On Integrated Bibliography Processing*

Michael A. Harrison Ethan V. Munson

April 5, 1993

Abstract

Bibliography processing systems are important to the production of scholarly and technical documents. While the existing systems are a significant aid to authors, their designs are not sufficient to handle the demands that have arisen with their continued use. These demands include larger bibliographic databases, sharing of databases among multiple authors, integration with document editors, and the desire for new features.

This paper examines these issues as they are reflected in three enhancements to the bibliography processing facilities of the GNU Emacs BibTeX-Mode and TeX-Mode integrated editing environment. The added features were a reference annotation facility, forms-based queries for automatic citation, and an enhanced reference inspection facility supporting WYSIWIG display of references. The design and implementation of the three features are discussed in detail. Their relationship to other bibliography processing tools is discussed.

1 Introduction

Document preparation systems are now in widespread use for many purposes on a variety of computers. Our concern is in systems which can be used for the preparation of scholarly and technical documents. The issues in the design and implementation of these systems have been discussed extensively in the research literature. Discussion of these issues and further references can be found in the paper by Chen and Harrison [6] and in Chen's thesis [4].

An essential part of the scholarly process is to provide references to the work of others through a bibliography. Efficient and convenient methods for searching bibliographic databases and then constructing bibliographies can lead to a significant improvement in the scholarly process. In this paper, we present some enhancements to an integrated bibliography processing system.

*Sponsored by the Defense Advanced Research Projects Agency (DoD), monitored by Space and Naval Warfare Systems Command under Contract N00039-88-C-0292.

The first bibliography processing system, `refer` [11], is now slightly more than a decade old. The user of `refer` specifies the logical components of his references with a simple language and places symbolic citations to these references in his `nroff` or `troff` document. `Refer` replaces the symbolic citations with the actual citations that appear in the formatted document and inserts formatting commands for a sorted bibliography at a user-specified point in the document. `Refer` is limited by an implementation that sacrifices speed for storage efficiency and by a lack of formatting options.

In the period since the introduction of `refer`, several other bibliography processing systems have emerged and their use has become widespread, at least within the computer science research community. Scribe [18] is a document processing system which includes a bibliographic component. Both its language for defining references and its method for specifying citations are quite different from `refer`. It also provides much greater control over bibliography and citation formatting. BibTeX [14] is designed to work with the LaTeX document processing system [10]. Its design is heavily influenced by Scribe. It is described in more detail in Section 2. `Bib` gives users of `nroff` and `troff` a more efficient implementation and greater control over bibliography and citation formatting than provided by `refer`, with which it is largely compatible. `Tib` [1] allows the use of `refer` and `bib` database files with the TeX family of document formatters [9, 10, 21]. GNU Emacs TeX-Mode [2] and BibTeX-Mode [3] provide an integrated environment for editing TeX and LaTeX documents and BibTeX databases.

With most software, the passage of time gives rise to demands which were not originally anticipated. For bibliography processing systems, the most common pressures are:

Database Size The bibliographic databases of long-term users are becoming fairly large, containing a few thousand entries. At this size, the lack of database management tools begins to tell. Another common problem is that the symbolic citation schemes used by these systems no longer work well. The problem is becoming more severe as these systems have spread outside the computer science community to fields which have a higher volume of publication and which make widespread use of on-line reference services.[1]

Sharing Entries in a bibliographic database generally do not change. So, it makes good sense to amortize data entry effort over several people by sharing the database. Also, a shared database is more likely to be useful as a general reference tool, since it can be used by an individual to locate sources he has not yet read. Existing bibliography processing systems provide little support for sharing.

Integration Bibliography processing tools are not used in isolation. Rather they are just

[1]We know of a `bib` database used by medical researchers which contains more than 50,000 entries [19].

one component of a set of editing, formatting, and information management tools. Users naturally want these tools to be integrated, which they generally are not.

New Features A bibliographic database can be used for more than just producing a document with citations and a bibliography. The sources identified by bibliographic references are a central component of the scholarly enterprise and, as such, can be put to many uses. Thus, a bibliographic database can be used to produce reading lists and annotated bibliographies or to maintain notes on the documents it refers to. Bibliography processing systems provide only limited support for these activities.

These pressures, combined with the inherent importance of bibliographic data, have made bibliography processing an area of continued research. Several interesting systems have resulted from this research. BiblioText [26] is a browser for `bib` databases. BibTool [7] is a bibliographic front-end to a relational database system compatible with `bib`. BibIX [20] is a collection of updates and additions to `bib` that corrects bugs and makes `bib` more suitable for use in a biomedical research environment.

This paper describes three recent extensions to the bibliographic features of GNU Emacs TeX-Mode and BibTeX-Mode. The three extensions are:

- Modification of BibTeX-Mode to support the attachment of annotations to an entry in the reference database.
- An improved query method for use with the automatic citation mechanism of TeX-Mode, which is better suited to large databases.
- A WYSIWIG display mechanism for the reference inspection feature of TeX-Mode.

This section is an introduction to the issues discussed in this paper. Section 2 gives some background on our document processing environment. The motivation and design goals of the newly added features are presented in Section 3, while Section 4 details their implementation and Section 5 discusses their relationship to other work on bibliography processing. The final section presents some conclusions and suggestions for further research.

2 Background

While the concepts investigated for this research apply to any bibliography processing environment, the tools which were developed to examine the concepts do not. These tools help users who prepare documents which will be formatted using the LaTeX document preparation system [10] and its companion bibliography processing system, BibTeX [14].

[2] They are implemented as part of GNU Emacs TeX-Mode and BibTeX-Mode, which run as part of the GNU Emacs programmable editor [22]. The remainder of this section gives brief descriptions of each of these five systems, with special emphasis on their bibliographic features.

2.1 LaTeX and BibTeX

LaTeX is one of the family of TeX document formatters. Like Scribe [18], on which it is based, LaTeX's formatting language is declarative, rather than procedural. BibTeX is its accompanying bibliography processing program and closely resembles the bibliographic component of Scribe.

The user provides LaTeX with a text file containing formatting commands and the text of his document. All formatting commands begin with the backslash ('\') character. If LaTeX is run on a file called `foo.tex`, it will produce a log file (`foo.log`), a device-independent output file (`foo.dvi`), and an auxiliary file (`foo.aux`). When the user wants to cite a source, he places a symbolic citation command in his file. For example, the LaTeX command:

```
\cite{kay:fairshare}
```

cites a reference whose key is `kay:fairshare`. LaTeX records each symbolic citation in the auxiliary file along with the bibliography style requested and the set of bibliography database files specified by the user.

BibTeX reads the auxiliary file and searches the database files specified there for entries whose keys match the symbolic citations. BibTeX then sorts the references according to some criteria and computes the actual citations that will appear in the final document. The formatted references and the actual citations are recorded in a bibliography file (which, for a main document named `foo.tex`, will be called `foo.bbl`).

A BibTeX database is a collection of text files containing entries specified in a simple language. This language is based on the one used by Scribe and BibTeX will run correctly with a Scribe reference database. A sample entry in shown in Figure 1. Each entry is composed of a type name, a left brace or parenthesis, a key, a list of fields separated by commas, and a right brace or parenthesis. The type name explicitly specifies the reference type (e.g. `@BOOK` or `@PHDTHESIS`). The key acts as a unique identifier for the reference (it is the user's responsibility to insure uniqueness). Each field is composed of a name, an equals sign (possibly surrounded by white space), and some text delimited either by matching braces or double quotes.

[2]Actually, the features of GNU Emacs TeX-Mode allow the use of BibTeX with all members of the TeX family of document formatting systems. However, for the sake of simplicity, we will only consider LaTeX documents.

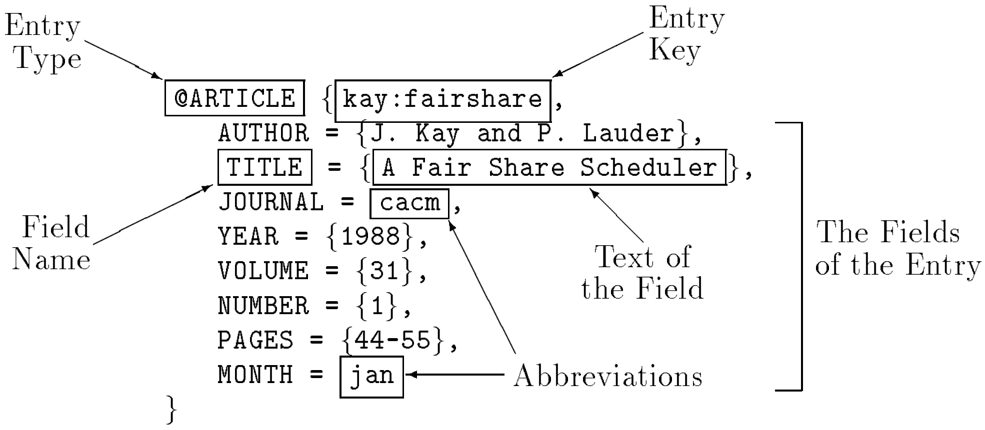

Figure 1: A sample entry from a BibTeX database file.

Each of the fourteen entry types has a (possibly empty) set of required fields and a set of optional fields. If the user fails to include a required field, BibTeX emits a warning but continues to format the reference as best it can. In general, all required and optional fields will appear in the bibliography (some styles may not output certain fields). BibTeX ignores fields for which it does not have formatting instructions, effectively giving the user the ability to add useful non-printing fields.

2.2 GNU Emacs TeX-Mode and BibTeX-Mode

GNU Emacs is a screen-oriented text editor, which runs on both glass-tty devices and bit-mapped displays (using a tty-style interface). Emacs allows the user to have multiple files open and to view several files simultaneously by dividing the tty screen into a number of windows. User input and prompting is performed through a one-line window at the bottom of the screen called the minibuffer.

For our purposes, the most important feature of Emacs is that it contains an embedded Lisp interpreter. By writing programs for this interpreter, it is possible to change key bindings, create keyboard macros, and to add complex features to the editor. Routines that are useful when editing a particular type of file are often collected into a *mode*. These routines can communicate with the user through the minibuffer and can perform any editing operation available to the user.

GNU Emacs TeX-Mode is just such a mode, designed to speed and simplify the production of TeX and LaTeX documents. Its non-bibliographic features include automated insertion of simple commands, spelling checking, indexing support [5], and automated invocation of formatters, previewers, and printers. TeX-Mode provides three bibliographic features:

- *Bibliography preprocessing* scans the LaTeX source file for citations, creates a dummy auxiliary file, and runs BibTeX. This eliminates the need for the first run of LaTeX. This facility can also replace symbolic citations with actual citations in the source file and reverse the process, which allows the use of BibTeX with TeX documents.

- *Automatic citation* prompts the user for a regular expression and then searches for matching entries in his BibTeX database files. When the user tells BibTeX-Mode that the correct entry has been found, a citation of that entry is placed in the document file.

- *Reference inspection* is essentially the inverse of automatic citation. It lets the user view, in a separate Emacs window, the reference corresponding to a particular citation in the document source file.

GNU Emacs BibTeX-Mode is used when editing BibTeX database files. Its most important feature is forms-based editing. As can be seen from Figure 1, the language used to specify references is rather verbose. The forms-based editing routines can create a blank template of a reference for the user to fill in. They also provide entry-wise and field-wise navigation and deletion operations. These routines greatly reduce the user's data entry effort. BibTeX-Mode also supports automatic construction and formatting of draft bibliographies, abbreviations which fill in the text of more than one field, and routines for sorting database files.

3 Design Goals

As with any system in active use, experience with BibTeX-Mode and the bibliographic features of TeX-Mode has identified areas where additional functionality would be desirable. The research described here involved the design and implementation of three new features intended to meet some of these desires. The three new features were:

- A mechanism for attaching annotations to references,

- A more powerful query method for use with the automatic citation mechanism of TeX-Mode,

- Enhancement of the reference inspection mechanism of TeX-Mode to allow the user to view fully formatted references.

This section describes the design goals for each of the three features.

3.1 Annotations

The user of a bibliography processing system may wish to attach various types of annotations to the references in his database. Some examples of possible reference annotations are:

- an on-line version of the original document,
- source code of a relevant program,
- a collection of electronic mail messages discussing the document,
- the user's evaluation of the document in his role of referee for a journal,
- informal notes about the document.

The variety of types of annotation shows clearly that reference annotation is an interesting instance of the general problem of document annotation. Many types of documents can be viewed as using annotations. For instance, comments in a program are a kind of annotation. A link in a hypertext system can be viewed as pointing to an annotation. It is quite possible that a mechanism for annotation of bibliographic references can be extended to other domains.

BibTeX already supports the traditional notion of reference annotation with the `ANNOTE` field, which is intended to be printed in annotated bibliographies. However, not all annotations should appear in an annotated bibliography and others do not belong in the reference database. For example, some annotations, such as the original document, could easily be larger than the bibliography database. The electronic mail messages might well contain characters strings which would cause formatting errors. Also, some of these annotation types present security problems (e.g. the reviewer's comments or program source). It would be odd to place the entire text of a document in an annotated bibliography and it would be inappropriate for a journal referee to store his comments in a shared reference database. Clearly, this approach to annotation is too limited.

After considering these issues, we set the following goals for BibTeX-Mode's annotation facility:

Integration The user should be able to view a reference's annotations at any time that the reference's database entry is visible, without exiting the editor. If the annotation is itself a TeX document or BibTeX database file, all the features of the appropriate mode should be available while visiting it.

Security If a reference database is shared, the user must be able to make his annotations as public or private as he wishes. If possible, he should be able to hide the fact that he has made any annotations to a reference at all.

Low Overhead The annotations mechanism should be simple and place only limited additional burdens on the user. It must not require alterations in BibTeX or TeX or require that the user maintain special files. Encryption-based security is possible but is outside the scope of the present paper.

3.2 Improved Queries for Automatic Citation

The purpose of the automatic citation mechanism of TeX-Mode is to help the user quickly cite a reference without having to remember its unique identifier. Instead of remembering the unique identifier, the user must provide a regular expression [22] that matches something in the reference. TeX-Mode then searches through the database for matching references. The user inspects them one by one until he finds the correct reference, at which point he instructs the system to insert a citation of that reference into the document he is editing.

It is possible to express a wide range of complex queries using regular expressions. However, it is not always easy. Suppose the user wants to locate a reference whose authors include both Chen and Harrison. In TeX-Mode, he would use the case-insensitive regular expression,

```
author *= *{[^}]*\(chen[^}]*harrison\|harrison[^}]*chen\).*}
```

This regular expression is not a simple construction, because it must account for alternate orderings of the two names and the possibility of intervening text. In fact, it took a minute to conceive and correctly type it. Yet, for all its complexity, there exist correct BibTeX reference entries which it would not match, because, for instance, it ignores the possible use of tabs or newlines instead of spaces to separate the word 'author' from the equals sign. Because of this complexity, searches are usually performed with simple regular expressions. Often the regular expression will just be a single word, such as an author's last name or a word that appears in the title. These simple regular expressions cannot specify which field a word appears in or specify alternate orderings of words separated by intervening text.

When searching small reference databases, simple search queries may be adequate. However, when used to search large databases, they are likely to match many irrelevant references. For instance, suppose the user knows that the reference he wishes to cite was written by Knuth and contains the word 'sorting' in its title. He could search his database using either 'Knuth' or 'sorting' as his one word query. Whatever word is chosen is likely to match many more references than a query which could specify the presence of both words and the fields in which they were expected to appear.

In light of the shortcomings of regular expressions, we decided to implement an improved query mechanism. The approach we took was suggested by a synthesis of the Query-by-Example database query language [8] and the forms-based editing provided by BibTeX-Mode. A sample query is presented in Figure 2. This query would match any entry whose

```
@QUERY{,
        ANYFIELD = {},
        AUTHOR/EDITOR = {harrison chen},
        TITLE = {multiple},
        JOURNAL/BOOKTITLE = {},
        INSTITUTION/ORGANIZATION/SCHOOL = {},
        YEAR = {198[6-8]}
}
```

Figure 2: A sample forms-based query

authors or editors included Harrison and Chen, contained the word "multiple" in its title, and was published between 1986 and 1988. More precisely, a query is constructed by placing sets of regular expressions in the fields of the query template. If multiple regular expressions are placed in a field, they must be separated by white space. If a regular expression contains white space, it must be enclosed in double quotes. For an entry to match the query, it must contain every regular expression in the query and each regular expression must appear in its specified field. Regular expressions that appear in fields with multiple labels (e.g. `AUTHOR/EDITOR`) may be matched by text appearing in either type of field in the reference entry. Regular expressions that appear in the "`ANYFIELD`" field of the query may be matched by any text in the reference entry. We call this approach a *forms-based query* because the user specifies his query by "filling out a form".

The forms-based query has a number of advantages over regular expressions. The user can easily specify in which field a string occurs, although this is not necessary. It is also straightforward to specify multiple strings appearing in different fields. Since no assumptions are made about the order in which regular expressions occur, it is not necessary to specify all possible orderings (contrast Figure 2 with the regular expression example above). Finally, it appears informally that is easy for users to compose correct forms-based queries. The same cannot be said of complex regular expressions.

In addition to the general form of the new query method, we established several other design goals. First, the query template shown in Figure 2 represents a reasonable default form, but it may not suit all users. For example, the author and editor fields were combined because they seldom occur together and both contain similar types of information (i.e. names). Some users may prefer to keep fields separate that are currently combined. They may wish to reduce the size of the template by eliminating fields or wish to change their order. Therefore, the choice of which fields appear and their order should be user-definable. A second goal is that the new query method should not be substantially slower than the

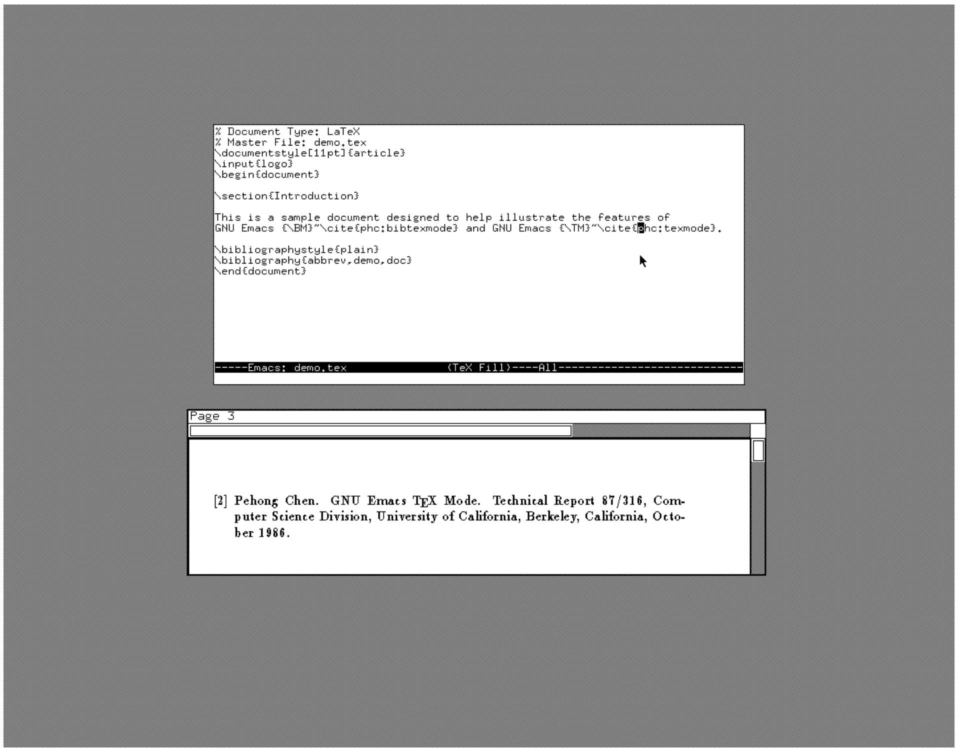

Figure 3: Sample display of fully-formated reference inspection.

regular expression based search. The last design goal was that the new query mechanism should be implemented as an independent component which can later be used for other search-based operations.

3.3 Enhanced Reference Inspection

The original implementation of reference inspection suffered from a major shortcoming. When the user invoked the reference inspection routine, what he wanted to see was something like what appears in Figure 3. Instead what he saw under the original implementation is part of the bibliography file produced by BIBTEX. This text contained many macros in which the user has no interest. For example, this partially formatted version of the same TEX-Mode reference is shown in Figure 4.

The goal of the third bibliographic enhancement was to support the display of fully formatted references, such as shown in Figure 3. This goal was chosen for two reasons. First, it would enhance the aesthetic qualities of TEX-Mode. Second, it would demonstrate

```
\bibitem{phc:texmode}
Pehong Chen.
\newblock Gnu emacs {\TeX} mode.
\newblock Technical Report 87/316, Computer Science Division, University
  of California, Berkeley, California, October 1986.
```

Figure 4: Sample display of partially-formatted reference inspection.

that TEX-Mode could be used to control the on-screen page previewers [12, 16] developed as part of the VORTEX project. While TEX-Mode was already able to start the previewers, it did not control them once they had been started. Thus, the implementation of this feature demonstrates a new level of integration between the previewers and the editing environment.

4 Implementation

This section describes the implementation of the three enhancements whose design goals were presented in the previous section.

4.1 Annotations

The annotation mechanism was implemented as part of BIBTEX-Mode. Annotations are stored in files whose names are recorded in a new field, the `NOTEFILES` field. For example, a reference might contain the line:

```
NOTEFILES = {/usr/public/publicnotes, mynotes, hisnotes}
```

which names three different files of annotations. The user can examine annotations by placing the editor's cursor somewhere within the text of the entry and typing a sequence of two keys. He is then given the option of viewing each file in the list, one at a time. If he chooses to view a file, it is shown in a separate window of the editor. The user may then decide to move to the next file in the list or to edit the file of annotations. New files of annotations can be added, simply by adding new file names to the list.

Security of annotations is inherited from features of the file system. If a user wishes to prevent others from modifying or even from having access to a set of annotations, he simply places them in a file to which only he has access. Then, even though others know that such a file exists, it is protected to the same extent any other file can be protected. However, the list of file names is not protected.

The file names recorded in the `NOTEFILES` field may be either *absolute* or *relative*. Absolute file names have only one interpretation to BibTeX-Mode. When a relative file name is given, a two step search is made. First, the file is searched for in the context of the current working directory. If an accessible file of that name is not found, then the second step is performed. In the second step, the file is located by moving to each directory in the path defined by the environment variable `BTXMODENOTEFILES` and searching for the file from this new context.

The use of relative file names improves security because it allows a user to place a name for his annotation file in the reference database without fully specifying its location. Creative use of this feature can allow a user to cloak both his ownership of the file and even the file's existence. The disadvantage of allowing relative file names is best illustrated with an example. Suppose that user A creates an annotation file `foo` and records its relative name in the shared bibliographic database. If user B also has a file `foo` somewhere in the directory path specified by his `BTXMODENOTEFILES` environment variable, he will be led to believe that his file `foo` contains annotations for that particular reference when it probably does not.

Experience with the annotations facility of BibTeX-Mode has shown it to be useful. The most common use of the facility has been to attach informal notes to a reference, particularly sources for which it is impractical to maintain physical copies. The user interface is acceptable, but it is probably somewhat too rigid. In particular, it does not easily support simultaneous viewing of multiple annotation files.

4.2 Forms-Based Queries for Automatic Citation

When the user invokes the automatic citation mechanism, he is asked to provide a bibliographic file name, as before. Then, instead of being asked for a regular expression, a new Emacs window is opened which contains a form like that shown in Figure 2, only not filled in, and a recursive editing session is begun. The Emacs window which contains the query operates under BibTeX-Mode, so the user is able to use familiar commands to move from field to field. When the query has been completed, the user ends the recursive editing session. The query is parsed and the database files are searched for matching references. The user then browses the references in precisely the same manner he did under the previous implementation. From the user's point of view only the manner in which queries are stated has changed.

There are two user definable options. The user can choose whether to use regular expressions or forms-based queries by setting a boolean variable. This value of this variable can be toggled with short key sequences during the editing session. Also, the fields which appear in the query are defined by a GNU Emacs Lisp variable. This variable holds a list whose elements are lists of the field names which will appear on each line of the query. For example, in the form shown in Figure 2, the line labelled `AUTHOR/EDITOR` corresponds to the list element `("AUTHOR" "EDITOR")`. It is likely that most users will set these options by placing the relevant commands in the startup file used by GNU Emacs.

One of the goals for this implementation was that the performance of forms-based query be comparable to that for a regular expression search. To test whether the implementation fulfilled this goal, a series of informal tests were run to compare its performance to that of the original regular expression method when searching a database of 1000 entries. The results of these tests indicate that the forms-based query method is about 20% slower than the regular expression method for searches having no "false positives". However, when a regular expression search did generate false positives, the user had to examine a number of matching entries and reject them by striking the appropriate key. For searches with many false positives, the need for user interaction can slow down the search quite a bit. In an extreme case where there were nineteen such false positives, a forms-based query (which generated no false positives) was found to be at least 35% faster than the regular expression method.

This implementation fulfills the goals set for it. The informal measurements described above indicate that it is able to locate references based on a forms-based query nearly as fast as the regular expression search does. The display, editing, and parsing of the query is implemented as an independent set of functions which can be used by other routines. The user is able to freely define the fields which will appear in the query and which fields will be combined. Finally, experience in using the new queries has been positive. The queries are easy to compose and their expressive power is great enough to justify the small keystroke overhead required to move from line to line and end the recursive editing session.

4.3 Enhanced Reference Inspection

Reference inspection has two steps. In the first step, the main document file is searched for a citation. Then, the most recent file of output from BibTeX must be searched for the corresponding reference. The implementation of this step was not altered. The second step displays the reference on the screen. It was to this portion of the reference inspection facility that the enhancements were made.

The implementation of the enhanced reference inspection feature of TeX-Mode depends heavily on Steven J. Procter's `dvi2x` [16] and its auxiliary programs. `Dvi2x` is a previewer for TeX `dvi` files which runs under the X window system. Commands for `dvi2x` can be issued either by the user (via the mouse and keyboard) or by other programs (via a UNIX socket with a well-known identifier). One of `dvi2x`'s auxiliary programs is `dvisend`, which is used to send messages to the previewer over this socket. Any command which is available to an interactive user of `dvi2x` can be invoked using `dvisend`. In particular, it is possible to move the display of `dvi2x` up and down on the page and to jump to specific pages.

Without Procter's foresight in supporting the full functionality of `dvi2x` via inter-process communication primitives, the implementation would have been much more difficult.

The basic notion in the enhanced reference inspection facility was to format the existing bibliography file as part of a dummy document, preview the dummy document using `dvi2x` and then use `dvisend` to move the `dvi2x` window to display the proper reference. The only problem with this approach is that there is no way to know from the bibliography file where a reference will appear on the page or even what page it will be on. Our solution to this problem was to modify the bibliography so that each reference in the bibliography fell on a separate page. Since references are basically just small paragraphs, they should never require more than a page each. Thus, it is straightforward to keep track of the correspondence between the reference's position in the bibliography and its page in the dummy document.

TEX-Mode produces the type of display seen in Figure 3 through a three step process. If the user wishes to inspect the references for '`foo.tex`', he must have already run BIBTEX to generate the file '`foo.bbl`'. TEX-Mode makes a copy of '`foo.bbl`' called `foo++.bbl` which contains pagebreaks prior to each reference. Next, TEX-Mode creates a dummy document file called `foo++.tex`, which has no text but does have a bibliography, and runs LATEX on this file. Finally, `dvi2x` is run to display the resulting output. TEX-Mode controls the display of the references by invoking `dvisend` from within Emacs. Each time a citation is inspected, a `dvisend` process is started which instructs `dvi2x` to display the proper page.

This new implementation of reference inspection has both considerable power and substantial limitations. As can be seen from the screen image in Figure 3, it results in a considerable improvement in display quality. It also demonstrates that a programmable editor can be used to control other interactive programs in a powerful and interesting way. However, the success of the implementation results directly from the fact that it can be assumed that references take up less than one page. There are many related tasks for which WYSIWIG display would be a valuable tool. However, in many cases, it is not possible to have the *a priori* knowledge about the formatted version of the document that is necessary to construct this type of facility.

5 Related Work

5.1 Annotations

The design of the reference annotation facility of GNU Emacs BIBTEX-Mode was influenced by Van De Vanter's earlier work on *bibview*, a browser for `bib` databases [27]. *Bibview* allowed the user to attach a single file of annotations to each reference in the database by placing its name in the `%Z` field, which `bib` does not normally use. These annotations could then be viewed and edited in a separate window. Van De Vanter has recently released a descendant of *bibview*, called BiblioText [26]. BiblioText has a more general annotation mechanism whereby up to five fields may be defined to hold either *file links* or *cite links*. File links point to files, while cite links point to other citations in the database.

BIBTEX-Mode lacks the sophisticated user interface of BiblioText and has no true browsing facility. However, its annotation facility does have some advantages:

- There are no limits on the number of files of annotations that may be attached to a reference.
- The `BTXMODENOTEFILES` environment variable can be used to gain greater security.
- Because BIBTEX-Mode is part of an integrated editing environment and because changes to BIBTEX database files do not require regeneration of an inverted index (as with `bib`), the user can add file names to the `NOTEFILES` field and view them in the same session.

5.2 Forms-Based Query

There are a number of interesting query methods used by other tools for bibliography processing. One of the most common is the *imprecise citation*. An imprecise citation is just a collection of words which all appear in the reference being searched for. It was first used as the symbolic citation method in the `refer` [11] bibliography processing system. It is also found in most systems which have descended for `refer`, including BiblioText and the reference search program, `lookup`, which accompanies the `bib` system. BIBTEX-Mode's forms-based queries can be used to duplicate imprecise citations by placing all the words of the imprecise citation in the `ANYFIELD` field of the query.

Pro-Cite [15], a bibliography database system for personal computers, uses a system of boolean queries to search its database. These boolean queries can express any possible query. To reduce effort in constructing the queries, Pro-Cite also supports the use of numeric and date ranges and wildcard characters and provides a query construction dialog box. In contrast, the forms-based queries of BIBTEX-Mode can only express disjunction for text within a single field and do not support negation. However, it is not clear whether the additional expressive power of Pro-Cite's boolean query language is worth the increased complexity required for its use.

EndNote [13], another bibliography database system for personal computers, uses a forms-based approach. When the user requests a search, he is presented with a dialog box containing three fields labeled "Author", "Year", and "Text", which correspond to the `AUTHOR`, `YEAR`, and `ANYFIELD` fields seen under BIBTEX-Mode. EndNote does not support regular expressions, wildcards or numeric ranges and the query dialog is not configurable.

Thus, while EndNote uses a query method superficially similar to the forms-based query of BibTeX-Mode, it has considerably less power.

Bib-Tool [7] is a bibliographic front-end to the Postgres database system [23]. It supports database searches using both imprecise citations and a forms-based query method. The only limitations, relative to BibTeX-Mode, to its forms-based queries are that the query form is not configurable and it can not accept multiple regular expressions in a single field.

5.3 Reference Inspection

Two other systems, BiblioText and Grif, can also be said to support reference inspection. Both systems do so because reference inspection is a special case of a more general function they provide.

BiblioText supports reference inspection as a special case of browsing. To perform reference inspection, the user must be viewing the main document file in some other window. Using the mouse and the features of the Sunview [24] selection service, the user copies the citation from that window to the BiblioText prompter. BiblioText then searches the database for any matching references and displays them. There should only be one such reference and, as with all references BiblioText displays, it will be shown in formatted form. Thus, BiblioText can be said to provide reference inspection.

Grif [17], a direct manipulation editor for structured documents, has a much more general approach. In Grif, bibliographic citations are just one instance of a *reference*, which is a link to some other item in the document. One of the many search operations supported by Grif is a search for a "referenced item". The referenced item of a bibliographic citation is its entry in the bibliography. So, by selecting a citation and searching for its referenced item, the user can perform reference inspection.

6 Conclusions and Other Research Questions

These three enhancements to the bibliography management environment provided by GNU Emacs TeX-Mode and BibTeX-Mode are quite successful. The annotations mechanism represents the first feature of the environment which leaves the domain of document processing and moves toward a more general system of academic information management. Relying on well-understood file system primitives, it supports sharing of both bibliographic data and annotations while providing substantial assurances of privacy of annotations. One question that still needs to be answered is whether the current system of handling relative file names is acceptable.

The forms-based query system is a considerable improvement over the previous regular expression mechanism. It is only slightly, if at all, more difficult to form simple queries and complex queries are much more easily expressed. Most personal bibliographic databases contain less than two thousand entries. However, there do exist shared databases with many more entries. The passage of time is bound to create much larger databases with many entries of substantial similarity. Users of these larger databases will need more powerful queries like the forms-based queries described here. The forms-based query is not the most powerful approach, but we believe it offers the best compromise between power and ease of use of any existing system.

The enhanced reference inspection feature succeeds on several fronts. First, it is aesthetically far superior to the earlier text-oriented reference display. Secondly, it is a good example of how the regularity of a class of objects (in this case, formatted references) can be used to make a difficult problem quite tractable. Finally, it clearly illustrates the benefits of building systems which provide full control to both programs and users. Here, the ability to send `dvi2x` the same commands as a user, but from GNU Emacs, was critical to the successful implementation of enhanced reference inspection.

GNU Emacs TEX-Mode and BIBTEX-Mode are an evolving editing system. In general, they represent an attempt to emulate a monolithic, integrated editing and formatting environment using a loosely connected set of programs. The enhancements to this system have extended it into new areas and thus illustrate the power of a system based on a programmable editor.

There is room for further research on bibliography processing. The most fundamental problem with current bibliography processing systems is that they use simple text files as databases. These systems provide only limited assistance to the user for assuring the correctness of database entries and of citations. A bibliography system which could integrate a real database with the document formatting aspects of bibliography processing would be a substantial advance. One possible model is that of a bibliography server. The server would allow users on many machines to input, retrieve, and manipulate entries in a common database. It would communicate through a well-known interprocess communication port. Text editors, browsers, and document formatting programs could be altered to interact directly with the server. Such a system might be able to support very large reference databases and would naturally support sharing.

Bibliography processing brings together database, hypertext, editor design, and document formatting issues and poses an important question. Should bibliography processing be attacked with a monolithic, integrated system or with a collection of independent routines or with some compromise between the two models? Perhaps the answer for bibliography processing is also the answer for systems in general.

References

[1] J. C. Alexander. `Tib`, *A TEX bibliographic preprocessor*. Department of Mathematics,

University of Maryland, 1986. Version 1.3.

[2] Pehong Chen. GNU Emacs TEX-Mode. Technical Report 87/316, Computer Science Division, University of California, Berkeley, California, October 1986.

[3] Pehong Chen. GNU Emacs BIBTEX-Mode. Technical Report 87/317, Computer Science Division, University of California, Berkeley, California, October 1986.

[4] Pehong Chen and Michael A. Harrison. Multiple representation document development. Technical Report 87/367, Computer Science Division, University of California, Berkeley, California, July 1987.

[5] Pehong Chen and Michael A. Harrison. Index preparation and processing. *Software, Practice and Experience*, 18(9):897–915, September 1988.

[6] Pehong Chen and Michael A. Harrison. Multiple representation document development. *IEEE Computer*, 21(1):15–31, January 1988.

[7] William C. Hunter. BIB-TOOL: A bibliographic references manager. Master's thesis, Computer Science Division, University of California, 1988.

[8] IBM, White Plains, N.Y. *Query-By-Example Terminal Users Guide*, SH20-2078-0 edition, 1978. as described in [25].

[9] Donald E. Knuth. *The TEX Book.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1984. Reprinted as Vol. A of *Computers & Typesetting*, 1986.

[10] Leslie Lamport. *LATEX: A Document Preparation System. User's Guide and Reference Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[11] Michael E. Lesk. Some applications of inverted indexes on the UNIX system. Computer Science Technical Report 69, AT&T Bell Laboratories, Murray Hill, New Jersey, June 1978. Also available in UNIX User's Manual.

[12] Jeffrey W. McCarrell. An overview of dvitool, a TEX dvi previewer under the sunview window system. VORTEX internal report, Computer Science Division, University of California, Berkeley, California, December 1986.

[13] Niles & Associates, 2200 Powell, Suite 765, Emeryville, CA 94608. *EndNote: A Reference Database and Bibliography Maker*, 1989.

[14] Oren Patashnik. *BibTEXing.* Computer Science Department, Stanford University, Stanford, California, January 1988. Available in the BIBTEX release.

[15] Personal Bilbliographic Software, Inc., P. O. Box 4250, Ann Arbor, MI 48106. *Pro-Cite for the Macintosh, User's Manual*, first edition, April 1988. Version 1.3.

[16] Steven J. Procter. Documentation on dvi2x, a TEX dvi previewer under the x window system. VORTEX internal report, Computer Science Division, University of California, Berkeley, California, March 1987.

[17] Vincent Quint and Irène Vatton. Grif: An interactive system for structured document manipulation. In J. C. van Vliet, editor, *Text processing and document manipulation*, pages 200–213. Cambridge University Press, April 1986.

[18] Brian K. Reid. *Scribe: A document specification language and its compiler*. PhD thesis, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, October 1980. Available as technical report CMU-CS-81-100.

[19] R. P. C. Rodgers. Personal Communication, September 1988. Report of `bib` database containing 50,000+ entries.

[20] R. P. C. Rodgers, Kenneth Gardels, and Anat Finkelshtain. Bibix—a bibliographic data base & text formatting system for UNIX. CALM/MedIX Technical Report 86-1.2, UCSF Laboratory Medicine, July 1987.

[21] Michael D. Spivak. *The Joy of TEX*. American Mathematical Society, 1985.

[22] Richard M. Stallman. *GNU Emacs Manual, Fifth Edition, Version 18*. Free Software Foundation, Cambridge, Massachusetts, December 1986.

[23] Michael Stonebraker and Lawrence Rowe. The design of POSTGRES. In *Proceedings, 1986 ACM-SIGMOD International Conference on the Management of Data*, June 1986.

[24] Sun Microsystems, Mountain View, California. *SunView Programmer's Guide, Release A of 17*, February 1986.

[25] Jeffrey D. Ullman. *Principles of Database Systems*. Computer Science Press, second edition, 1982.

[26] Michael Van De Vanter. BiblioText: a hypertext browser for bibliographic data and notes. Technical Report 88/455, Computer Science Division, University of California, Berkeley, California, October 1988.

[27] Michael Van De Vanter. The user interface for *bibview*, a bibliographic browser. Unpublished Report, 1988.