Select:        False
BiblioText.CommonWords:        /usr/local/lib/bmac/common
BiblioText.AbbrevStyle:        Tib
BiblioText.Abbrev1.Label:      Short
BiblioText.Abbrev1.FileName:   /usr/local/lib/tib/bibinc.shortnames
BiblioText.Abbrev1.Mode:       Terse
BiblioText.Abbrev2.Label:      Full
BiblioText.Abbrev2.FileName:   /usr/local/lib/tib/bibinc.fullnames
BiblioText.Abbrev2.Mode:       Verbose
BiblioText.Link2.Key:          Y
BiblioText.Link2.Type:         Cite
BiblioText.Link4.Key:          Z
BiblioText.Link4.Type:         File
BiblioText.Display.Sources:    True
```

Figure 6: Example X resource definitions for BiblioText

These "generic tool arguments" specify initial window position the like; they are documented in the *man* page for XView. Other than generic tool arguments, all configuration information for BiblioText comes from resources in your X defaults file. Figure 6 contains an example collection of resources for BiblioText, where each resource is defined by a line of the form

`BiblioText`.*resource*:*definition*

The remainder of this section describes the X resources that BiblioText recognizes.

4.1 Databases

BiblioText needs descriptions of your bibliographic databases. These are groups of raw data files accompanied by indexes built with *invert* (or *indxbib* or *tibdex* respectively), as discussed in Section 3 "How to Organize Your Data for BiblioText".

BiblioText can browse as many as thirty-two databases. The X resource names for these are `Db1` through `Db32`. The numbers used to name the databases have no significance other than to order and distinguish among them. Any subset of `Db1` through `Db32` may be

described, but there must be at least one specified for BiblioText's Keyword Search command to operate at all. For example, the definitions in Figure 6 specify `Db1`, `Db4`, and `Db12`.

Four resources describe each database, only the first two of which are required:

`Label` This resource contains a string by which BiblioText will refer to the database when communicating with you, the user. Set it to a name that will allow you to identify the particular database at a glance.

`FileName` This resource contains the fully specified name of the file that contains the index for the database.

> It is safe to rebuild an active index while BiblioText is running. Just be sure to wait for *invert* to finish before starting any BiblioText operations that might attempt to read the index.

`KeyLength` The program *invert* truncates useful keywords to some maximum length when it builds an index, specified with a command line option. If you build an index with a maximum key length other than *invert*'s default, which is `6`, you should specify that same value in this resource. The default value is also `6`, so in the normal case you need not specify it.

`Select` Set this optional resource to `True` if you wish a database to be active when BiblioText starts (default is `False`). Once BiblioText is running, you may use the Select Databases command at any time to make individual databases active or inactive (more about this in the next section).

The program *invert*, when extracting potential keywords for indexing, discards certain words that occur frequently and carry little information. These words are enumerated in a file whose name is a command line option to *invert*, and whose default is `/usr/new/lib/bmac/common`. For keyword search to be accurate, BiblioText must use the same list of words; an optional resource allows you to specify this.

`CommonWords` Set this resource to the name of the file used by *invert* while building your indexes. The default value for this resource `/usr/new/lib/bmac/common`, the same as *invert*'s default, so in the normal case you need not specify it.

> BiblioText, when performing a Keyword Search (see Section 5), preprocesses its argument string by eliminating punctuation and any words in the file named by the resource `CommonWords`. It is possible (though not likely) to select a string of words for searching, only to have BiblioText complain that no "usable" keywords were supplied.

4.2 Abbreviations

Document preprocessors *bib* and *tib* expand abbreviated words with macro facilities; when doing so they use either short or long expansions for the abbreviated words, depending

on the user's choice of bibliographic style. BiblioText supports the same macro expansion facility during browsing, but allows a bit more flexibility in configuration.

The macro expansion mechanisms of the two preprocessors differ slightly, and BiblioText can operate in either manner. Specify which expansion technique to use with the following resource:

`AbbrevStyle` Set this resource to `Bib` (the default) if you use *bib*-style macros; this expansion style replaces all occurrences of the macro names, as long as they are delimited by spaces or punctuation marks. Set it to `Tib` if you use *tib*-style macros; this expansion style replaces only those names delimited by vertical bars. In both cases, macro expansion is recursive.

> Mixing definitions of the two styles during browsing results in nothing worse than incomplete or inelegant formatting of refs. in BiblioText's viewer.

You may describe as many as ten files containing macro definitions (also called abbreviations in this document). BiblioText reads these files when it starts and builds a table of definitions for use by the viewer. The X resource names for these are `Abbrev1` through `Abbrev10`. The numbers used to name the files have no significance other than to order and distinguish among them. Any subset (including none) of `Abbrev1` through `Abbrev10` may be described. For example, the definitions in Figure 6 specify `Abbrev1` and `Abbrev2`.

Three resources describe each file of definitions:

`Label` BiblioText's Browse Abbreviations commands allows you to browse the files of abbreviations currently in use. This resource contains a string by which BiblioText will refer to this file when the browser appears. Set it to a short name that will allow you to identify this particular file at a glance.

`FileName` This resource contains the fully specified name of the file containing the definitions.

> BiblioText ignores the "`I`" operator in macro definition files. This operator, which specifies inclusion of other definition files, is rendered unnecessary by BiblioText's configurability.

`Mode` BiblioText's table of definitions allows two potentially different definitions for each macro: a short one (for terse display when Display Verbose is off) and a long one (when Display Verbose is on). This resource tells BiblioText to treat each file in one of three ways: `Terse`, `Verbose`, or `Both`.

> BiblioText builds its dictionary by reading the macro files in order. Redundant definitions replace earlier ones. If a macro has been defined with both `Verbose` and `Terse` definitions, a subsequent `Both` mode definition will replace both earlier definitions. If a macro has been defined with a `Both` mode definition, a subsequent `Verbose` definition will leave the earlier `Terse` definition (and conversely for a `Terse` definition).

4.3 Links

BiblioText extends the basic bibliographic data format by allowing you to insert *links* into raw reference data. BiblioText announces the presence of any links when it formats a ref. for viewing, and makes available a convenient mechanism for traversing them.

You may reserve as many as ten field labels for use as links in your data. BiblioText sets aside the fields you specify and uses them for no other purpose. The X resource names for these are `Link1` through `Link10`. The numbers used to name the links have no significance other than to order and distinguish among them. Any subset (including none) of `Link1` through `Link10` may be described. For example, the definitions in Figure 6 specify `Link2` and `Link4`.

Two resources describe each link:

`Key` This resource contains a single character that labels the field you wish to set aside for linking.

> If you request a predefined field for a link, the role of the field as link supersedes its other uses. Thus it is best to avoid fields like `%A` and `%T` for your links.

`Type` This resource specifies which kind of traversal BiblioText should use for the link. Set this option to `Cite` when the value of the link field should be treated as an imprecise citation, referring to another ref. in the bibliographic database. Set this option to `File` when the value of the link field should be treated as a file name, referring to a document that is not part of the bibliographic database proper.

4.4 The Display

You may configure the way in which BiblioText displays the reference collection, using the following optional resources:

`Display.Verbose` Set this resource to `True` if you wish BiblioText's viewer to start with Display Verbose on; otherwise set it to `False` (the default). Once BiblioText is running, you may change the mode at any time. Section 5 explains the effects of this mode.

`Display.Sources` Set this resource to `True` if you wish BiblioText's viewer to start with Display Sources on; otherwise set it to `False` (the default). Once BiblioText is running, you may change the mode at any time. Section 5 explains the effects of this mode.

`Display.Columns` This resource specifies the initial width of the browser. The default is 60 columns and the minimum is 32 (so that all the control buttons will appear correctly). Once started, the browser may be resized at any time.

`Display.Rows` This resource specifies the initial height of the browser's viewer. The default is 30 rows.

`Display.Font` This resource specifies the font to be used in the browser's viewer; it does not affect the control area, menus, or any of the auxiliary windows. The default is medium times roman, if available on window server.

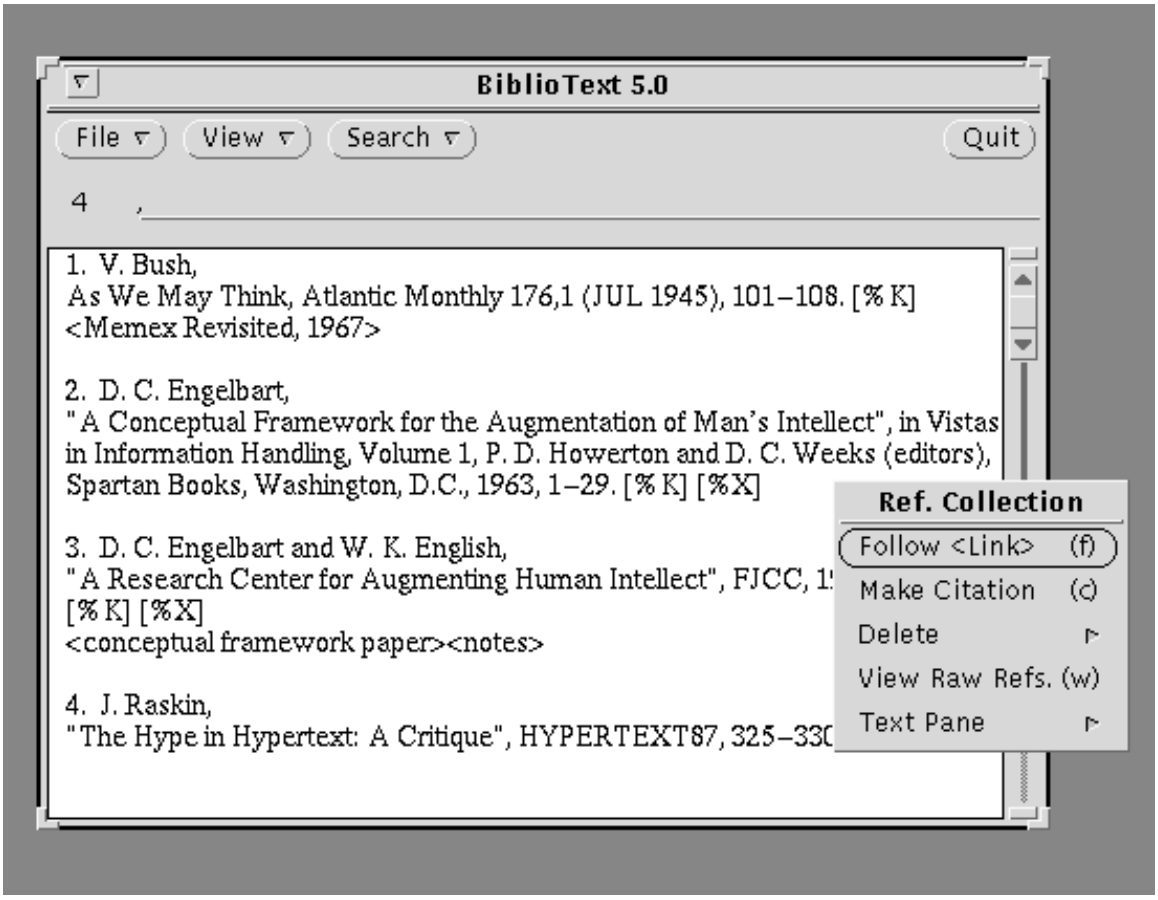

Figure 7: BiblioText with viewer menu visible

5 How BiblioText Works

This section explains the mechanisms and basic concepts behind BiblioText. This discussion emphasizes what BiblioText does; it complements the more tutorial presentation in Section 6 "How to Use BiblioText".

BiblioText runs on a bit-mapped workstation with mouse. Figure 7 shows how BiblioText appears, as a window consisting of a control area and a scrollable viewer. The *control area* contains active control objects and displays useful information. Figure 8 lists the commands available from each menu button in the control area. For convenience the first command available under each menu button is the default and may be invoked by a simple left click over the button. The *viewer* displays a formatted version the current collection of refs.; it permits operations on selected refs. with the *viewer menu* (visible in Figure 7). For convenience, a number of *keyboard accelerators* are available, summarized in Figure 9. These are single characters that can be typed when BiblioText's viewer has the input focus; they invoke the most frequently used of BiblioText's commands.

File Button	View Button	Search Button
Load File	Display Verbose	Keyword Search
Save Collection	Display Sources	Select Databases
	Sort	
	Browse Abbreviations	

Figure 8: Commands available from BiblioText's menu buttons

Character	BiblioText Command
b	Select Databases
c	Make Citation
d	Delete Selected Refs.
f	Follow <Link>
q	Quit
r	Display Sources (toggle)
s	Keyword Search
v	Display Verbose (toggle)
x	Delete Unselected Refs.
w	View Raw Refs.

Figure 9: Keyboard command accelerators (viewer area only)

This remainder of this section describes BiblioText's basic operating concepts and commands.

5.1 The Ref.

BiblioText's central function is to maintain a collection of refs. for viewing. This section begins by introducing the notion of a single ref.

During a session with BiblioText, a *ref.* is an internally stored copy of a bibliographic reference in raw form (see Section 3.1 "Bibliographic Databases"). It should be emphasized that the ref. is just a *copy* of a raw ref. from your bibliographic database; BiblioText cannot modify your bibliographic database.

As shown in earlier examples (see Figure 5, page 8), a ref. consists of lines of text. Each line contains a field, labeled by a character following "**%**" on each line. Along with each ref. in raw form, BiblioText keeps track internally (for possible display) of the following associated information:

Ref. Source: This is the name of the file from which the raw ref. was read (not the name of the index used to find it).

> Since BiblioText does not retain duplicate refs. in the current collection, and since there is only one source recorded per ref., it is in general not possible to discover multiple sources for a single ref.

Cite Source: If a ref. in the collection was located via imprecise citation in a document, this annotation contains (a) the name of the document file containing the citation, and (b) the citation string itself.

Links: BiblioText treats specially any fields you have reserved for links, keeping an internal map of link values for use during possible traversals. The example ref. in Figure 5 contains two links; the **%Y** field is a cite link, and the **%Z** field is a file link. See the discussion of the Follow <Link> command below for the semantics of the two kinds of links.

5.2 The Collection

BiblioText maintains a collection of refs. that changes dynamically in response to your commands. A discussion of these commands will follow, but for now three important aspects of the collection are worth mentioning:

- the collection is an ordered list of refs.;
- the collection contains no duplicates; and

  > Refs. are compared literally, character by character, for equality. This prevents the repeated addition of the same ref. by successive keyword searches, or by having the same ref. source file indexed multiply. This does *not* prevent the addition of the same ref. from two different raw ref. files, unless they happen to have been typed identically.

1. D. C. Engelbart and W. K. English,
A Research Center for Augmenting Human Intellect, Fall Joint Comp.
Conf. (1968), 395-410. [%K] [%X]
<conceptual framework paper><notes>

Figure 10: A formatted ref. as it appears in the viewer

- the collection is always numbered sequentially.

  There are some implementation restrictions in BiblioText, as described in the *man* page. One of these is a fixed limit on size of the collection (along with limits on the total character count in a single ref. and on the character count in a single field of a ref.).

5.3 The Viewer

BiblioText's *viewer* is a scrollable subwindow that constantly displays the current collection, formatted for rapid browsing. The viewer presents the refs. in order, separated by blank lines and numbered according to position in the current collection. The viewer formats each ref. compactly, according to simple rules that approximate standard reference formatting in *bib*. Thus the example ref. shown in raw form in Figure 5 might appear in the viewer as in Figure 10.

The viewer is an XView text subwindow. It is configured to be read-only, but other text subwindow functionality is available. You can scroll through the display using the standard XView scrollbar. You can use the Text Pane⇒ submenu of the viewer menu for textual searches in the collection. You can even split the viewer into multiple, independently scrollable, subwindows, although BiblioText isn't designed to manage split viewers particularly gracefully.

You may select the amount of information presented by the viewer by switching two *display modes* on and off, described in the following paragraphs. Switching either mode causes the viewer to redisplay the collection appropriately.

Display Verbose Mode

When Display Verbose is off, the viewer presents refs. in a terse format designed for rapid browsing (as shown in Figure 10). BiblioText uses several techniques for the terse display. Authors' first names are abbreviated, appearing as "D. C. Engelbart" for example, instead of "Douglas C. Engelbart." Macro (abbreviation) expansion uses short names instead of full names, such as "Proc." instead of "Proceedings." Finally, certain fields are elided, as explained below. When Display Verbose is on, the formatted ref. shown in Figure 10 might appear instead as shown in Figure 11.

1. Douglas C. Engelbart and William K. English,
A Research Center for Augmenting Human Intellect, Fall Joint Computer Conference (1968), 395-410.
[%K] hypertext mouse chord keyboard
[%X] seminal paper, accompanied by astounding demo
<conceptual framework paper><notes>

Figure 11: A formatted ref. with Display Verbose on

1. D. C. Engelbart and W. K. English,
A Research Center for Augmenting Human Intellect, Fall Joint Comp. Conf. (1968), 395-410. [%K] [%X]
<conceptual framework paper><notes>
Ref.: /users/mlvdv/bib/example.ref

Figure 12: A formatted ref. with Display Sources on

Display Sources Mode

When Display Sources is on, the viewer annotates each formatted ref. with file names representing the ref. source and the cite source (if any), as described earlier. The example ref. shown formatted in Figures 10 and 11 might then appear (with Display Verbose turned back off) as in Figure 12.

Field Formatting

The following list summarizes how the viewer presents various fields from each raw ref. Consult documentation for the various bibliographic databases for the standard field definitions

- *Standard fields:* `%A`, `%B`, `%C`, `%D`, `%E`, `%G`, `%I`, `%J`, `%N`, `%O`, `%P`, `%R`, `%T`, `%V`. The viewer formats various subsets of these fields according to standard preprocessor convention, depending on the type of document. A single ref. may contain multiple instances of `%A` and `%E` fields. If any fields in this list (other than `%A` or `%E`) appear more than once in a ref., the viewer presents only the last of each.

- *Link fields:* Fields defined to be links in your data will be presented in a special way, independent of the two viewing modes. The viewer presents the name of each link, surrounded by the bracketing characters "<" and ">"; the Follow <Link> command is available to traverse these links, as discussed below. Any number of links may be present; all will be displayed.

- *All other fields:* The viewer formats all other (not specially defined) fields in a uniform way. When Display Verbose is off, only the field label appears (in the form "[%K]") to announce the presence of the field. When Display Verbose is on, each label is followed by the contents of the field (as in "[%K] hypertext mouse chord keyboard"). A single ref. may contain multiple instances of any of these other fields; all will appear in the viewer.

The Viewer Menu

The remainder of this section describes various BiblioText commands for operating on the contents of the current collection. Some of these commands, in particular the ones that operate on selected references from the collection, may be invoked from BiblioText's *viewer menu*, visible in Figure 7. To invoke the viewer menu, press the right mouse button while the cursor is over the viewer. They are also available as keyboard accelerators (see Figure 9).

5.4 Selection

Many of BiblioText's commands require arguments that you may supply by prior mouse selection. One command requires string arguments, some commands require that you select one more more refs. from the current collection, and one requires that you select a specific link from the display.

String Arguments

The Keyword Search command requires a string argument. When invoked it always attempts to obtain the primary text selection from the window manager. When the primary text selection is unavailable (or unusable as a keyword string) a dialog box appears into which you may type the required string argument.

> If you are running under OpenWindows, this and other BiblioText dialog boxes are "pinnable." You may keep them on the screen continuously for faster interaction.

> You may select text for string arguments anywhere on your workstation screen, not just in the BiblioText window. This is useful for picking up keywords from your text editor window, for example.

Ref. Selection

BiblioText commands that require a selection of refs. work slightly less obviously. To select a single ref., use the mouse to select any text in BiblioText's viewer that corresponds to that ref. Only the characters you select will be highlighted, as with any primary text selection, but BiblioText commands will identify the selected ref. from as little as a single character selection. To select more than one ref., simply select a string of characters that begins somewhere in the first ref. you want and that ends somewhere in the last ref. you want.

When one of these commands is unable to interpret the primary selection as a selection of refs. (in particular when the primary text selection is in another window on your screen) BiblioText will complain and ask you to try again.

> It is possible to select the blank line between two refs. in the viewer. BiblioText treats this selection as belonging to the previous ref.

Link Selection

The Follow <Link> command, discussed in more detail below, requires that you select a specific link from the display in the viewer. BiblioText derives the selected link from the primary text selection in the viewer by considering only the first character selected. Thus, a single left click anywhere over the name of a link (including the bracketing characters) suffices to select a link.

5.5 Databases

BiblioText commands that involve database search (for example Keyword Search, Load File from a document, and Follow <Link>) operate over your currently *active databases*. To see which databases are active invoke the Select Databases command. An auxiliary window appears that lists the databases currently available to you, along with a check box for switching the status of each between active and inactive. Use the left mouse button to select or deselect the databases you want to use, and then make the auxiliary window disappear by clicking the Dismiss button. Section 4 explains how to make databases available for use.

5.6 Adding to the Collection

BiblioText normally begins each session with an empty collection. You may add refs. to the collection in several ways: by keyword search (in the manner of *lookup*), by loading files of raw refs. directly, by extracting citations from a document (in the manner of *bib*), or by traversing BiblioText's "cite links". In every case BiblioText will do the following with newly added refs.:

- discard any new ref. that is already in the collection, preventing duplicates;
- append remaining new refs. to the collection, in the order in which they were found; and
- scroll the viewer automatically so that the first of the newly added refs. appears at the top of the viewer.

Keyword Search

You may search your active databases by invoking the Keyword Search command. This search works much like the program *lookup*, and uses the keywords you specify as a string

argument. If there are no usable keywords selected (anywhere on the screen), you will be prompted for them in a dialog box. The search proceeds in order through every active database. *Every* ref. that matches the keywords will be added to the collection.

> BiblioText, like *lookup*, extracts matches from each database by looping in order over the keywords you supply (the semantics of *lookup* are to match refs. containing *all* of the specified keywords). Both programs have an implementation restriction that limits how many refs. the *first* keyword may match, per database index. Thus, you should avoid using words like "computer" as the first keyword (they won't cause problems if they aren't used as the first keyword).

Load Raw Refs.

You may add a file of refs. in raw format directly to the collection by invoking the Load File command. A dialog box appears in which you can specify a file and initiate the operation. Specifying a filename command line argument when invoking BiblioText has the effect of an initial invocation of Load File.

> When running with the OpenWindows window manager, you can invoke the Load File command by "dragging and dropping" onto BiblioText's viewer a file icon selected in the *filemgr*.

Load Document Citations

You may add refs. by extracting all the refs. cited in a document file; this amounts to previewing the reference list of the document. Invoke the Load File command, the same command used for reading refs. in raw form. BiblioText determines by inspection which kind of file you have specified and performs the appropriate kind of loading.

BiblioText's Load File command, when given a document file, mimics the operation (and semantics) of *bib*. For each citation that it encounters in the document, it searches the active databases in order, but only until it matches something. It is a *bib* error to match multiple refs. in a single database; likewise, it is a *bib* error for a citation to match nothing at all. When Load File discovers any such errors, it announces them in an special auxiliary window.

> The Load File command mimics the semantics of *bib*, while the Keyword Search command mimics the semantics of *lookup*. The subtle distinction between them, as suggested above, can create some confusion if you look too closely and if you aren't especially familiar with *bib*.

5.7 Traversing Links

When a ref. in BiblioText's current collection contains links, the names of the links appear in the viewer, delimited by "<" and ">", as shown Figures 10 through 12. To examine the object pointed to by a link, select the link and then invoking the Follow <Link> command. As described in Section 5.4, you can select a link with a left button press over any character in the name that appears in the viewer (including either bracketing character). BiblioText attempts to traverse the selected link, either to another ref. or to a file, depending on the type of the link you have selected.

Cite Links

If you have selected a cite link for traversal (like the link named "<conceptual framework paper>" in the examples), BiblioText attempts to locate the cited ref. in the same way that *bib* does. BiblioText searches the active databases in order, but only until it matches something. Although it is a *bib* error for a citation to match multiple refs. in a single database, BiblioText is more tolerant when traversing cite links and just prints a message to that effect. In any case, BiblioText adds the cited ref. (or refs.) to the collection, if it isn't already there, and then positions the viewer so that the cited ref. appears at the top of the viewer.

> If the value of a cite link field matches more than one ref., BiblioText positions the viewer at the first one found, whether or not it was already in the collection, and whether or not any of the others were.

File Links

When you select a file link for traversal (like the link named "<notes>" in the examples), BiblioText attempts to display the named file for viewing (and editing) in an auxiliary window. If the file name in the link field does not begin with "/" BiblioText interprets the name relative to the directory from which the raw ref. was originally read.

> The window in which cited documents appear is a standard XView text subwindow, which is also an instance of the XView editor. You may edit in these windows since this does not affect the validity of your database index.

5.8 Creating Citations

Once you have used BiblioText to locate a particular ref., you may want to cite it in a document by adding an *imprecise citation*, as defined by *bib*, that identifies the ref. You may request the creation of such a citation by invoking the Make Citation command. This command attempts to create a unique imprecise citation for the first of the currently selected refs.; it does so with heuristics and repeated search of the databases. When the heuristics fail, BiblioText will notify you.

When Make Citation succeeds the citation is guaranteed to identify uniquely the selected ref., but only relative to the set of databases currently active when the command is invoked. An evolving database can render previously unique citations erroneous; the Load File command, described earlier, is a convenient way to verify the integrity of citations in your document files.

When the Make Citation command succeeds, it writes the result, complete with surrounding "[." and ".]" delimiters, into a text field in the control area. From there you can either copy it into your document manually, or you use the window manager's cut-and-paste selection service to copy it automatically.

5.9 Deleting from the Collection

When you want to discard refs. from the current collection, select a command from the Delete⇒ submenu of the viewer menu. These commands affect only the contents of BiblioText's current collection and can have no effect on your bibliographic databases.

Delete Selected Refs.

The Delete Selected Refs. command requires that you have selected one or more of the refs. in the collection, as discussed in Section 5.4. It removes the selected refs. and renumbers the collection. For convenience, this command is the default selection of this submenu, so you may invoke Delete Selected Refs. by simply selecting Delete⇒ from the viewer menu; it is also available as a keyboard accelerator (Figure 9).

BiblioText treats the space between two refs. as part of the *preceding* ref. Thus, if you attempt to select all the text in a single ref. and you accidentally include the preceding blank line with your selection, you will have selected two refs. instead of one.

Delete Unselected Refs.

The Delete Unselected Refs. command, like Delete Selected Refs. requires that you have selected one or more of the refs. in the collection. It removes all except the selected refs. and renumbers the collection.

Delete Collection

The Delete Collection command empties the collection and requires no selection. As a precaution, BiblioText requires a confirmation before proceeding and allows cancellation.

5.10 Sorting the Collection

To sort the current collection, use the submenu Sort from the View menu. Choose from the menu of possible orderings (for example "Author-Date"), and BiblioText will sort the collection.

Sorting is a good way to find particular refs. when the collection is large. Another good way is to use the viewer's textual search facility, available via the standard XView Find command in the Text Pane⇒ submenu of the viewer menu.

5.11 Saving the Collection

You may save the current collection in your choice of format by invoking the Save Collection command. A dialog box appears in which you can select the format, specify a file, and initiate the operation. BiblioText offers four formats for saving the collection:

Formatted This format produces a text file that approximates what you see in the viewer. This is convenient for a quick hard copy, but has the disadvantage that the information may be incomplete and none of the information it contains is recoverable by BiblioText.

Raw This format produces a file of raw refs. This is useful for manipulating your bibliographic database in some cases. It may also be necessary for a hard copy when the formatted version is insufficient for some reason. The file may be loaded back into BiblioText, but since the original source is not recorded, refs. will then be treated as duplicates from different sources.

Expanded Raw This format produces a file of raw refs. with all the abbreviations replaced by their current definitions (either "Terse" or "Verbose" depending on the current display mode. This is helpful when exporting data to some other system, for example when you intend to convert the result to BibTEX format.

Citations This format produces a list of imprecise citations (using the same method as the Make Citation command), one per line in a text file. Unlike saving raw refs., which copies the database entry, this command stores what amounts to a pointer to the original database entry. This file may be loaded like any other document file containing citations, and the original referents will be retrieved.

Accurate reloading of a collection via citation list is the most flexible, but assumes the same set of active databases, and that the databases have not changed enough to render ambiguous any of the citations.

5.12 View Raw Refs.

You may view in raw form one or more of the refs. in the collection by invoking the View Raw Refs. command. This command displays the raw form of the selected refs. in an auxiliary window.

5.13 Abbreviations

BiblioText expands raw refs. using a macro definition (or abbreviation) facility. You may examine the abbreviations BiblioText is currently using by invoking the Browse Abbreviations command. An auxiliary frame will appear that consists of a text subwindow, in which one file may be viewed, and a small control area, from which you may select the definition file for viewing. This browser also displays the full file name and mode of the file being displayed. Section 4 explains how to make these files available to BiblioText.

6 How to Use BiblioText

Earlier sections described aspects of BiblioText's operation that you need to understand to make it work: data (Section 3), configuration (Section 4), and operation (Section 5). This section draws on all three aspects, but from a different perspective: how to use BiblioText to get your work done.

As with any tool, it is impossible to enumerate completely how BiblioText might be used, but this section should suggest some ways to get started. It is organized around several general (and overlapping) tasks for which you might find BiblioText useful. The presentation is ordered so that the tasks appearing first require the least amount of special preparation. As BiblioText becomes more and more part of your work, you may find that the overhead required for the later tasks in this list becomes worthwhile.

6.1 Writing A Paper

For this section, assume that you are going to write a paper in which bibliographic references play a role. The discussion presumes that you use *bib* as a bibliographic preprocessor to the *troff* document typesetter, but you could just as well be using *tib* and TEX.

Assume further that you have a simple bibliographic database. If you are new to *bib*, you might have created the database just for the paper at hand, adding the references you expect to cite. If you have used *bib* before, your database may contain at least all the documents cited in other papers you've written. Finally, assume that you have BiblioText running on your workstation, alongside a text editor in which you are writing the paper.

At this basic level, BiblioText can help with three tasks involved with writing a paper.

- Locate the bibliographic data for a document you have decided to cite.
- Create an imprecise citation for the ref. and insert it into your document.
- Preview your document's reference list, and ensure that imprecise citations are valid.

Locate a Ref.

You often don't know exactly which documents you will cite until you are writing the paper. BiblioText helps you to retrieve relevant bibliographic data when you need them, without having to leave your workstation to rummage through books, journals, and notes.

If you have a very small bibliographic database, created for just one paper, then you can conveniently browse through its entire contents. Use the Load File command to add the raw ref. file to the current collection, where it will be formatted for easy perusal. Scroll through the collection until you find the ref. you want; sorting the collection sometimes helps.

If you have a large bibliographic database, on the other hand, you can locate a particular ref. quickly with the Keyword Search command. Just enter one or two author names for keywords, or perhaps one important word from the title. To save time, select the authors' names directly from your editor's window, if the names happen to appear in the text you are writing. If the keywords match no refs., then try again with new keywords; you may have misspelled an author's name. Don't worry about keyword searches matching more than one ref.; it is usually convenient to scroll through the matching refs. until you find the one you have in mind.

Create and Insert a Citation

Once you have located the ref. you have in mind, you will want to insert an imprecise citation at the appropriate point in your document. This traditionally required guesswork, since you must combine keywords that identify exactly one ref. in your database. Using *bib* it is difficult to discover errors until you attempt to print your document.

With BiblioText you can use the Make Citation command, after first selecting the ref. you want to cite. BiblioText will attempt to create a citation, using heuristics, and display the entire citation string, complete with "`[.`" and "`.]`" delimiters, in the control area. Since the procedure is heuristic, it can fail when your database contains nearly identical refs. When the procedure succeeds, the resulting citation is guaranteed correct for the databases you have active in BiblioText at the time.

When the citation appears, you can use the window manager's cut-and-paste service to copy the string out of BiblioText's control area, complete with delimiters, into the text of your document.

Preview the Reference List and Check for Errors

If you are examining a document that already contains citations you can preview the document's reference list by extracting them and adding them to BiblioText's collection. Use the Load File command on the document file. If you are using the OpenWindows window manager, simply "drag and drop" onto BiblioText's viewer the icon corresponding to your document in the *filemgr*.

When BiblioText loads references via citations from a document file, it mimics the semantics of *bib* and attempts to find a unique match for each citation. It is a *bib* error for a citation to match other than exactly one ref. in a single database; BiblioText collects notes on any such errors and displays them in an auxiliary window when Load File finishes.

Running this command is a useful precaution when you change your bibliographic database; it is an inherent weakness of *bib*'s imprecise citation mechanisms that database changes can invalidate previously correct citations.

6.2 Keeping Notes

The previous section assumed only that you maintain a simple bibliographic database. If you also take notes on some of your readings, and if you choose to keep them online, this section suggests how you might organize your notes to take further advantage of BiblioText.

Nothing in this discussion insists that these notes (or, more accurately, online documents) be of your own creation. They can also be comments sent by electronic mail from colleagues. They can be reviews and recommendations downloaded from electronic bulletin boards. They might even be abstracts of the documents involved, when an online source is available.

At this level, where you maintain a bibliographic database with associated online documents, BiblioText can help with two additional tasks.

- Given a particular ref. in your bibliographic database, quickly locate various related online documents for viewing.
- Given an online document containing notes about a ref., quickly locate the ref. itself from the bibliographic database (and, by extension, any other notes related to the ref.).

Links from Refs. to Documents

When you create or receive an online document that concerns a document mentioned in your bibliographic database, you can record the relationship with a *file link*. In the simplest case, suppose that you are reading a paper that will be important for your work. Add a ref. describing the paper to your bibliographic database, if it isn't already there; include in that ref. a file link like the one appearing in the earlier example (Figure 5, page 8):

```
%Z <notes> note/Engelbart68
```

The word "`notes`" is the *name* of the link. The rest of the link field describes the file where you will write your notes.

During a browsing session with BiblioText, you can select the name of the link (which appears in the viewer as "<notes>" and use the Follow <Link> command to display the linked file in a separate window for further browsing. If, during your reading, you had the foresight to record important quotes or other items from the paper, you can use the window manager's cut-and-paste service to copy excerpts of your notes out of BiblioText, directly into your document.

The name of a link can be any string that will remind you of the relationship between the ref. and the document, for example "`Rebecca's critique`," "`short excerpt`," or "`abstract`." A single ref. can contain any number of file links, although it is a good idea to name them all differently to avoid confusion. Conversely, a single online document may be relevant to more than one ref. in your bibliographic database; record this many-to-one relationship by adding an identical file link to each ref.

Links from Documents to Refs.

When a file link points from a ref. to an online document, it can also be helpful to record the reverse relationship, pointing from the file back to a ref. A mechanism already exists for doing this, the imprecise citation. When you write an ordinary paper with citations, you are doing just this, adding links at various places in your document, pointing to refs. in your bibliographic database.

To follow this type of link during browsing, simply select the text of the citation (anywhere on the workstation screen) and use BiblioText's Keyword Search command to locate the ref. from your database. Don't worry about selecting the delimiters in the document, "`[.`" and "`.]`", BiblioText discards punctuation before producing keywords from your selection.

In a document file containing notes describing one or more refs., you might want to formalize the citations (links) a bit by putting them in a special place at the beginning. Write comments around the links for extra identification, especially if more than one appear in a single file of notes.

For printed copies of your notes, use the *bib* preprocessor to replace each citation (link) with a full description of the ref. For best results, try hiding the citation in a *troff* comment (as explained in Section 3.2). Then use the "`-f`" option to *bib*; this removes the formatted citation and inserts the reference data at the point of citation. As a final refinement, develop your own *bib* reference formatting style for this application; for example, you might want to format fields like `%W` or `%X`, which *bib* otherwise ignores.

6.3 Keeping Subject and Reading Lists

This section suggests how you might organize your data to help maintain subject lists of documents or lists of documents that you might want to read eventually. In an information-based working environment, there are typically many more documents like this than there is time for reading, but it can become important to locate something you noticed earlier.

The traditional manual technique is to accumulate documents in *piles*, organized by subject or project. This technique can be effective [11], but it doesn't scale well. Aside from the obvious environmental hazards, complications arise when the piles become too large or too numerous. For example, it becomes difficult to retrieve particular documents, when the need for them arises in a new context. Sometimes a single document might belong in more than one pile, and sometimes you want to add to a pile a document that you don't own, or that you have lent to a colleague.

With some additional investment in your data, you can use BiblioText to replace your physical piles of documents. Assume instead that you have some other way of organizing your documents so that, when you locate a ref. in your bibliographic database, you can physically locate the corresponding document without too much trouble (see the following section for some hints on how to use BiblioText for this, too).

Given the flexibility of the bibliographic database and BiblioText, you can choose from three different techniques for managing your subject and reading lists; the rest of this

section describes each in turn. Whichever technique you adopt, you will probably want to accomplish the following tasks at various times:

- Start a new subject list from scratch, perhaps drawing on information already stored in your bibliographic database.
- Add to a list a ref. that you have just noticed (whether or not you have read it yet).
- Make a note about why a particular ref. should be on a particular list.
- Browse through a list.
- Print a copy of a list, along with various notes and annotations.

Subject-based Raw Ref. Files

The simplest technique, and the one that scales least well, is to create files of raw ref. data that correspond to physical piles. Name each file according to the subject or project for which it was collected, such as `hypertext.ref`, and index the `.ref` files together.

To start a new subject list, or add to an existing one, use your editor to add raw refs. to a new file or to an existing one (and then, as always, rebuild your index). Use inline comments (in the `%X` field) to record the reason for a particular ref. appearing in a particular list.

To browse a subject list in BiblioText, use the Load File command to add the entire raw ref. file to the collection. There are several ways to print a copy. For example, use the preprocessor *listrefs* (or *roffbib*) for *troff*, or the preprocessor *tiblist* for TEX. Or, for a simpler version, load the file into BiblioText and write out the collection in the simple form that approximates the display in BiblioText's viewer.

The drawbacks to this approach are structural. As your database grows, fluid notions like "subject" and "project" do not partition your data neatly, nor are the distinctions likely to be stable. Some refs. may belong in more than one list, while others move frequently. Finally, you cannot conveniently include refs. from a colleague's database without making a personal copy.

Special Keywords for Indexing

A somewhat better approach separates the structure of the database from notions of "subject" and "project." Instead, create a unique keyword for each list you wish to maintain (for example "`htext1`") and add this keyword to each ref. you want to have in the list.

To start a new subject list, or add to an existing one, use your editor to add the line:

```
%K htext1
```

to each ref. you want on the new list (and then, as always, rebuild your database index).

To browse a particular subject or project list, use the Keyword Search command in BiblioText to find all refs. with the desired keyword. To print the list use the Save Collection

command. If you save a formatted version you will get a file you can print directly, approximating the display in the viewer. If you save a raw version you can typeset the resulting file by using the *listrefs* (or *roffbib*) preprocessor for *troff*, or the *tiblist* preprocessor for TEX. If you save citations you can typeset the resulting file of citations by using *bib* with the `-f` option as the preprocessor to *troff*.

One drawback to this approach is the danger of keyword collision. Since most database indexes are built with keywords truncated to 6 characters, careless selection can result in unanticipated refs. falling into a list. A second drawback is that you still must modify the database to add a ref. to a list. Third, if you want to add refs. from a colleague's database, you will still have to make your own copy. The final drawback is the lack of any natural place for comments. A ref. may be included in two lists by adding two keywords, but you can only annotate the ref. itself, not the reasons for membership in a particular list.

Reading Lists with Citations

The most general approach, and the one which scales best, is to maintain subject lists as separate document files, each containing a list of imprecise citations. Name each file according to the subject or project for which it was collected, such as `hypertext.doc`.

To create a new list, browse your bibliographic databases with BiblioText, using the commands Keyword Search, Sort, Delete Selected Refs., and Delete Unselected Refs. until the current collection corresponds to the list you want. Then write a file of imprecise citations with the Save Collection command.

To add to an existing list, use the Load File command to recreate the list in BiblioText's collection, and repeat the steps for making a new list. Alternately (and more conveniently), edit the list with your text editor, use BiblioText to locate the ref. you want to add, use BiblioText's Make Citation command to create a new citation, and finally use the window manager's cut-and-paste service to copy the citation string into the subject list file.

You can now add comments to each list, in the form of general text around the citation strings. This document amounts to the beginning of an annotated bibliography, and you can prepare it for typesetting as much as seems appropriate.

To browse the contents of a list, use BiblioText's Load File command to extract the cited refs. You have several choices for producing printed copies of these lists, depending on how you format the files. One convenient approach is to hide each citation string in a *troff* (or TEX) comment, as described in Section 3.2, and then use *bib* (or *tib*) with the `-f` option as a preprocessor to *troff*(or TEX). This causes each citation to be replaced in the formatted output with the relevant reference information extracted from your bibliographic database. As mentioned in Section 6.2, you might develop a special formatting style for this application, printing out the contents of fields like `%W` and `%X` that *bib* normally ignores.

This third organization has drawbacks too, but fewer than the first two. And these drawbacks are the ones inherited from the bibliographic database model. For example, imprecise citations in a subject list can (like a citation in any document) become invalid as you add new refs. to your database.

6.4 Library Management

Closely related to management of your bibliographic reference data is the need for physical management of the documents you own. If you organize your database appropriately, BiblioText can help manage your personal library. In particular, BiblioText can help with three tasks.

- Given a particular ref. in your bibliographic database, determine who owns a copy and exactly where can it be found.
- Print a "shelf list" containing all the documents you own of a particular type (e.g. technical books).
- Keep a circulation list of documents you've loaned to colleagues.

There are two general approaches you might take to organizing your data for library management, a single-level scheme using the `%W` field, and a two-level scheme using location-based raw ref. files in addition to the `%W` field.

The %W Field

Most bibliographic database formats reserve the `%W` field for *location information*, even though none of the standard tools support this field directly. BiblioText makes this field visible, and you can put it to use for library management.

The simplest approach is to reserve a specific word (say "`mlvdv`") and add the line

```
%W mlvdv
```

to the refs. for every document you own. When you locate a ref. during a session with BiblioText, and you want to discover its physical location, turn on Display Verbose to see the value of the `%W` field. To extract from your database a list of the documents you own, use the Keyword Search command with the keyword `mlvdv` (assuming, of course, that you have included the `%W` field in your indexing).

When you add a ref. to your database for a document you don't own, you can record other information in the `%W` field to help you remember where it can be found. For example, you might use a library call number or the name of a colleague from whom you borrowed the document.

If your library is so large that locating particular documents can be a problem, you can refine further the keywords you add to the `%W` fields, perhaps reflecting different physical collections within your library. For example, you might use keywords like "`mlv-file`," "`mlv-cs-book`," and "`mlv-tr`" to correspond to a filing cabinet, technical bookshelf, and technical report shelf respectively.

If you use BiblioText to browse databases belonging to several colleagues, and not all of them use `%W` fields, you can still determine the origin of a particular ref. by turning on Display Sources. In this mode, BiblioText annotates each displayed ref. with the name of the raw data file from which the raw ref. was extracted. This is usually enough information to determine the owner of the database.

```
%A Jef Raskin
%T The Hype in Hypertext:  A Critique
%J HYPERTEXT87
%P 325-330
```

Figure 13: Compact representation for a ref. in a collection

One drawback to the single-level approach is the extra storage required for a `%W` field in every ref. Another drawback was alluded to earlier, the likelihood of keyword collision when keywords are truncated to 6 characters for indexing. It is difficult to maintain a collection of keywords that are both meaningful and unique in the first 6 characters, when mixed in with all the other keywords in your database. Of course, you can index with more than 6 characters, but this increases the size of your database indexes.

Location-based Raw Ref. Files

A more general approach to library management, and one that scales better, is to adopt a two-level organization. First, structure the raw ref. files in your database to reflect the physical arrangement of your library. Within that organization, use `%W` fields when appropriate for additional information.

Start by creating a file of raw refs. corresponding to each group of documents you own, and name each file suggestively. For example, books on your technical shelf might be in the file `tech.book.ref`, papers in your filing cabinet might be in the file `file.ref`, and technical reports on a separate shelf might be in the file `tr.ref`.

Create also a separate file for each major collection. A collection might be the proceedings of a conference, a collection of reprints, a class reader, or any other physical group that contains multiple bibliographic entities. Develop a naming convention for these, too; for example, articles in a recent conference reside in the file `hypertext.87.ref`. Judicious use of the abbreviation facility will help you keep these files compact. For example, a ref. in a collection might appear in the file as shown in Figure 13. Here the abbreviation "`HYPERTEXT87`" expands into several fields that describe the conference.

Finally, maintain all of your raw ref. files in one directory and index them together. Use a Makefile to automate some of the management, as suggested in earlier sections.

During browsing, you can discover the location of a ref. by turning on Display Sources, since the name of the file now encodes this information. Furthermore, each raw ref. file is itself a "shelf list" for part of your library. To print a shelf list, use any of the techniques described earlier for printing a nicely formatted version of a raw ref. file.

Within each group (or file), adopt an appropriate strategy for recording additional location information in the `%W` field as needed. For example, in a file containing conference proceedings, the `%P` field containing page numbers will suffice; no additional information is necessary. In a file containing technical books, where the corresponding shelf is sorted by author, the last name of the first author will suffice; again, no additional information

is necessary. On the other hand, in a file containing a bound collection of reprints, you may want to add %W fields with page numbers, since the %P fields will contain page numbers only from the original source of each article. Use the %W field also for exceptions to your standard organizing scheme, for example, when a reprint is filed under the second author's name. Another use for %W is to record a library call number.

Using this organization, you won't need the %W field often, but when it is present, BiblioText will notify you with "[%W]" in the viewer. Turn on Display Verbose when you want to see the additional %W field information.

As a final organizational refinement, consider keeping your ref. files sorted in an order that reflects their physical arrangement. For example, a shelf of technical books might be sorted by author, but a shelf of technical reports might be sorted first by institution and then by report number. Special *make* rules using *sortbib* can keep files sorted as you add new refs.

An important part of library management is to track circulation. Rather than annotate the database directly with circulation data, at the cost of extra index rebuilding, use BiblioText to maintain a circulation list. This is just a variant on the annotated bibliography (or annotated subject list) described earlier. Keep a file of imprecise citations, along with notes about when and to whom they were loaned.

6.5 General Browsing

The structure of your data is as rich as you choose to make it, and if you use your database for the other tasks described in this section, it will be quite rich. BiblioText allows you to browse this information structure as an extended hypertext document, with all the advantages that have been claimed for the hypertext model. This section discusses a few aspects of this relatively open-ended task.

Explicit Relationships

Once you begin to view your data as hypertext, you will begin to add links that aren't strictly required for the more specific tasks discussed earlier. These links can capture important relationships that you have uncovered while reading and thinking; these relationships can themselves carry information.

Most kinds of explicit links have been discussed in earlier sections. For example, file links can record the relationship between a ref. and your comments, a colleague's comments, the abstract, or other supporting material. If it is important to identify the subject lists (see above) to which a ref. belongs, file links can record these relationships too. BiblioText's Follow <Link> command makes it convenient to traverse from a ref. to a linked file.

Imprecise citations are links in the reverse direction, between a document file and a ref. Uses for citations include ordinary document citations for typesetting, and special lists of documents of various sorts: subject lists, circulation lists, bibliographies, and the like. BiblioText's Keyword Search command makes it convenient to traverse from a citation to a cited ref.

You can use cite links to record whatever relationships among various refs. that you find important. For example, you might want to record that a technical report you own was condensed and published as a journal article. More generally, you might want to leave a trail through a particular body of work, when later work amplifies on the earlier. Since you can name links arbitrarily, link names can themselves contain useful information. BiblioText's Follow <Link> command makes it convenient to traverse from a ref. to a linked ref.

Implicit Relationships

During browsing, you can also take advantage of the general indexing mechanism to discover relationships that you haven't explicitly recorded. For example, you might discover an interesting ref. and wonder what else a particular author has written. Select the author's name in BiblioText's viewer and use the Keyword Search command to collect more refs. from the database.

The same technique works for project names, system names, keywords, and anything else that you can discover during browsing. The indexing mechanism is necessarily approximate, so you will often discover irrelevant refs. among those you locate. This kind of unstructured searching doesn't scale up for large libraries, but it works well for typical personal and shared group libraries.

Extended Browsing Domains

If possible, locate bibliographic databases that belong to colleagues with similar interests, and configure your BiblioText environment to read those databases too. Then, when browsing, you might discover a title that looks interesting, but which you don't own. Select the last name of an author (or the name of a project or system) and use the Keyword Search command. Among the results, you might find that a colleague owns something closely related by the same author (or about the same project or system).

Extend the range of your browsing even further by importing bibliographic data from outside sources and converting it to *bib* format for browsing. The necessary conversion will depend on the source, but often can be automated. You can then browse quickly through new arrivals, simply by loading the file of converted raw refs. If you keep archives of imported data, distribution lists for example, you might keep separate databases for all of these for later browsing.

7 Themes and Issues

BiblioText is a byproduct of ongoing research in the design of software tools for personal working environments. This section briefly mentions some of the issues addressed during this research; some are discussed more fully in an earlier report [16].

Task Oriented Design and Coevolution

The creation and subsequent evolution of BiblioText exemplifies several important ideas in the study of how programs get designed. The first prototype of BiblioText was, and continues to be, an example of *task-oriented* design. It was built to get a particular job done, in a particular existing niche, and not to create a new computational metaphor.

During BiblioText's evolution it has become clear that the notion of *niche* must be broadly defined. Both original design and subsequent evolution have responded to environmental factors that include, among others, the computational environment (hardware, operating system, window manager), preexisting sets of tools and data (like *bib* and the established databases in *bib* format), and the skills and expectations of potential users.

Furthermore, these environmental influences are all in constant change. Hardware speed has increased, three different releases of the original SunView window manager arrived, a new database format was designed, and the user community has become more accustomed to certain window/mouse interactive styles. Most recently BiblioText was ported to the X11 system, reflecting community trends. Even the first experimental prototype of BiblioText was a response to this kind of change; it was conceived as an interactive, window/mouse based version of the batch program *lookup*.

Finally, the history of BiblioText shows that its presence in local working environments has influenced how people do their work, leading in turn to more ideas for enhancements to BiblioText. It is through this kind of coevolution, by being useful to some community at every step, that small useful systems often grow into larger useful systems.

Principled User-Interface Design

At one point during its history BiblioText was the subject for an experiment in the design of user interfaces. Part of that work involved the construction of several explanatory models; Sections 5 and 6 of this document inherited some of these. Another part of that work was a principled redesign, following cognitive strategies like those proposed as part of "User Centered System Design" [12]. After an empirical evaluation by some new users, the results of the whole project were used to prepare yet another redesign of the user interface.

BiblioText's user interface has continued to evolve since the experimental redesign, but it has always been a vehicle for the exploration of user-interface themes, as well as a useful tool.

Limitations and Future Prospects

Although BiblioText could undoubtedly continue to evolve and improve as it has in the past, two particularly troublesome sets of limitations stand in the way.

The first problem is the bibliographic database. The data model upon which it is based was never intended to support applications of the size and complexity of those discussed here. For example, important non-bibliographic entities (such as journals, authors, and the like) are not treated as first class objects in the database. Furthermore, the supporting

mechanisms do not scale well as databases grow. It is computationally expensive and inconvenient to rebuild an index after any change to the raw data.

A second set of problems concerns BiblioText's document viewing mechanism. BiblioText presents document files in a simplified XView editor; this editor is not widely used in this environment, and is not as configurable and extensible as one would like. The result is that many users of BiblioText have to deal with two different editors.

A solution to both problems would be to integrate BiblioText with a more general, window/mouse editing system. A promising candidate is *Pan*, also being developed as part of the PIPER projects at UC Berkeley. *Pan* is designed for language-based and general structure/text editing. *Pan* supports more windows, and is fully configurable and extensible [2].

8 Acknowledgements

Robert Ballance supplied helpful suggestions for the original prototype design. Eduardo Pelegrí-Llopart and Dain Samples helped evaluate the user interface of an intermediate version. Dain Samples and Bruce Forstall contributed several rounds of constructive criticism. Dain Samples provided local support and enhancements for both *bib* and *tib*. Phil Garrison and Robert R. Henry provided earlier local support for *bib*.

Profs. Andy diSessa, Susan Graham, Mike Harrison, Peter Pirolli, and Larry Rowe all helped create the academic setting in which this research could take place.

9 References

1. Alexander, J. C. *Tib: A TeX Bibliographic Preprocessor*. Department of Mathematics, University of Maryland, 1986.
2. Ballance, R. A., Graham, S. L. and Van De Vanter, M. L. The Pan Language-Based Editing System For Integrated Development Environments . *Proceedings ACM SIGSOFT '90: Fourth Symposium on Software Development Environments* (December 1990).
3. Budd, T. A. and Levin, G. M. A UNIX Bibliographic Database Facility. University of Arizona Technical Report 82-1, 1982.
4. Conklin, J. Hypertext: An Introduction and Survey. *Computer*, *20*, 9 (September 1987), 17–41.
5. Feldman, S. I. Make – A Program for Maintaining Computer Programs. *Software–Practice & Experience*, *9*, 3 (March 1979), 255–265.
6. Goodman, D. *The Complete HyperCard Handbook*. Bantam Books, New York, 1987.
7. Halasz, F. G., Moran, T. P. and Trigg, R. H. NoteCards in a Nutshell. *Proceedings SIGCHI Conference on Human Factors in Computing Systems*, Toronto, Canada (April 1987).

8. Joy, W. *An Introduction to Display Editing with Vi.* Computer Science Division, EECS, University of California, Berkeley, March 1979.

9. Knuth, D. *The TeXbook*. Addison Wesley, Reading, Massachusetts, 1984.

10. Lesk, M. E. *Some Applications of Inverted Indexes on the UNIX System*. Bell Laboratories, Murray Hill, New Jersey, 1980.

11. Malone, T. W. How Do People Organize Their Desks? Implications for the Design of Office Information Systems. *Proceedings ACM-SIGOA Conference on Office Information Systems*, Philadelphia, Pennsylvania (June 1982), (Extended Abstract).

12. Norman, D. A. and Draper, S. W., Eds. *User Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986.

13. Ossanna, J. F. *NROFF/TROFF User's Manual*. Bell Laboratories, Murray Hill, New Jersey, 1976.

14. Rodgers, R. P. C., Gardels, K. and Finkelshtain, A. BibIX – A Bibliographic Data Base & Text Formatting System for UNIX. UCSF Laboratory Medicine, CALM/MedIX Technical Report 86-1.2, July 1987, Release 1.2.

15. Stallman, R. M. EMACS: The Extensible, Customizable, Self-Documenting Display Editor. *Proceedings of the ACM-SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices*, *16*, 6 (June 8-10 1981), 147–156, Reprinted in BarShrSan, Chapter 15.

16. Van De Vanter, M. L. Designing BiblioText: An Experiment in User Interface Design. Computer Science Division, EECS, University of California, Berkeley, 88/454, October 24, 1988.

17. Van De Vanter, M. L. BiblioText: A Hypertext Browser for Bibliographic Data and Notes. Computer Science Division, EECS, University of California, Berkeley, 88/455, October 25, 1988.