Identifying and Merging Related Bibliographic Records

by

Jeremy A. Hylton

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Master of Engineering in Electrical Engineering and Computer Science

and

Bachelor of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Author ....

Department of Electrical Engineering and Computer Science

February 13, 1996

Certified by ....

Jerome H. Saltzer

Professor of Computer Science and Engineering, Emeritus

Thesis Supervisor

Accepted by ....

Frederic R. Morgenthaler

Chairman, Departmental Committee on Graduate Theses

Identifying and Merging Related Bibliographic Records

by

Jeremy A. Hylton

Submitted to the Department of Electrical Engineering and Computer Science
on February 13, 1996, in partial fulfillment of the
requirements for the degrees of
Master of Engineering in Electrical Engineering and Computer Science
and
Bachelor of Science in Computer Science and Engineering

Abstract

Bibliographic records freely available on the Internet can be used to construct a high-quality, digital finding aid that provides the ability to discover paper and electronic documents. The key challenge to providing such a service is integrating mixed-quality bibliographic records, coming from multiple sources and in multiple formats. This thesis describes an algorithm that automatically identifies records that refer to the same work and clusters them together; the algorithm clusters records for which both author and title match. It tolerates errors and cataloging variations within the records by using a full-text search engine and an n-gram-based approximate string matching algorithm to build the clusters. The algorithm identifies more than 90 percent of the related records and includes incorrect records in less than 1 percent of the clusters. It has been used to construct a 250,000-record collection of the computer science literature. This thesis also presents preliminary work on automatic linking between bibliographic records and copies of documents available on the Internet.

The thesis will be published as M.I.T. Laboratory for Computer Science Technical Report 678.

Thesis Supervisor: Jerome H. Saltzer
Title: Professor of Computer Science and Engineering, Emeritus

Acknowledgments

I would like to thank

- Jerry Saltzer, my thesis adviser, for his detailed criticism of this thesis. He helped me, in particular, to chose my words more carefully and use them more precisely.

- Mitchell Charity for many conversations that helped to shape the design and implementation of the system described in this thesis. He frequently provided a valuable sounding board for ill-formed ideas.

- Tara Gilligan and my parents, Bill and Judi, for their patience and support as I finished a thesis that took longer than any of us expected. My father, who as a writer and editor is so familiar with finishing the manuscript, offered exactly the encouragement and commiseration I needed.

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

Contents

1 Introduction **15**

1.1 What is a digital library? 15

1.2 Integrating bibliographic databases 16

1.3 Overview of related work 18

1.3.1 Networked information discovery and retrieval 18

1.3.2 Libraries and cataloging 20

1.3.3 Database systems 20

1.3.4 Information retrieval 22

2 Cataloging and the Computer Science Collection **23**

2.1 Bibliographic relationships and cataloging 23

2.1.1 Issues in cataloging and library science 25

2.1.2 Taxonomy of relationships between records 27

2.1.3 Expressing relations between Bibtex records 29

2.2 Practical issues for identifying relationships 30

2.2.1 Bibtex record format 31

2.2.2 CS-TR record format 32

2.2.3 MARC records 33

2.2.4 Effects of cataloging practices 34

2.3 Describing the contents of DIFWICS 35
2.3.1 Subjects covered in collection 36
2.3.2 Other characteristics of collection 38

3 Identifying Related Records **41**
3.1 Record comparison in the presence of errors 42
3.2 Algorithm for author-title clusters 43
3.2.1 Comparing author lists 45
3.2.2 String comparisons with n-grams 46
3.2.3 Performance of algorithm 48
3.3 Other systems for identifying related records 50
3.3.1 Duplicate detection in library catalogs 51
3.3.2 The database merge/purge problem 54
3.4 Analysis of author-title clusters 54

4 Merging Related Records **59**
4.1 Goals of merger and outline of process 60
4.2 Creating the union record 62
4.2.1 Problems with union records 63
4.2.2 Refinements to merger process 66
4.3 Clusters sizes and composition in DIFWICS 68

5 Presenting Relations and Clusters **71**
5.1 The basic Web interface 72
5.2 Assessing the quality of union records 73

6 Automatic linking **79**
6.1 Searching for related Web citations 79
6.2 Principles for fully-automated system 81

7 Conclusions and Future Directions **85**

7.1 Future directions 85

7.1.1 Performance, portability, and production 85

7.1.2 Improving the quality of records 87

7.1.3 Identifying other bibliographic relationships 88

7.1.4 Integrating non-bibliographic information 89

7.1.5 Enabling librarianship and human input 90

7.2 Conclusions 91

List of Figures

2-1 Number of citations per year in bibliography collection and *ACM Guide to Computing Literature* . . . 38
2-2 Distribution of records by size . . . 39
2-3 Distribution of Bibtex records, arranged by the number of fields used 40

3-1 N-gram comparison of two strings with a letter-pair transposition . . 48
3-2 Errors in author-title clusters for 937-record sample . . . 55
3-3 Two falsely matched records . . . 56

4-1 CS-TR records for one TR from two publishers . . . 64
4-2 Bibtex records exhibiting the conference-journal problem . . . 65
4-3 Variation in fields values within author-title clusters of the same type 70

5-1 Sample author-title cluster display from Web interface . . . 74
5-2 Source match and field consensus ratios for cluster including a false match . . . 77
5-3 Source match and field consenus ratios for correct cluster . . . 77

List of Tables

2.1 Versions of the *Whole Internet Catalog* cataloged in the OCLC union catalog . . . 27
2.2 Bibtex document types . . . 32

3.1 Statistics for size of potential match pools . . . 50

4.1 Cluster sizes . . . 68
4.2 Souce records, by type . . . 69

Chapter 1

Introduction

The growth in the volume of information available via computer networks has increased both the usefulness of network information and the difficulty of managing it. The information freely available today offers the opportunity to provide library-like services. Organizing the information, which is available from many sources and in many forms, is one of the key challenges of building a useful information service. This thesis presents a system for integrating bibliographic information from many heterogeneous sources that identifies related bibliographic records and presents them together. The system has been used to construct the Digital Index for Works in Computer Science (DIFWICS), a 240,000-record catalog of the computer science literature.

1.1 What is a digital library?

The term digital library is used widely, but little consensus exists about what exactly a digital library is. Before discussing the kind of information sources used to construct the library and how these sources were integrated, it is useful to clarify the term and explain how the particular vision of a digital library affected the design of the system.

The digital library envisioned here is a computer system that provides a single point of access to an organized collection of information. *The* digital library is, of course, not a single system, but a loose federation of many systems and services; nonetheless, in many cases it should appear to operate as a single system. Providing interoperability among these systems remains a major research topic [25].

One kind of interoperability that is sometimes overlooked is the interoperation between physical and digital objects in the library. Most documents exist today only on paper, and digital libraries must provide access to both paper and digital documents to be able to satisfy most users' needs.

The components of the digital library serve many purposes, including the storage of information in digital repositories, providing aids to discovering information that satisfies users' needs, and locating the information that users want. The integrated database of bibliographic records addresses the discovery process. It offers an index of document citations and abstracts.

1.2 Integrating bibliographic databases

The emphasis of the research reported here is to make effective use of the diversity of bibliographic information freely available on the Internet. At least 450,000 bibliographic records describing the computer science literature are freely available, mostly in the form of Bibtex citations; they describe a large part of the computer science literature, but they very widely in quality and accuracy. Nonetheless, when combined with papers available from researchers' Web pages and servers and with on-line library catalogs, these records provide enough raw data to build a fairly useful library.

The bibliographic records present many challenges for creating an integrated information service. The records contain typographical and cataloging errors; there are many duplicate records; and there are few shared standards for entering information in the records. The records come from many sources: Some are prepared by librarians for their patrons, others come from individual authors' personal collections or are assembled from Usenet postings.

Although the heterogeneity of the source records poses a problem for combining all the records into a single collection, it can also be exploited to improve the quality of the overall collection. The heterogeneity provides considerable leverage on the problems of extracting the best possible information from records and providing links between closely related records (an observation made by Buckland, et al. [8]). By identifying related records, a union record can be created that combines the best information from each of the source records.

One of the primary contributions of this thesis is an algorithm for identifying related bibliographic records. The algorithm finds records that share the same author and title fields and groups them together in a cluster of records that all describe the same work; the algorithm works despite variations and errors in the source records. The clusters produced by the algorithm may include several different but related documents, such as a paper published as a technical report and later as a journal article. A catalog of the records, with record clusters presented together, forms the core of DIFWICS.

DIFWICS is intended primarily as an aid to the discovery process, helping users to explore the information collected in the library. In addition to the index of bibliographic records, it provides a simple system for locating cited documents when they are available in digital form. This second system works equally well as a means of locating documents in a traditional library's catalog, which helps to locate physical copies, and provides the groundwork for a more ambitious automatic linking project.

The preliminary automatic linking work, presented in Chapter 6, uses Web indexes like Alta Vista to find copies of papers on the Web. When papers are available on the Web, they are usually referenced by at least one other Web page, which includes a standard, human-readable citation along with a hypertext link. Some of the same techniques used to identify related bibliographic records in DIFWICS can be used to find related citations on the Web and to link the Web pages to the bibliographic records.

Two important characteristics of my work set it apart from related work. First, the collection is assembled without specific coordination among the information providers; this limits the overhead involved for authors and publishers to make their documents available and makes data from many existing services available for use in the current system. The second novel feature of this research is the algorithm for identifying related records. Systems for identifying related records in library catalogs and database systems use a single key for record comparison; my algorithm uses a full-text index of the records to help identify related records. The details of the duplicate detection algorithm are presented in Chapter 3.

1.3 Overview of related work

This thesis builds on research in both computer science and library science. The problem of duplicate detection, for example, occurs in somewhat different form in the database community (the multidatabase or semantic integration problem) and in the library community (creating union catalogs and the de-duplication problem). This section gives an overview of some related areas of research.

1.3.1 Networked information discovery and retrieval

Networked information discovery and retrieval (NIDR) is a broad category encompassing nearly any kind of information access using a large scale computer network [26]. DIFWICS is an example NIDR application and is informed by a variety of work in this area.

This thesis touches at least tangentially on many NIDR issues—including locating, naming, and cataloging network resources—but the clearest connection is to several projects that have used the World-Wide Web and other Internet information resources to provide access to some part of the computer science literature. These systems work primarily with technical reports, because they are often freely available and organized for Internet access by the publishing organization.

The Unified Computer Science Technical Report Index (UCSTRI) [43] automatically collects information about technical reports distributed on the Internet and provides an index of that information with links to the original report.

The Harvest system [6] is a more general information discovery system that combines tools for building local, content-specific indexes and sharing them to build indexes that span many sites; these tools include support for replicate and caching. The Harvest implementors developed a sample index of computer science technical reports. Harvest was designed to illustrate the principles for scalable access to Internet resources described in Bowman, et al. [7].

A third system is the Networked Computer Science Technical Report Library (NCSTRL) [12], which uses the Dienst protocol [21]. Dienst provides a repository for storing documents, a distributed indexing scheme, and a user interface for the documents in a repository.

All three systems rely, in vary degrees, on publishers for making documents available and providing bibliographic information. The publisher-centric model is limiting. Information not organized by publishers, including most online information, lacks standards for naming and cataloging.

Other systems, like Alta Vista and Lycos, index large quantities of information available via the World-Wide Web. They are not selective about the material they index, and are somewhat less useful for information retrieval purposes as result. If a user is looking for a particular document, however, and has information like the author and title, these indexes can be quite useful for finding it (if it is available). Not only do these Web indices index the papers themselves, but they index pages that point to the papers. Chapter 5 discusses some possibilities for integrating these citations with the more traditional bibliographic citations used to build the computer science library.

1.3.2 Libraries and cataloging

The recent development of library-like services for using network information, like UCSTRI or Harvest, parallels the traditional library community's development of large-scale union catalogs and online public access catalogs in the late 70s and early 80s. The OCLC Online Union Catalog merged several million bibliographic records and developed one of the first duplicate detection systems [18].

More recently, the library community has begun to re-evaluate its cataloging standards. Several papers [5, 15, 40, 48] suggest that catalogers should focus more on describing "works"—particular, identifiable intellectual works—rather than "documents"—the particular physical versions of a work. For example, Shakespeare's play *Hamlet* is a clearly identifiable work; it has been published in many editions, each a "document."

This thesis makes use of this distinction when it labels as duplicates records for different documents that instantiate a particular work. Levy and Marshall [24] raise similar questions about how people actually use libraries in their discussion of future digital libraries.

1.3.3 Database systems

Heterogeneous databases differ from more conventional database systems because they included distributed components that do not all share the same database model;

the component databases may have different data models, query languages, or schemas [13]. One of the problems that arises in multidatabase systems is the integration of the underlying schemas to provide users with a standard interface.

Duplicate detection is closely related to integration of heterogeneous databases, but is complicated by the fact that bibliographic formats impose little structure on the data they contain; the wide variations in quality and accuracy that typify collections of Bibtex records further complicate the problem. Papakonstantinou, et al. [30] present a more thorough discussion of the differences between the integration of databases and the integration of information systems like bibliographic record collections. The merge/purge problem described by Hernández and Stolfo [17] implements a duplicate detection system for mailing lists that copes with variations and errors in the underlying data by making multiple passes over the data, each time using a different key to compare the records.

Mediators [45] are a different approach to the problem of integrating information from multiple sources. Mediators are part of a model of the networked information environment that includes database access as its lowest level and users and information gathering applications at the top level. The mediators operate in between the databases and the users, providing an abstraction boundary that captures enough knowledge about various underlying databases to present a new, uniform view of that data to users.

Data warehousing extends the mediator model by creating a new database that contains the integrated contents of other databases rather than providing a dynamic mediation layer on top of them. DIFWICS integrates distributed bibliographies in a similar way.

1.3.4 Information retrieval

Duplicate detection in information retrieval is at the opposite end of the spectrum from database schema integration; information retrieval deals with the full-text of documents with little or no formal structure. Duplicate takes on a wider meaning in this field: Consider a search for stories about an earthquake in a collection of newspaper articles; there are probably many stories about the earthquake, from many different news sources. The stories are duplicates because their content overlaps substantially and not because of some external feature like their title or date of publication. Yan and Garcia-Molina [49] provide a more detailed discussion of duplicate detection in this context.

Chapter 2

Cataloging and the Computer Science Collection

This chapter introduces the conceptual framework for creating, using, and relating bibliographic records. It also discusses some of the more practical issues associated with the specific records used in the experimental computer science library.

2.1 Bibliographic relationships and cataloging

Bibliographic records have traditionally described specific physical objects or units of publication. This thesis describes a different use of bibliographic information and a different focus for cataloging: I use bibliographic records to describe a *work* instead of a particular *document.* The term *work* is used here to mean a unit of intellectual content, which may take on one or more published forms (or none at all); each published form is a document. A paper, presented at a conference and later revised and published in a journal or collected in a book, is a work, each version a different "document."

I emphasize the work over the document because I believe that the primary use of the library catalog is to find works. A patron looking for a copy of *Hamlet* is usually looking for the work *Hamlet*, and, depending on circumstance, any particular copy of *Hamlet* might do. The library catalog should help patrons identify the works available and then chose a particular document, based on the patron's needs.

The MIT library catalog, for example, returns a list of 17 items as the result of a title search for "Hamlet." It takes careful review to determine that the list contains eight copies of Shakespeare's play, three recorded performances, one book about performances, three musical works inspired by the play, and two copies of Faulkner's novel titled *The hamlet*. (Copies that exist in various collected works do not appear in the search results.)

The Hamlet search illustrates the problem of cataloging particular documents but not the works they represent. Deciphering the search results took several minutes and required study of the source bibliographic records to determine exactly what each item was. The search results would be easy to understand if they were organized around works and relationships. The eight copies of the play are all versions of the same work, *Hamlet*, and the performances might be considered derivative works. In a music library, it might be useful to highlight the three distinct musical works, and note that each is related to the play.

This chapter discusses recent work in library science that suggests catalogers should focus on the work instead of the document. It presents a taxonomy of bibliographic relations, which helps to clarify the difference between a work and a document, and it discusses the experimental collection these theories will be applied to, a collection of 250,000 computer science citations.

2.1.1 Issues in cataloging and library science

Standards for cataloging and for using bibliographic information are a current subject of research in the library community. Two majors themes run through several recent papers [5, 15, 37, 48]:

- Library catalogs should make is easier for users to understand relationships between different entries. In particular, catalogs should identify particular works.
- Increasingly bibliographic information is being used to find information in a networked environment, where the catalog of bibliographic records is less likely to described the contents of local library and more likely to be a union catalog.

The most recent theoretical framework for descriptive cataloging was formulated in the 1950s by Seymour Lubetzky. According to Wilson [48], Lubetzky suggested the library catalog serves two functions: the finding function and the collocation function.

The finding function. If a patron knows the author, the title, or the subject of the book, the catalog should enable him or her to determine whether the library has the book.

The collocation function. The catalog should show what the library has by a particular author or on a particular subject, and it should show the various editions or translations of a given work.

Current cataloging practice places more emphasis on the first function than the second. This emphasis is strange, Wilson says, because most discussions of the theoretical background conclude that patrons are not interested in particular documents, so much as in the works they represent. The catalog standards result partly from the historical development of catalogs as simple shelf lists and partly from the ease of cataloging discrete physical objects, rather than works, which might constitute only part of an object or span several of them.

The emphasis on the physical object over the work is inadequate in several ways.

The user's interest in the work rather than the document has already been noted. Smiraglia and Leazer [37] note that anecdotal evidence supports this claim and that catalog usage studies show that the bibliographic fields used to differentiate between variant editions are seldom used.

Library catalogs and other collections of bibliographic records are used increasingly as networked information discovery and retrieval tools, where discovering what kinds of works exist is more important than finding out which works are in the local library. When a user wants an item, there are many other retrieval options other than the local library, including an Internet search and inter-library borrowing.

Trends in publishing make the bibliographic model increasingly unwieldy: Electronic publishing and advances in paper publishing technologies have made it easy to change and update documents. As a result, it is now common for each new printing of a book to incorporate some corrections and additions or for authors to publish electronic copies of their papers that include changes made after print publication. OCLC cataloging rules require a new bibliographic record be created for each copy of a work that has different date of impression and different text. Heaney [15] cites a message from Bob Strauss to the Autocat mailing list that describes the problem; Strauss observes that between the original publication of Ed Krol's *Whole Internet Catalog* in 1992 and his search on Dec. 2, 1993, nine different versions have been cataloged in the OCLC union catalog (see Table 2-1).

The problem raises two questions: First, is it sensible to create new records to describe each version of a document? Second, should the average user be exposed to this level of detail? The answer to the first question is unclear; the answer to the second, in many cases, may be no.

Number	Holdings	Feature
1	323	1992
2	10	minor corr, 1992
3	883	[Corr ed]
4	968	1st ed (same as #1?)
5	51	July 1993, minor corr.
6	136	“May 1993”
7	116	[Corr ed] (1993)
8	19	[Corr ed]
9	132	“Feb. 1993; minor corr”
total	2638	

Table 2.1: Versions of the *Whole Internet Catalog* cataloged in the OCLC union catalog

2.1.2 Taxonomy of relationships between records

Tillett [41] and others have developed taxonomies of bibliographic relationships. Tillett's taxonomy provides a useful vocabulary for discussing the different kinds of documents that describe the same work, as well as relationships between different works. The seven categories presented here are based on Tillett's taxonomy, although some of the categories are slightly different.

Equivalence relationship. Equivalence holds between records that describe the same document but in different mediums, e.g. reprint, microfilm, original book.

Derivative relationship. The derivative relation holds between different versions of the text, e.g. different editions or translations, arrangements or adaptations.

Referential relationships. The referential relationship holds when one document explicitly contains a reference or link to another document. Tillett describes reviews, critiques, abstracts, and other secondary references as “descriptive” relations; the referential category expands Tillett's definition to include other kinds of references between works, including citations of one work within another and survey articles.

In an online environment, where a paper's citation list may be as accessible as standard bibliographic information, this expanded notion seems useful.

Sequential relationship. The sequential, or chronological, relationship describes documents intentionally published as a sequence or group. Examples included the successive issues of a journal or volumes of a book series, like an almanac or encyclopedia.

Hierarchical relationship. The hierarchical relationship holds between the whole and the parts of a particular work. It applies to relations between a book and its chapters, and vice versa, or articles in a journal or collection. (Tillett prefers the term "whole-part" relationship.)

Accompanying relationship. The accompanying relationship holds between two documents that are intentionally linked, but not necessarily in a hierarchical relationship. Examples include a textbook and an instructional supplement, or a concordance, index, or catalog that describes another work (or group of works).

"Shared characteristic" relationship. The "shared-characteristic" relationship holds between bibliographic records that have the same value in a particular field. It is a kind of catch-all category that describes almost any sort of relationship; the relation seems to be most useful for fields like publisher, subject heading, or classification code, but year of publication or page count are also potentially shared characteristics.

User queries are another example of a kind of shared characteristic relationship. The results of a query all share the particular characteristic described by the query.

2.1.3 Expressing relations between Bibtex records

Identifying and representing each of the relationships described in the previous section is beyond the scope of this thesis. Instead, I focus on identifying a limited set of relationships and presenting the rough form of a user-interface for those relationships.

The algorithm presented in the next chapter identifies related records based on the author and title fields; if the fields are the same, it concludes the records describe the same work. Two relationships hold between records in such a cluster: equivalence and derivative relationships; some records will be duplicate citations of the same document and others will cite different documents that represent the same work.

Identifying works is difficult. Even when a human cataloger is reviewing two bibliographic records, it can be difficult to tell it they describe the same work without referring to actual copies of the cited work. A cluster is an algorithmically-generated set of related records, which may or may not be the same as the actual set of records for a particular work. A cluster generated by one algorithm may be better in one way or another than a cluster generated by a different algorithm. (Indeed, the next chapter describes several algorithms that identify only duplicate records and do not consider works.)

This thesis uses *author-title clusters*, which identifies a work as a unique author and title combination. Any pair of records with the same title and same authors are considered equivalent for the purpose of creating an author-title cluster, although there may be unusual cases where this test does not discriminate between two different works.

Using the term equivalent requires some care; it sounds simple enough, but equivalence depends entirely on the context in which some equivalence test is applied. (Consider, for example, the four different equality tests in Common Lisp [38].) The records in an author-title cluster are equivalent for the purpose of identifying a work, but are probably not equivalent for the purpose of locating the work in a library or retrieving it across the network.

The two uses of equivalent above isolate two separate problems that must be addressed in a catalog that is work-centered, but constructed from bibliographic records that have not been prepared with this use in mind. The first problem is identifying the works described by the records in the catalog. The second problem is identifying the separate documents in the author-title cluster and presenting them as different instances of the main work. This process involves identifying the different documents within the cluster and merging duplicate citations for each document into a single, composite record.

The derivative relation holds between the different documents in an author-title cluster, but identifying each different document is complicated by many factors, including the version problem and the difficulty of relying on Bibtex for finely-nuanced descriptions. I solve a simplified version of the problem by using the Bibtex entry type to identify the different, horizontal classes of records within a cluster. Thus, a cluster might be presented to the user as containing two document types—an article and a technical report—but would not distinguish between, say, different editions of a book.

2.2 Practical issues for identifying relationships

Bibliographic formats affect how well relationships and works can be identified. The format not only dictates what information can be recorded, but also tends to affect the practice of recording.

Many freely available bibliographic records use the Bibtex format [22], which is used to produce citations lists in the LaTeX document preparation system. About 90 percent of the records in DIFWICS are Bibtex records.

Being able to accept bibliographic records in any format is a design goals of DIFWICS, because it minimizes the need for coordination among publishers, catalogers, and libraries and maximizes the number of records available for immediate use.

Because other bibliographic formats, like Refer or Tib, are less common than Bibtex records, the current implementation supports only one other bibliographic format, the CS-TR format developed as part of the Computer Science Technical Report Project (CS-TR) [19] and defined by RFC 1807 [23].

The two formats differ in both syntax and semantics, so using both formats interchangeably requires a common format that both can be converted into. The common format involves some information loss, when one format captures more information about a particular field than the other format is capable of expressing. For example, the CS-TR format has separate fields for authors and corporate authors, but Bibtex has only a single author field for both kinds of author; the common format does not capture the distinction, because it is not possible to determine which kind of author is being referred to in Bibtex.

Several characteristics of the Bibtex and CS-TR formats affect the design of the library and the kinds of bibliographic relationships that can be identified. Bibtex is currently used as the common format, because Bibtex is capable of describing any document that can be described with a CS-TR record (albeit with some loss of information).

2.2.1 Bibtex record format

Bibtex files are used for organizing citations and preparing bibliographies. The format is organized around several different entry types (see Table 2-2), which describe how a document was published. Each type uses several of the two dozen standard fields

Article	MastersThesis
Book	Misc
Booklet	PhDThesis
InBook	Proceedings
InCollection	TechReport
InProceedings	Unpublished
Manual	

Table 2.2: Bibtex document types

to describe the publication.

The format is very flexible. There are few rules governing precisely how a field must be formatted and users are encouraged to define their own fields as necessary.

The individual fields fall into three categories—required, optional, and ignored—depending on the document type; the journal field is required for an Article, but ignored for a TechnReport. Ignored fields allow users to define their own fields. Throughout this thesis, I call the required and optional fields the standard fields and the ignored fields non-standard.

Most of the specific fields are easy to use, process, and understand—like month, year, or journal—but a few fields contain unstructured information about the document being cited, notably note, annote, abstract, and keywords. In practice, the note, annote, and (non-standard) keywords field often appear to be confused; the note field is intended for miscellaneous information to print with a citation and the annote field for comments about the cited document, such as would be included in an annotated bibliography.

2.2.2 CS-TR record format

The CS-TR format was designed specifically for universities and R&D organizations to exchange information about technical reports. The fields it uses are geared specifically towards describing technical reports, allowing a more detailed description of technical reports than standard Bibtex fields, but limits its usefulness for describing other documents.

CS-TR defines a few mandatory fields used for record management and 25 other fields, all of which are optional. Some of the fields can be easily converted to Bibtex—CS-TR date maps to Bibtex month and year (with loss of the day) and CS-TR title is the same as Bibtex title. Most of the CS-TR fields don't have an analogue in the standard Bibtex fields, and must be omitted or placed in a non-standard field or the note field.

2.2.3 MARC records

MARC is the predominant bibliographic format in the library community. While it is not used by the system presented here, the MARC record makes an interesting point of comparison.

The MARC record is a highly structured format; its use emphasizes precise labels for fields and detailed descriptions of the items being cataloged. Crawford [11] provides an overview of MARC and its use in libraries; he observes that all MARC records share five characteristics:

- Each record has a title or identifying name.
- Each object is produced, published, or released at a specific time, by a specific person or group.
- Each object is described physically.
- Most nonfiction objects have subjects—what the object is about.
- Notes are made about what is being cataloged, e.g. restrictions on use or notes about the reproduction.

The MARC format defines several hundred fields, many of which have subfields, that specify the format and content of the field values exactingly. The primary field used for author (field number 100, "Main Entry–Personal Name"), has subfield codes for specifying the personal name, titles or dates associated with the name, and fuller forms of the name; another code indicates whether the personal name begins with a forename, single surname, or multiple surnames.

MARC's precision makes comparing records more difficult for several reasons. There is more opportunity for small errors in MARC; several different fields can be used to enter the same information; and there is some flexibility as to how much information must be entered. Users of electronic library catalogs will probably be familiar with the problem of determining when two author entries are the same—separate listings appear when one record has date of birth, while another has dates of birth and death and a third may contain a fuller form of the author's name.

The practical implication of the differences between MARC records and citation-oriented records like Bibtex is that while Bibtex records are not as rich in information they provide a much simpler structure for extracting information, like author and title, which are used to distinguish between different works.

2.2.4 Effects of cataloging practices

Bibtex's flexibilty allows people to enter information in many ways. The use of abbreviations in fields values is very common, which makes it difficult to compare two fields to see if they have the same value; ignoring capitalization, *Communications of the ACM* is abbreviated variously as "CACM", "C. ACM", "C.A.C.M.", "Comm. ACM", "Comm. of the ACM (CACM)", etc.

Notes about the document, e.g. that it is an abstract only or that it is a revised edition, are entered in many different ways. Although the notes field seems to be the most likely candidate for this information, it is variously entered in the title field (complicating comparisons), in the note or annote field, or in the edition field (where “2nd” is as likely as “second”).

There appears to be less variation among the different sources of CS-TR records, because the definition of each field is fairly specific and because technical reports have fewer unusual cases than other document types.

The problems of abbreviations and variations are less pronounced in CS-TR records because they are produced by the publishing institutions, which tend to be consistent within their own records. CS-TR records are also easier to handle because many of the fields are unused in the records available today; more than half the records use no more than seven descriptive fields.

2.3 Describing the contents of DIFWICS

DIFWICS incorporates bibliographic records from two major collections available on the Internet. The primary source is Alf-Christian Achilles' collection of 450,000 Bibtex records, titled “A Collection of Computer Science Bibliographies” [2]. This work organizes several hundred individual collections of varying size and quality. The second source is the CS-TR records produced by the five participants in the CS-TR project; the collections is composed of approximately 6,000 technical report records from Berkeley, Carnegie Mellon, Cornell, M.I.T., and Stanford.

The first collection requires some explanation to understand what kinds of records it provides and how they affect the system for identifying related records. The individual bibliographies fall into three major categories—personal bibliographies organized by individual researchers, journal and conference proceedings bibliographies, and bibliographies organized around a particular subject. Although Bibtex is commonly used to prepare citation lists for papers, none of the source files suggest they were prepared for that purpose.

There are only a few personal bibliographies, but each is quite large and appears to have been created and checked with some care. Joel Seiferas's collection holds 43,000 theory citations and Gio Wiederhold's collection holds 10,000 citations, mostly about databases.

The journal and conference bibliographies tend to be fairly complete listings of the articles or papers published. The bibliography for the *Journal of the ACM*, for example, includes every article published from 1954 to 1995.

The topical collections range widely from a 5,000-record collection on programming languages and compilers to a 24-record collection on fuzzy Petri nets.

2.3.1 Subjects covered in collection

Achilles has organized the collection into major subject areas and we have selected an arbitrary subset of the records in each category to include in DIFWICS—in all, about 240,000 records; the remainder of the collection had not been processed at the time of this writing. A brief description of each categories and the number of records included from it follows. (Sizes are rounded to the nearest 5,000.)

Artificial Intelligence. 30,000 records covering most areas of AI, with most extensive coverage of logic programming and natural language processing. Includes 13 journal bibliographies.

Compiler Technology and Type Theory. 20,000 records covering programming languages and compilers. Includes five journals and six conferences.

Databases. 20,000 records. Half from personal collection of Gio Weiderhold. Includes one journal and one conference.

Distributed Systems. 10,000 records. Scattered coverage of networking, distributed systems (including Mach and Amoeba bibliographies), mobile computing.

Graphics. 15,000 records covering topic including vision and ray tracing. Includes complete SIGGRAPH bibliography, five journals.

Mathematics. 5,000 records. Computer algebra, *ACM Transactions on Mathematical Systems*, applied statistics.

Neural Networks. 5,000 records. Sampling of a few seemingly representative collections.

Object-Oriented Programming and Systems. 5,000 records. Includes OOPSLA and ECOOP proceedings.

Operating Systems. 10,000 records. Mostly from the Univ. of Erlangen library and a USENIX bibliography.

Parallel Processing. 15,000 records. Broad coverage, including transputers, multiprocessors, numerical methods, parallel vision. Many journals and conferences.

Software Engineering. 10,000 records. Process modelling, specification, verification. Includes three journals.

Theory. 60,000 records. Major topics include concurrency, hashing, logic, term rewriting, cryptography, and computability. Mostly from the Seiferas collection, which includes citation from 34 journals and conferences. Also includes two journals and two conferences.

Miscellaneous. 35,000 records. Records that do not fit into the other categories or span multiple categories. Includes 5,000 record from the SEL-HPC archive, a 3,200-record Communications of the ACM bibliography, and 16 other journals.

2.3.2 Other characteristics of collection

The time period spanned by the DIFWICS collection, the size of the records, and the number of fields they contain also provide some useful measures of the collection.

The year field gives a fairly precise measure of when documents in the collection were published, although some 17,000 records (about 5 percent of the collection) contain year fields where the actual year cannot be recognized automatically.

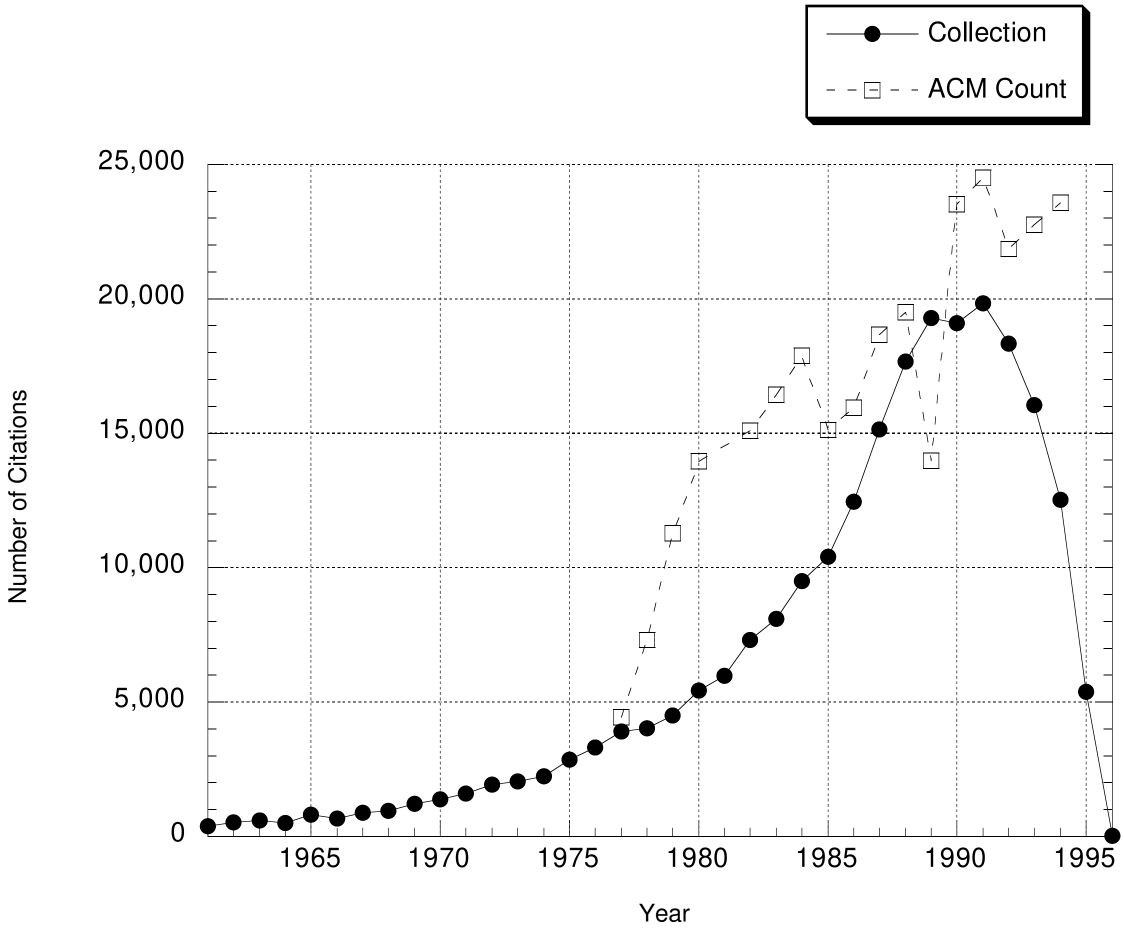

Figure 2-1: Number of citations per year in bibliography collection and *ACM Guide to Computing Literature*

Half the records included in the computer science library cite documents published between 1988 and 1993, with a peak in 1991. The number of citations grows exponentially from 124 citations for documents published in 1956, to 1212 citations for 1969, and 10402 citations in 1985. Figure 2-1 compares the DIFWICS collection to the number of documents cited in the annual *ACM Guide to Computing Literature*.

The number of citations drops sharply after 1991—with 12,527 cites for 1994 and only 5,382 for 1995. It is not clear what the cause of the drop is. Two plausible hypotheses are that Bibtex is becoming less popular for preparing citation lists or that there is a delay of a year or between the time a paper is published and the time it accumulates enough citations to appear in a collection. (Achilles's collection includes the most recent copies of the individual bibliographies, with little aparent delay.)

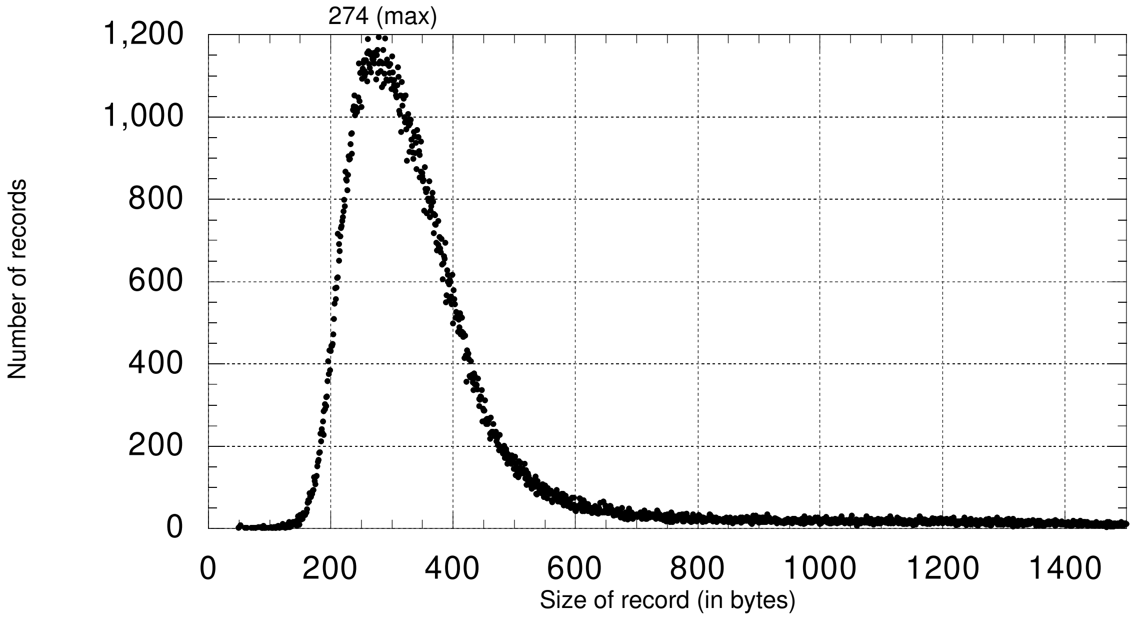

Figure 2-2: Distribution of records by size

The source records range in size from from 50 bytes to 10,000 bytes, but more than 90 percent of them are between [illegible] and [illegible] bytes long with an average length of 397 bytes. (See Figure 2-2.)

The Bibtex records have, on average, entries in 6.7 different fields, and 90 percent of the records use between four and nine fields. (See Figure 2-3.)

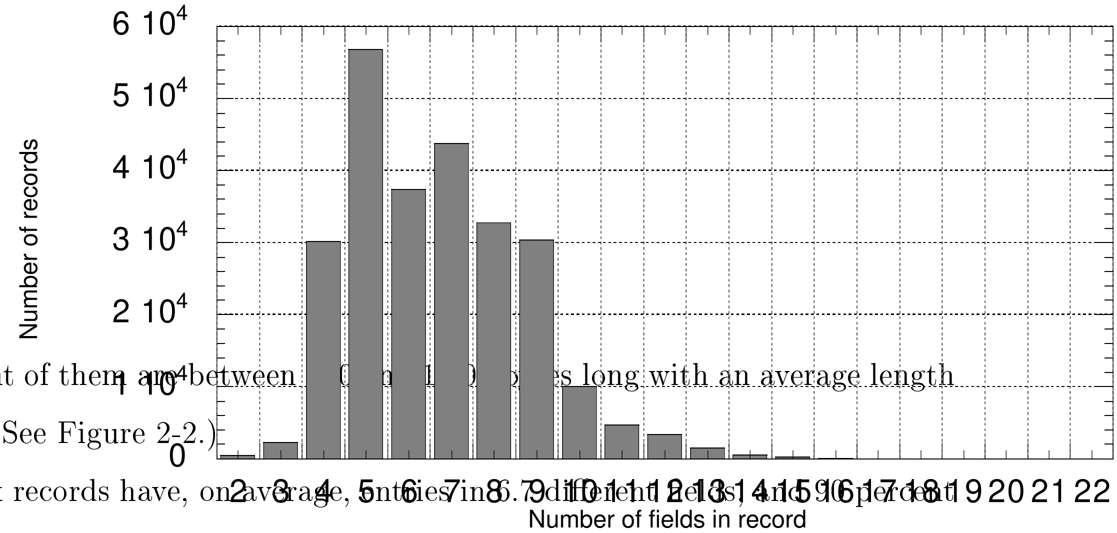

Figure 2-3: Distribution of Bibtex records, arranged by the number of fields used

Chapter 3

Identifying Related Records

Libraries have faced the problem of duplicate records for more than 20 years; as they developed computer systems for managing library catalogs and, in particular, for sharing bibliographic records in a networked environment, duplicate records were identified as a problem and techniques for eliminating them were developed. Traditional de-duplication, however, is significantly different from identifying all the citations of a particular work; in many ways, identifying works is easier.

This chapter describes in detail the algorithm for identifying author-title clusters and the characteristics of the source records that make this problem hard. It presents a novel algorithm that uses randomized full-text searches to detect potential duplicates. The algorithm is analyzed and two kinds of failures are examined—including a record in a cluster when it cites a different work (false match) and creating two different clusters whose members describe the same work (missed match). The chapter also surveys other approaches to identifying related records—two early systems used for de-duplication in library catalogs and a recent multidatabase integration system.

3.1 Record comparison in the presence of errors

The record comparison test addresses three sources of variation between records that describe the same document: entry errors, which are primarily typographic, differences in cataloging standards, and abbreviations.

Entry errors are unintentional errors made during the creation and transcription of bibliographic records. The most common entry errors are simple misspellings and typographical errors, but improperly formatted records and records which omit required fields are also common. Typical format errors include placing information in the wrong field, such as putting the month and year in the year field, or not using the required syntax for a field, e.g. entering the year as a two-digit number instead of a four-digit number.

Misspelling and typographical errors are common to most text databases and the literature about them is substantial. O'Neill and Vizine-Goetz's overview of quality control in text databases [29] provides a thorough, though somewhat dated discussion of the sources of errors and some techniques for correcting them; Kukich [20] surveys techniques for automatically detecting and correcting spelling errors.

While entry errors are an unintentional source of variability, differing cataloging standards are often intentional. Many bibliographies apply a consistent set of cataloging rules, but the standards are quite different from one bibliography to another.

Cataloging standards affect what kind of information is recorded in a citation and where it is recorded. If a citation describes an extended abstract, for example, some bibliographies record that fact in the title field and others record it in the note or annote field.

Cataloging standards also determine what document type the citation is assigned or whether a new citation is created for each instance of publication. When an article is published more than once, some databases include a record that describes one of the publications fully and includes information about the other publication in one of the comment fields. Similarly, many conference proceedings are published as issues of a journal; sometime these articles are cataloged as Bibtex InProceedings citations and other times as Article.

Authors' names are a particular source of trouble. The same name can be cataloged many ways: last name only, first initial and last name, all initials and last name, first name and last name, etc. In a list of authors, some bibliographies include only the first author (only sometimes indicating that there are other authors) and others include all the names.

Unusual names frequently cause problems. In a sample of 1,000 records, 14 uses of "Jr." were found, but none follow the specified format [22]. The correct placement of "Jr." within the author string is: "Steele, Jr., Guy L."

Abbreviations are another source of problems caused by differences in cataloging. Journal and conference names and publishers are abbreviated in many non-standard ways, making comparisons using these fields difficult. (Bibtex allows users to define abbreviations within a Bibtex file; the abbreviations are expanded when a bibliography is processed. These are not the abbreviations discussed here; rather, the problem lies with abbreviations in the actual text of the field.)

The problems posed by abbreviations are not addressed here. Abbreviations are seldom used in the author or title fields, so they do not present a problem for identifying related records. The problem is discussed in somewhat more detail in Chapter 6 in the section on authority control.

3.2 Algorithm for author-title clusters

The algorithm for identifying clusters of related records considers each record in the collection and determines whether it has (approximately) the same author and title field as any other record. There are two primary concerns for the algorithm's design. First, the algorithm should avoid comparing each record against every other record—an $O(n^2)$ proposition. Second, the algorithm, and in particular the test for similarity, must cope with the kinds of entry errors described above.

The algorithm uses two rounds of comparisons to identify author-title clusters. The first round creates a pool of potentially matching records using a full-text search of the entire collection; the pool consists of the results of three queries, with words randomly selected from the author and title of the source record. (The index, however, includes words that appear anywhere in a record, not just in the author and title fields.) In the second round, each potential match is compared to the source record, and an author-title cluster is created for the matching records.

Three different queries are used to construct the potential match pool. Each query includes one of the authors' last names and two words from the title field. Title words that are either one- and two-letters long or are in the stoplist, which includes the 50 most common words in the index, are not used. In cases where there are not enough words to construct three full queries, queries use fewer words. (And in some extreme cases only one or two queries are performed.)

The second phase of testing compares the author and title field of each record in the potential match pool against the source record. It uses an n-gram-based approximate string matching algorithm to compare the two fields. The string matching algorithm is explained in detail below.

The algorithm is applied to every source record, regardless of whether it has been placed in a cluster already; thus, a cluster with five records will be checked by five different passes. The algorithm enforces transitivity between records: If record A matches record B and record B matches C, all three are placed in the same cluster, whether or not A matches C.

The algorithm is tolerant of format errors and typographic errors. Given a pair of records that cite documents with the same author and title, the potential match pool for one record will contain the other as long as one of the three queries contains words that match the second record; a small number of errors is unlikely to cause a problem. The approximate string match will tolerate variations as large as transposed or missing words in a long title string.

3.2.1 Comparing author lists

Comparing two entries' author fields is more complex than comparing title fields, because there is much wider variation in how names are entered than in how titles are entered. Instead of comparing author strings directly, the algorithm performs a more complicated comparison.

To compare two author entries, each author string is separated into a list of individual authors and each name is compared; the name comparison considers each part of the name—first name, last name, etc.—separately. If the names and the order of the names both match, then the two author field are considered to be the same. When one author list contains matching names, in the same order as the first list, but omits names from the end of the last, the two strings are also considered the same; testing suggested it was likely either there was a cataloging error or that the same work had been published with slightly different author lists. Finally, an author list that includes a single name and the notation "and others" will match any list of two or more names that has a matching first author. (The Bibtex format specifies the use of "and others," so variants like "et al." are not handled.)

Individual author names are compared by separating the two name strings into four parts: first name, middle name, last name, and suffix (e.g. Jr.). Two parts of a name match if any of three conditions is met:

1. A trigram comparison reports the strings are the same.

2. One of the strings is an initial and the other string is a name that starts with that initial, or both strings are the same initial

3. Either of the strings is blank.

Thus, "G. Steele" and "Guy L. Steele Jr." are considered the same. However, "B. Clifford Neuman" and "Clifford Neuman" are not considered the same (even though Neuman is cited both ways).

3.2.2 String comparisons with n-grams

An n-gram is a vector representation that includes all the n-letter combinations in a string. The n-gram vector has a component vector for every possible n-letter combination. (For an alphabet which contains the letters 'a' to 'z' and the numbers '0' to '9', the vector space is 36^n-dimensional.) The n-gram representation of a string has a non-zero component vector for each n-letter substring it contains, where the magnitude is equal to the number of times the substring occurs. The string comparison algorithm uses 3-grams, or trigrams, so I will limit further discussion to trigrams.

Formally, a trigram vector $\vec{A}$ for a string s is defined as

$$\begin{aligned} \vec{A}_s \quad &= \{a_{aaa}, a_{aab}, \ldots, a_{d5f}, \ldots, a_{999}\}, \\ \text{where} \quad a_{aaa} &= \text{number of times ``aaa}'' \text{ appears in } s. \end{aligned}$$

The trigram vector for the string "record" contains four components: "rec", "eco", "cor", and "ord." If the trigram "eco" appeared twice in the string, then $a_{eco} = 2$.

The string comparison algorithm forms trigram vectors for the two input strings and subtracts one vector from the other. The magnitude of the resulting vector difference is compared to a threshold value; if the magnitude of the difference is less than the threshold, the two strings are declared to be the same. If each trigram appears only once in an input string, the difference vector is the same as the list of trigrams that appears in one string but not the other. In general, the magnitude of the difference vector for two inputs $\vec{A}$ and $\vec{B}$ is:

$$||\vec{D}|| = \sqrt{\sum_{i=aaa}^{999} (a_i - b_i)^2}$$

The threshold value was determined experimentally, using several dozen pairs of strings which had been labelled the same or different, based on how the algorithm should compare them. The threshold that worked best varied linearly with the total number of distinct trigrams in the two input strings. The threshold T used for comparing two strings with a total of n trigrams is

$$T = 2.486 + 0.025n$$

The variable threshold allowed the test to tolerate several errors in a long title string, such as missings words or multiple misspellings, and still work well with short title strings. A higher, fixed threshold would have admitted many short strings that had completely different words.

The threshold was chosen using a collection of about 50 pairs of strings, each of which was labelled *a priori* as the same or different. The computed threshold minimized the number of pairs that was incorrectly identified. The sample strings, however, were not chosen with great care and the effects of small variations in the threshold were not studied.

The trigrams are produced by converting all strings to lower case letters and numbers; all spaces and punctuation are removed. Most n-gram implementations pad the string with leading and trailing spaces, so that "record" would produce a

$$
\begin{aligned}
s_1 &= \text{“Machine Vision”} \\
\vec{A}_{s_1} &= \{ \text{ mac, ach, chi, hin, ine, nev, evi, vis, isi, sio, ion } \} \\[1ex]
s_2 &= \text{“Machien Vision”} \\
\vec{A}_{s_2} &= \{ \text{ mac, ach, chi, hie, ien, env, nvi, vis, isi, sio, ion } \} \\[1ex]
\vec{D} = \vec{A}_{s_1} - \vec{A}_{s_2} &= \{ \text{ hin, hie, ine, ien, nev, env, evi, nvi } \} \\[1ex]
\|\vec{D}\| &= \sqrt{1^2+1^2+1^2+1^2+1^2+1^2+1^2+1^2} = 2.828 \\
&\leq T = 2.486 + 0.025 \cdot 15 = 2.861
\end{aligned}
$$

Figure 3-1: N-gram comparison of two strings with a letter-pair transposition

trigram including "␣␣r" and "␣re." My implementation, however, begins the first trigram with the first letter and ends with the trigram that includes the last three letters; this test is slightly more tolerant of errors at the beginning and end of strings.

Consider the example comparison in Figure 3-1 between two strings "Machine Vision" and "Machien Vision." The difference vector $\vec{D}$ contains eight trigrams, each appearing once, so the magnitude of the difference is 2.828. The threshold, based on a count of 15 trigrams in the input, is 2.861. The strings are declared to be the same, because the magnitude of the difference is just less than the threshold.

The trigram string comparator worked well enough to create author-title clusters that tolerated small errors, but did not include fields that a human cataloger would consider distinct. There were some problems, however, and it seems likely that it could be improved. Many similarity functions for string and document vectors are discussed in the information retrieval literature; Salton [33] offers a brief overview.

3.2.3 Performance of algorithm

The first round, which identifies a pool of potential matches, greatly reduces the number of comparisons that must be made to create author-title clusters. The brute force approach—compare each record to every other record–requires $O(n^2)$ comparisons:

$$\begin{aligned}(n-1)+(n-2) &+& (n-3)+\ldots+1 \\ &=& \frac{n^2-n}{2}\end{aligned}$$

The pool of potential matches tends to be very small; the average pool contained fewer than 30 records when the entire 240,000 record collection was processed. The savings over pairwise comparison is big—7.5 million comparisons instead of 31.2 billion.

The number of comparisons that must be performed grows with the average pool size, but it is difficult to judge how the collection size affects the average pool size. The pools are the results of a full-text search that includes one of the authors' last names (unless no authors are cited); although the number of occurrences of any word tends to grow linearly with collection size, names behave differently. The number of times "Szolovits" appears, for example, is a function of two things: how many papers Szolovits has written and how many duplicate citations for these papers are in the collection.

A closer examination of the potential pools that were produced while processing the full collection helps to map out the possibilities for collections up to hundreds of thousands of records, but does not provide a definitive answer.

I recorded the size of the potential match pools for the first 5,800 records, based on a collection with 50,000, 125,000, and 240,000 records. (Only a single pool was created, but records outside the first 50,000 were ignored for the first sample and records outside the first 125,000 were ignored for the second sample.) The distribution of pool sizes is roughly exponential. The majority of the pools contained fewer than 13 records in each sample. The tail of the exponential grows longer as the collection size increases; the largest pool increases from 14,294 for the 50,000-record sample to

45,750 for the 240,000-record sample.

Size of collection	Average pool size	Median pool size
50,000	15.76	8
125,000	18.30	10
240,000	32.29	13

Table 3.1: Statistics for size of potential match pools

Most of the pools are quite small, but occasionally large pools are returned; the latter are often caused by books with with some missing information, e.g. no author, or very common title words, e.g. "introduction to algorithms" or, worst of all, both. A reasonable optimization might be to ignore records that contain extremely large potential pools, and treat them as failures for which related records cannot be recognized.

3.3 Other systems for identifying related records

Several papers describe specific systems for identifying duplicates records. Two early duplicate detection projects are surveyed below—the IUCS scheme [47] and the original OCLC On-Line Union Catalog [18]; O'Neill, et. al [28] presents some characteristics of duplicate records in the OCLC catalog. Toney [42] describes a more recent effort, and provides a good overview of the design space for library duplicate detection systems.

The OCLC and IUCS projects were completed in the late 70s, when the library community had just begun sharing bibliographic records among institutions and building large online union catalogs. Subsequent work reflects the general approach mapped out in these two seminal studies.

The database merge/purge problem, described by Hernández and Stolfo [17], is very similar to the duplication detection problem. In the merge/purge problem, several different databases with similar records must be merged and the duplicates purged; an example is joining several mailing lists, which may contain inconsistent or incorrect addresses.

The recent information retrieval literature contains several reports from Stanford on duplicate detection in a Usenet awareness service [49] and several schemes for copy detection in a digital library [35, 36]. Duplicate detection in information retrieval is a substantially different problem, because the actual content of two documents is compared; the reported work discusses several different approaches to identifying overlapping content at the granularity of words and sentences.

3.3.1 Duplicate detection in library catalogs

The key difference between the OCLC and IUCS algorithms and my algorithm is that the former identify *duplicate* records. Records are duplicates when they describe exactly the same document and not just the same work. Although the precise rules for defining a duplicate vary somewhat between the algorithms, they both compare many fields. The OCLC rules require that the following fields match for a pair of records to be considered duplicates: author, title, publisher, copyright date, date of impression (if text differs), edition, medium, series, page or number of volumes, illustrator, and translator.

Most techniques for duplicate identification share the same gross structure: They use two rounds of comparisons, identifying possible duplicates on the first round and making more careful examination on the second round. The first round uses a fixed-length key, created from one or more field values; records with the same key are considered potential duplicates. A longer key, incorporating several fields, is used for the second round. (The second-round keys seem to be used primarily to avoid loading entire records into memory for comparison—a concern more important in 1979 than today.)

Duplicate detection keys summarize the information contained in the record; they consist of a series of fixed-length encodings of field values. In the OCLC and UICS schemes, different keys are used for round one and round two of duplicate identification. Fields are typically encoded in one of three ways:

1. Selecting certain characters from the field. The OCLC scheme uses characters 1, 2, 3, 5, 8, 13, 21, and 34 from a normalized title string to construct one of its keys. Character selection rules can be more complicated, specifying certain characters from specific words, such as the first and fourth characters of the second word.

2. For fields like reproduction code, which has a limited range of values, each possible value can be assigned a specific value.

3. More complicated encodings construct hashes of normalized field values. The second round of the OCLC scheme hashes the entire title into a 109-bit key. The key is built by converting the trigrams that do not span words to integers and applying a hash function to the sequence of integers. The IUCS scheme builds a Harrison key [14] from trigrams. The Hamming distance between two keys can be compared, allowing tolerance of typographical errors and other small variations between two strings.

The IUCS scheme uses a short, fixed-length title-date key for its first round, taking eight characters from the beginning and end of titles plus some date information. (MacLaury [27] described the choice of title characters.) Records that have the same title-date key are compared using three more keys: the first five characters of the author field, a 72-bit Harrison key of the title field, and the highest number taken from the various page fields.

The OCLC scheme is similar. The first round key looks for exact matches of four fields—the date, record type, reproduction code, and the 8-character title selection. The second round compares the entire key, but defines 16 different conditions under which the keys can be declared a match. These conditions examine different parts of the key, looking for either an exact match or partial match. Partial matches are defined for only a few fields of the key; a partial match of of Harrison keys occurs when the bits set in one key field are a subset of those set in the other key. One of the conditions for matching illustrates the approach: Two records are duplicates if there is an exact match of the Government Document Number and page fields, and partial matches of the LCCN, ISBN, edition, and series fields; partial matches occur when one record has no value in a particular field.

The QUALCAT expert system [31] is an interesting alternative to the key-based comparisons. A team of catalogers developed a set of rules that describe whether a certain combination of fields values makes it more or less likely that two records are duplicates. Each record pair is described by a possibility that it is a duplicate (*poss*) and the certainty it is a duplicate (*cert*). Initially, the *cert* is 0 and the *poss* is 100. Each rule increases the *cert* or lowers the *poss*. Once all possible rules have been applied, a pair of records with sufficiently high *cert* and *poss* are automatically labelled duplicates, records with low values are not, and records in between are referred to a human cataloger for review.

The expert system was used to compare records during the second round of QUALCAT testing. The first round was a fairly typical key-based filter using Universal Standard Bibliographic Code (USBC). The USBC is similar to the OCLC and IUCS first rounds keys, but Toney [42] notes that it is more dependent on clean data. Quantitative results of the QUALCAT system were not provided.

In the OCLC study, Hickey and Rypka [18] observe three major causes of failure to match duplicates. A difference in the first 34 characters of the title string that the initial date-title keys is drawn from causes 12 percent of the missed matches. The two other causes are differences in place of publication and pagination.

One question that remains to be answered is whether the probalistic first round proposed here offers any improvement over the key-based approach. Because both systems were not available for testing, we can at best hypothesize that the randomized text-search approach is more tolerant of errors. Word order errors, differences in cataloging, and typographical errors would all cause problems for the date-title key. The randomized text-search approach can tolerate these errors as long as one of the three queries uses words that do not contain typographical errors. It is not affected by word errors or by cataloging variations that affect which words are included in the title.

3.3.2 The database merge/purge problem

The approach to the merge/purge problem used by Hernandez and Stolfo [17] is similar to the one used for duplicate detection. Their algorithm use keys to partition their data into sets small enough that they are willing to apply computationally intensive comparisons. The source data varies widely, so any particular key is likely to miss many records; to overcome this problem, their algorithm makes several independent runs over the data with different keys and computes the transitive closure of the results. The accuracy of their test jumped from 70 percent with a single pass to 90 percent with three passes.

3.4 Analysis of author-title clusters

The algorithm for creating author-title clusters can fail in two ways. It can create clusters that include citations for more than one work (a false merge) or it can create two separate clusters that both contain citations for the same document (a missed match). The author-title cluster algorithm keeps these failures to a reasonable minimum; analysis of a sample set indicates that less than 1 percent of the clusters contained false merges and that missed matches occurred about 5 percent of the time.

The clusters were analyzed using a 937-record sample. The sample was selected so that each shares at least three title words with one or more other records; this characteristic was intended to increase the possibility of false merges. The clustering algorithm was run on the sample, and its results were examined by hand. The algorithm determined that the sample contained 554 unique clusters.

After the algorithm was run, I checked each cluster to verify that each record described the same document. I checked for missed matches looking for records in other clusters that contained similar title fields. I identified potential missed matches of a record by doing independent searches for each word in the title. (The search used a an approximate string searching program that did not use a n-gram approach.) If a different record was returned by at least two-thirds of the agrep searches, it was compared by hand with the source record.

According to my manual check of the records, the sample actually contained 554 unique works, 250 of which were cited by more than two records. The author-title clusters contained four false merges and 30 missed matches.

Clusters identified	554
Missed matches	30
False merges	4

Figure 3-2: Errors in author-title clusters for 937-record sample

The four false merges all involved closely related bibliographic records. In three of the four clusters, the two different works were part of a series, where each paper was labelled part one or part two. Figure 3-2 shows an example. The papers had

```
@Article{Mulmuley90,
title =            "A Fast Planar Partition Algorithm, I",
author =           "Mulmuley",
year =             "1990",
journal =          "Journal of Symbolic Computation",
volume =           "10", }

@Article{Mulmul91,
title =            "A Fast Planar Partition Algorithm, II",
author =           "K. Mulmuley",
year =             "1991",
month =            "[1]",
pages =            "74–103",
journal =          "Journal of the ACM, JACM",
volume =           "38",
number =           "1", }
```

Figure 3-3: Two falsely matched records

the same authors and the approximate string match reported that the titles were the same because they differed only in the final trigram. The final false merge involved two different papers with very similar titles—"Towards Dataflow Analysis of Communicating Finite State Machines" and "Dataflow Analysis of Communicating Finite State Machines"—and the same authors.

The sample set is small enough that it is difficult to extrapolate the results to a large collection with great precision, but it appears that the failure rate would be acceptably low. Measured in percentages, false merges occurred in 0.7 percent of the clusters and 5.7 percent of the clusters were missed matches that should have been merged with another cluster.

I also used the results of two other algorithms to gauge the relative success of the author-title clustering algorithm presented here. The bibmerge program [1] creates title-date clusters; if two records have the same title and date, they are placed in the same cluster. Bibmerge has only limited utility as a benchmark, because it uses a very simple detection scheme that is not tolerant of formatting and typographical errors. The program's source contains the comment: "This script is written in the most unprofessional manner available to me, the only reason why I have not scrapped it is that is works." Nonetheless, it was the only other duplicate detection system available for testing on the same sample set.

Bibmerge found 650 unique clusters; it missed 126 matches and made one false merge. The title-data clusters identified 10 record pairs that were missed by the author-title clusters, because the title and date field was the same but the author was different. In seven cases, the author fields were improperly formatted. In two cases, the authors were listed in a different order in each record. The final case was bibmerge's lone false match: An issue of *Computing Surveys* contained two responses to an earlier article, both of which were titled "The File Assignment Problem."

A second check of the quality of the algorithm presented here is the results reported for the OCLC duplicate detection scheme described earlier, although the OCLC clusters were created under substanitally different rules. The analysis of the OCLC algorithm is the most substantial reported in the literature. Using a set of 184 pairs of records that had been identified in advance as duplicates, the OCLC system successfully identified 127 pairs, a 69 percent success rate, compared to a 94 percent success rate for my author-title algorithm The false match rate was measured by selecting 1,000 record pairs and applying the duplicate test to each pair; only some of the pairs were duplicates. The error rate on this sample was 1.3 percent—13 false matches—which is comparable to the 4 false merges out of 554 (0.7 percent) in the author-title clusters.

Chapter 4

Merging Related Records

This chapter describes the creation of a composite record that summarizes the duplicate records in an author-title cluster. It introduces two terms, information dossier and union record, to describe two different ways of grouping related bibliographic information.

A union record is a composite record created by merging several bibliographic records from distinct sources. A different union record is created for each different type of document included in the cluster; thus, a particular cluster may have union records for a journal article, a technical report, and a conference paper.

An information dossier [8] is a collection of information objects, e.g. bibliographic records, related to some way to one another. Specifically, I use the term to describe the source records that form an author-title cluster and the union records generated for the cluster. Although the current system does not include other objects, the next section presents a system for automatically linking records to electronic copies available on the Internet. The dossier would contain links or perhaps local copies of relevant item; in this way, a dossier is distinct from the records in a cluster.

Union records can be an imprecise summary of the source records, because the quality of the source records is variable and because there are sometimes too few records to be able to resolve conflicts between records. When there are conflicts, a single representative value is chosen for the union record instead of omitting the field or creating a hybrid value.

Because of this decision, I also calculate statistics that describe the quality of the composite. When the source records vary significantly from the union record, the user may wish to examine the source records. These statistics are described in Chapter 5.

4.1 Goals of merger and outline of process

There are three primary goals for the merger process that creates union records and dossiers:

- to eliminate redundancy in query results
- to identify multiple access paths to a work
- to create complete and accurate union records from conflicting and incomplete source records

Because some author-title clusters may also include false merges, the dossier should contain some information about how closely the union record matches each source record.

Duplicate listings in the results of a query limit the ease with which the results can be used. A long list takes longer to transmit, process, and read than a short list, but the real difficulty is that it can be difficult for a person to identify the duplicate and related records. For a person, identifying duplicates in a long list can be quite difficult and error prone.

When the union record is created, there is an opportunity to eliminate errors and to create a record that is more complete than any of the sources. An error in one or a few source records can be purged, if there are enough records with the correct information; the correct value for the field is chosen by voting. Fields that are missing in one record can be drawn from another record. Unfortunately, these kinds of quality control have limited utility, because in the common case there are only a few records in a cluster.

Finally, the dossier brings together information about each different instance of publication, which allows a user to choose the physical document that is easiest to retrieve. When works are published several times—in different journals, proceedings, or books—having an exhaustive list of these publications makes it easier to find the work when the local library does not hold all of the publications. Because many universities publish their technical reports in digital form, knowing that a work was issued as a technical report means the work is more likely to be found online.

A secondary goal for the dossier is to minimize the amount information lost to the user when false merges occur. The dossier should include information about how closely individual records match the union record and how much variation there is in the value of a field across the source records. When a particular record differs substantially from the union record or when one field has a different value in each record, there may be cause for the user to suspect a false merge. The interface described in the next chapter provides access to all of the source records and a measure of the truthfulness of the union record.

The general strategy for merging records is based on the different kinds of entry types allowed by Bibtex. The source records are grouped by entry and a union record is produced for each type. The fields values in the union record are assigned, for most fields, by counting the occurrences of each value and choosing the most commonly occurring value for the union record. Some fields are treated differently. The author and title fields will be the same regardless of type, so we can apply to counting strategy can be applied globally. The author field is also different because the counting strategy is applied to the component names rather than the full author list.

4.2 Creating the union record

The merger process described here is very simple, and only copes with a few kinds of errors in the source records. However, a number of refinements are suggested for dealing with a wider range of errors; these refinements deal with specific kinds of errors but use the same basic strategy. (A few of these refinements have been implemented, but most have not.)

Records are merged a field at a time. For each field, the number of occurrences of each different value is counted and the most frequently occurring value is chosen. Sometimes, one or more values will occur with equal frequency, and a tie-breaker is needed; the longest value is chosen. The tie-breaker is arbitrary, although it is hoped that the longer values will be more likely to contain information that helps the user.

There are three exceptions to the rules for merging fields.

- Values for the month field are converted to the first three letters of the month, or they are ignored.

- Values other than a four-digit number are ignored in the year field.

- The author field is merged by considering each name individually, finding the longest form of that name, and assembling a list of these longest names.

There are three rough categories of fields, each of which will be affected somewhat differently by merging. The categories differ in how likely a field is to be the same in two different records for the same document.

The first category includes fields like title, date, or pages, which describe fixed, objective characteristics of the document. These fields are most common and the merger process is tailored to them.

The second category includes fields like note, annote, and keywords, which will vary widely from source record to source record (if they appear at all). There are no specific guidelines for the use of these fields, so each source may describe some different characteristic of the document. Each occurrence of one of these fields in a source record is included in the union record.

The abstract field is hard to classify. Although there should be a single abstract for each document, there is a lot of variation in what is actually recorded as the abstract. Currently, the longest abstract is chosen using the standard process.

The last category is fields which are used to manage bibliographic records or serve some other purpose specific to the record's creator. Standard fields like key and many non-standard fields, like bibdate or location, will appear in the union record, chosen by the standard counting scheme. However, it is unlikely that any of the field values are related and the value selected has little significance. The interface presented in the next chapter ignores these fields in the standard display.

4.2.1 Problems with union records

The counting approach does not work very well when there are only a few records of a particular type. Typographic and formatting errors also cause problems.

In the six records for the article "Scheduler Activations: Effective Kernel Support for the User-Level Management of Parallelism" by Anderson et al., three different values appear in the pages field—"53" and "53--70" each appear once and "53--79" appears four times. If the value "53--79" appeared only once, it would be impossible to distinguish between the correct value and the incorrect ones.

Another potential problem is the policy of creating a union record for each different

BIB-VERSION::	v2.0
ID::	MIT-LCS//MIT/LCS/TR-569
ENTRY::	February 25, 1995
ORGANIZATION::	Massachusetts Institute of Technology, Laboratory for Computer Science
TITLE::	Concurrent Garbage Collection of Persistent Heaps
TYPE::	Technical Report
AUTHOR::	Nettles, S.
AUTHOR::	O'Toole, J.
AUTHOR::	Gifford, D.
DATE::	June 1993
PAGES::	22
ABSTRACT::	We describe the first concurrent compacting garbage collector for a persistent heap. Client threads read and write the heap in primary memory [...]

BIB-VERSION::	CS-TR-V2.0
ID::	CMU//CS-93-137
ENTRY::	September 13, 1995
ORGANIZATION::	Carnegie Mellon University, School of Computer Science
TITLE::	Concurrent Garbage Collection of Persistent Heaps
AUTHOR::	Nettles, Scott
AUTHOR::	O'Toole, James
AUTHOR::	Gifford, David
DATE::	April 1993
PAGES::	22
ABSTRACT::	We describe the first concurrent compacting garbage collector for a persistent heap. Client threads read and write the heap in primary memory [...]

Figure 4-1: CS-TR records for one TR from two publishers

document type in a cluster. The policy assumes that there will be only a single document of a particular type in a cluster, i.e. that we will not find two different articles with the same author and publisher. This assumption does not hold in some circumstances, resulting in misleading union records.

One example of a failure is a technical report written by authors from different institutions and issued independently by each institution. (See Figure 4-1.) The system will create a single union record for these reports, which correctly represent most information—author, title, abstract—but will obscure or confuse the issuing organizations and the report's number or identifier. The problem is serious, because a the organization and report number are important for locating a copy of the document.

```
@Article{BNBTEAEDLHML89,
title =      "Lightweight Remote Procedure Call",
author =     "Brian N. Bershad and Thomas E. Anderson and Edward D.
             Lazowska and Henry M. Levy",
year =       "1989",
month =      dec,
pages =      "102–113",
journal =    "Proc. Twelfth ACM Symposium on Operating Systems",
volume =     "23",
number =     "5", }

@Article{BershadAndersonLazowskaLevy90,
title =      "Lightweight Remote Procedure Call",
author =     "Brian N. Bershad and Thomas E. Anderson and Edward D.
             Lazowska and Henry M. Levy",
year =       "1990",
month =      feb,
pages =      "37–55",
journal =    "ACM Transactions on Computer Systems",
volume =     "8",
number =     "1", }
```

Figure 4-2: Bibtex records exhibiting the conference-journal problem

Another example of a failure is caused by confusion about how to catalog the papers in a conference proceedings that are published as a journal article, e.g. the SOSP proceedings printed in *Operating Systems Review*. The proceedings *is* an issue of the journal, so it would be quite reasonable to catalog the conference paper as a journal article. But if a paper is cataloged as an article and is also published in a journal (say the *Transactions of Computer Systems*), then the record for the SOSP paper and the record for the TOCS article will be merged into a single union record.

4.2.2 Refinements to merger process

The general strategy just described improves significantly with a few refinements. Three refinements have been implemented: The author field is treated separately, as it was during cluster identification, because of its special formatting. Two simple filters are used to prevent field values with detectable errors from being counted. Several other refinements are suggested, but have not been implemented.

The author list is constructed differently because all of the author fields in the source records must match (with the approximate match described in Section 3.2.1) for the records to be placed in the same cluster. The merge algorithm extracts as much information about each name as possible and creates a new author list. When names are compared, each part (i.e. first, middle, last name) is expanded wherever possible; a blank entry becomes an initial or a full name, and an initial becomes a full name.

Filters, which validate field values before creating the union record, prevent invalid data from being included in the union record; they can also normalize field values, by testing for common mistakes and cataloging variants and attempting to correct them.

The month and year fields are merged using filters. Common problems in these fields include:

- The source record combines them, e.g. year = "Sept. 1987"
- The source record uses the number of the month instead of the name, e.g. month = "[2]"
- The source record uses question marks to indicate uncertainty, e.g. year = "199?"

The month filter normalizes all entries to the three-letter abbreviations used by Bibtex. Numbers are converted to text and full names shortened; entries that cannot be re-formatted are ignored. The year filter accepts only four-digit years, removes any data other than the year, and discards entries for which a valid year cannot be found.

More powerful heuristics for identifying mis-formatted and incorrect data and either discarding it or converting it to the correct format would further improve the quality of the union records. For example, it may be profitable to identify a group of field values that are similar but not exactly the same, e.g. two titles that differ in only a few positions or years that are similar, like "199?" and "1991. The merge process would then determine the most frequently occurring group, and then choose a representative element from that group. If three source records contained the year values "1989", "1991", and "199?", this strategy would choose 1991 as the most frequently occurring, correctly-formatted value.

When there are several fields that occur with equal frequency, we choose the longest value for the union record. There are many other heuristics that could be used instead, like a strictly random choice or choosing the shortest field; heuristics could be applied on a per-field basis, e.g. using the highest number in the year field.

4.3 Clusters sizes and composition in DIFWICS

A brief analysis of the author-title clusters in the DIFWICS suggests two broad observations. First, enough related records were found to justify the effort involved in identifying them. Second, within a cluster there is substantial variation among the source field values.

The DIFWICS collection consists of 243,000 source records and 162,000 author-title clusters. More than half of the records belong to a cluster that contains two or more records. Fewer clusters contain more than one record of the same type—about 30,000 clusters or 20 percent of the collection.

Cluster size	Number of clusters	Number of records	Percentage of all records
1	116,829	116,829	45.8%
2	28,865	57,730	22.6%
3	9,068	27,204	10.7%
4	3,885	15,540	6.1%
5	1,710	8,550	3.4%
6	878	5,268	2.1%
7	528	3,696	0.9%
8 or more	772	7,888	3.1%
total	162,535	242,705	100.0%

Table 4.1: Cluster sizes

Table 4-1 shows how many clusters of a particular size there are and what percentage of the total number of records are in clusters of that size. The average number of clusters in a record is 1.49.

The distribution of record types within the entire collection is basically the same as the distribution within clusters: Articles are most common, followed by papers in conference proceedings Table 4-2 shows how many source records of are particular type exist. Most of the clusters contain one type of record.

Clusters with more than one type of source record represent less than 10 percent

Type	Records	Percent
Article	108,985	(44.9%)
InProceedings	75,150	(31.0%)
TechReport	23,606	(9.7%)
Book	12,178	(5.0%)
InCollection	9,955	(4.1%)
PhdThesis	3,143	(1.3%)
other	9,688	(4.0%)
total	242,705	(100.0%)

Table 4.2: Souce records, by type

of the total number of clusters. The three most common combinations are Article and InProceedings (4,349 clusters), Article and TechReport (1,808 clusters), and InProceedings and TechReport (1,536 clusters). Fewer than 1,000 clusters contain three different record types and none contain four or more.

Within clusters that contained two or more records of the same type, I examined individual fields to see how often all the source records had the same value. Field values were compared by normalizing them to lowercase alphanumeric strings, eliminating formatting and punctuation.

The statistics were gathered using a 10 percent sample of the clusters, considering only those clusters that had two or more records of the same type. The sample included 3,753 Article clusters (11,836 records), 3,113 InProceedings clusters (9,705 records), and 1,335 TechReport clusters (3,345 records).

Table 4-3 summarizes the results of the analysis on several standard Bibtex fields. It shows the number of times the field appeared in the sample clusters and the number of times all the records in a cluster had the same normalized value.

The variation in the title field is interesting because it suggests how many more clusters would exist if approximate string matching wasn't used. About 10 or 15 percent of the titles don't contain the same normalized string, even though they are considered to be the same under the approximate string match.

Field	Article		InProceedings		TechReport	
	Present	Uniform	Present	Uniform	Present	Uniform
title	3753	3197 (85%)	3113	2665 (86%)	1355	1232 (91%)
year	3753	3546 (94%)	3075	2850 (93%)	1352	1212 (90%)
month	3435	1417 (41%)	2496	1636 (66%)	1031	900 (87%)
pages	3683	2640 (72%)	2973	2177 (73%)	417	393 (94%)
journal	3752	1193 (32%)				
institution					1350	763 (57%)
number	3578	3458 (97%)			995	585 (59%)
booktitle			3062	477 (16%)		
publisher			1769	1265 (72%)		

Figure 4-3: Variation in fields values within author-title clusters of the same type

Among the other fields, the year is the most consistent; records have the same year more than 90 percent of the time. The month field shows much more variation because there are several different abbreviations used for each month. Abbreviations cause similar problems in several other fields with high variation—journal, institution, booktitle, and publisher. The filters for improving the merger process, described above, could increase the number of fields with uniform values.

Chapter 5

Presenting Relations and Clusters

This chapter describes some preliminary work on presenting a collection of 240,000 records after author-title clusters are identified and information dossiers constructed. The Web-based interface allows simple full-text queries of the collection, presents related records together on the screen, and automatically creates links that invoke searches of Web-indices, technical report archives, and the local library collection.

Although the user interface is only a prototype, it does illustrate two important design principles.

1. The results of a search should display the works that match the query, presenting the union records for different versions of the same work together. The presentation should cull the most complete and accurate information available from any duplicate records for a document, but should not hide variations in the underlying records.

2. The related records in a particular cluster should be used to help the user find an easily accessible copy of a document. The combination of bibliographic information, which allows well-defined queries into Web indices, library catalogs, and other databases, and clusters, which link related documents, should increase

a user's ability to find a particular document (or a related one that represents the same abstract work).

5.1 The basic Web interface

The basic interface to the collection is a full-text index of all 240,000 source records, that allows basic Boolean queries. (The underlying search engine allows for queries that look for words only when they appear in certain fields; a Web interface for this feature is underway.) The records are stored in a database that maintains a unique identifier for each record, a separate name space for clusters, and a mapping from record id to cluster id. The index uses record ids internally, but returns a list of clusters in response to a user query.

A query using the Web interface returns a summary showing the title, author, and year of all the matching works, followed by a expanded entry for each cluster. Figure 5-1 shows the expanded entry for the paper "Lightweight Remote Procedure Call."

The first two lines of the cluster entry show the title and authors— the two defining characteristics of the document—followed by entries for the abstract and keywords, which will be the same for each document in the cluster.

A bulleted list of three document citations follows the keywords; the citations are produced from the union records. The first two lines of document citations are formatted to look like a citation for a traditional bibliography. In the first entry, an article in *ACM Transactions on Computer Systems*, it shows the date the article was published, the volume and number, and the pagination.

The third line of the document citation contains a hypertext link to a search of the local reading room catalog. The search will look for the particular document being cited. In the first entry, it searches for the journal the article appeared in; for the second entry, a conference paper, it searches for the proceedings.

The last line of the document citation contains two links to more information about the records for the document. The link to the expanded record displays a full union record, which includes nonstandard fields that are not displayed in the regular citation. The second link returns the source records used to make the union record; the link indiciates the number of source records and returns them in their original form.

The last line of the cluster entry (following the bulleted document citations) incorporates links to several other search services. There are links to the Alta Vista and Excite Web indices, which contain queries based on the title field and the authors' last names. When the work has been published as a technical report, links to the NCSTRL and UCSTRI technical report indices are included.

5.2 Assessing the quality of union records

The most serious shortcoming of the current interface is that provides no warning to the user when there are differences between the union record and the source records. Preliminary work for detecting and measuring these differences is presented here, but the results are not integrated into the current interface.

Because the displayed document citations are based on the union records created by merging duplicate records, there is a chance that the citation will contain mistakes or hide useful information that is contained in the source records. Problems could be the result of a record that describes a different document being included in the cluster by mistake. However, most problems arise because the source records contain different and conflicting information about the document. (These problems were discussed in greater detail in Chapter 4.)

When a record is included in an author-title cluster by mistake, two different kinds

Lightweight Remote Procedure Call
Brian N. Bershad, Thomas E. Anderson, Edward D. Lazowska, and Henry M. Levy

Abstract. Lightweight Remote Procedure Call (LRPC) is a communication facility designed and optimized for communication between protection domains on the same machine. In contemporary small−kernel operating systems, existing RPC systems incur an unnecessarily high cost when used for the type of communication that predominates −−− between protection domains on the same machine. This cost leads system designers to coalesce weakly−related subsystems into the same protection domain, trading safety for performance. By reducing the overhead of same−machine communication, LRPC encourages both safety and performance. LRPC combines the control transfer and communication model of capability systems with the programming semantics and large−grained protection model of RPC. LRPC achieves a factor of three performance improvement over more traditional approaches based on independent threads exchanging messages, reducing the cost of same−machine communication to nearly the lower bound imposed by conventional hardware. LRPC has been integrated into the Taos operating system of the DEC SRC Firefly multiprocessor workstation.

Keywords. RPC−Modellierung BERS 89,HUTC 89

- Article in ACM Transactions on Computer Systems.
  Feb 1990. vol. 8, no. 1. pages 37−55.
  Check in LCS/AI reading room.
  View expanded record, source records (9).

- Appeared in sosp12.
  Dec 1989. pages 102−113.
  Check in LCS/AI reading room.
  View expanded record, source records (7).

- Technical report. Department of Computer Science, University of Washington, no. 89−04−02.
  Apr 1989.
  Check in LCS/AI reading room.
  View expanded record, source records (3).

Search for this paper in Alta Vista, in Excite netSearch, in NCSTRL, in UCSTRI.

Figure 5-1: Sample author-title cluster display from Web interface of failure result.

1. In a cluster that contains several citations for a document type, the mis-matched record will be hidden. The union record will not describe the hidden record, and any query that matches the record will return the cluster it is mistakenly included in. The only way to discover the hidden record is to look at the source records.

2. In a cluster with only a few citations, the merger process may create a union record that includes some fields from the mis-matched record and some fields from the good records, because an arbitrary value is selected when a most frequently occuring field value can't be identified. The result is a confusing union record that mixes information from different documents or works.

The "source match" ratio can identify the first problem. It measures how closely the fields in the union record match the fields in a particular source record. It is computed by comparing the standard fields in the union record with the same fields in a source record. (Records not defined in the source record are ignored.) The value is a tuple of the number of fields that match and the number of field that do not match, e.g. (4,3) indicates four fields match and three fields do not.

The second problem can be identified with the "field consensus" ratio, which measures how many of the source records contain the same value as the union record for a particular field.

For each union record, we can compute the source match value for each source record and the field consensus value for each of the major fields. The result in each case is a list of ratios, one source match ratio for each source record and one field consensus ratio for each standard field.

Unfortunately, these ratios provide only a very rough measure of the differences between records or fields. The variations in Bibtex records that make identifying duplicate and related records hard also complicates the analysis of union records. Variations, like abbreviations and typos, cause fields to fail to match. To limit the effects of formatting errors, the fields values are normalized before comparison; all non-alphanumber characters are eliminated and all letters are converted to lowercase.

These problems make the ratios hard to interpret. A source match ratio of (2,5) could mean that the source describes a different document than does the union record. But it could just as well mean that the source record contains a number of non-standard abbreviations or typographical errors.

Despite the problems with creating and intepreting these ratios, some informal tests suggest that they are helpful for identifying clusters that contain false matches. Figure 5-2 shows an a sample cluster where the statistics help to identify a false match. The cluster contains four technical report citations, which the union record reports as "Process Migration in the Sprite Operating System." In fact, the cluster also contains a single citation for a different technical report by the same author, titled "Transparent Process Migration in the Sprite Operating System." The problem is clear in the source match ratio, which shows that the third source record shares one common field value with the union record and differs on five other fields. The single differences that appear in the field consensus ratios also suggest that one of the source records may not belong.

In the previous example, the difference between source record and union record was rather pronounced. Often, the statistics are more ambiguous. Figure 5-3 shows a more representative set of ratios. Although the first source record matches on two fields and disagrees on five fields, the cluster is correct.

If the ratios are useful, it seems less clear how to use them to automatically detect problems and warn the user. Always presenting the statistics to the user also seems cumbersome, because they further complicate the display and because they complicate understanding the display. A middle ground that displays the statistics

Fredrick Douglis. *Process Migration in the Sprite Operating System. Technical report.* Computer Science Division, University of California, no. UCB/CSD/ 87/343. Feb 1987.

Source match ratios

record	source match
#1	[5,1]
#2	[3,4]
#3	[1,5]
#4	[5,1]

Field consensus ratios

title	[3,1]	institution	[2,2]
month	[3,0]	pages	[1,1]
number	[2,2]	year	[3,1]

Figure 5-2: Source match and field consensus ratios for cluster including a false match

Michael L. Scott and others. *Implementation Issues for the Psyche Multiprocessor Operating System.* Appeared in *Proceedings of the Symposium on Experiences with Distributed and Multiprocessor Systems.* Oct 1989. pages 227-236.

Source match ratios

record	source match
#1	[2,5]
#2	[5,1]
#3	[4,3]

Field consensus ratios

title	[2,1]	booktitle	[1,2]
month	[1,1]	pages	[3,0]
year	[3,0]	publisher	[1,2]

Figure 5-3: Source match and field consenus ratios for correct cluster

when they show significant variation and supresses them when there is very little variation might strike the right balance. A warning indicator that showed one of three values—a mistake is likely, possible, or unlikely—would also display the information concisely.

Chapter 6

Automatic linking

The previous chapter described the current Web interface, which includes links to Web search services in its display of author-title clusters. The search links are a first step towards automatically linking records in DIFWICS directly to copies of documents on the Web.

The general scheme for automatically linking a cluster to copies of the work available on-line is the same as the scheme for identifying the clusters in the bibliographic collection: A full-text search using some arbitrarily selected words from the author and title field will turn up potential copies and a more detailed comparison of those copies will find actual instances of the work.

6.1 Searching for related Web citations

The work reported on here does not perform automatic linking, but it makes a first step in that direction and the preliminary results offer some insight on a full-scale automatic linking project. The basic insight is that any document that is accessible from the World-Wide Web has a hypertext link to it – either from another page or through some search system; the hypertext link is a citation for the document, and often the hypertext link is included from a traditional citation that describes the document.

The Web catalog interface's links to other search services – to the reading room catalog, the AltaVista and Excite Web indices, and the NCSTRL and UCSTRI technical report indices – provide the first-step full-text search for online documents. The current system requires that the user perform the second filter manually on the search results, but the process can be automated.

The individual search links are created with some knowledge of the particular search interface and the format of citations on the Web. The AltaVista search engine makes efficient use of long quoted strings, so the search looks for occurrences of the full title. The reading room catalog interface does not catalog journal articles, but does catalog journals and uses the word "holdings" in each journal entry; for journal articles, the catalog search uses a few words from the journal name and the word holdings to check the journal's availability.

A pair of examples illustrate some of the success and pitfalls of this approach to automatic linking. The first example is a search for the paper "Obliq: A Language with Distributed Scope" by Luca Cardelli. The paper was issued as a DEC SRC technical report, and is available from the author's personal Web pages. The results of a search for this paper are unusually good, because SRC's technical report archive includes a seperate page for each technical report, which matches the queries very closely.

Searches in AltaVista, Excite, UCSTRI, and the reading room catalog all return a link to the report in the SRC technical report archive as the best match for the search. (NCSTRL does not index SRC technical reports.) The two Web searches use relevance ranking to order all the Web pages that matched at least some of the query terms. Among the most relevant pages are:

- several pages about use of the Obliq language

- a Web page listing the contents of the issue of Computing Systems in which the paper was published
- a bibliography of Cardelli's papers from a Web site in Germany
- lecture notes for a class that discusses Cardelli's work on the semantic of multiple inheritance
- several papers that cite the Obliq paper
- an FAQ on object-oriented programming languages

The Web searches returned Cardelli's personal page with a link to a Postscript copy of the paper, but it is not ranked highly in the list of search results. It appears in about 30th position.

A search for the paper "A Theory of Primitive Objects: Second-Order Systems" by Martin Abadi and Cardelli produces more representative results, because it is not a SRC technical report. The search illustrates the benefits of the AltaVista full-string search over the Excite keyword-only search. The Excite search locates many pages that contain mention of the authors and their work on type theory, but no pages with links to the desired paper. The AltaVista search, on the other hand, locates two pages maintained by the authors with links to the paper. These pages appear to be several levels deep on their local filesystem, so it is likely they are not included in the smaller Excite index. (The reading room link shows that it has a copy of the conference proceedings that include the paper.)

6.2 Principles for fully-automated system

These basic results suggest several things about how to design a system for automatically tracking down citations on other Web pages, instead of requiring the user to take the second step of examining individual Web pages.

1. Large-scale Web indexes are likely to index many pages that contain references to the paper being sought. These references will be a mix of normal citations, found in other papers, lecture notes, and other works, and of Web-based citations, like those found in personal publication lists. Some but not all of these citations will contain hypertext links to a digital copy of the document.

2. The citations that included hypertext links are often found on Web pages several levels deeper than a server's main Web page. Many of the smaller Web indexes omit these pages, so searches of these indexes are more likely to return no relevant pages or pages that lead to the revelant citation but do not actually contain it. For example, the second Excite search described above returned a page about a book on objects written by Abadi and Cardelli, which in turn contained links to the author's personal pages, which contained links to lists of publications.

3. Very few papers are available on the Web in an easily indexed format like HTML or plain text. Most papers are available as Postscript, which is not easily indexed. As a result, it is uncommon to discover pages where the title or search summary clearly indicates that the page contains the sought-after document.

We experimented with two services that augmented the current interface for searching the Web. One service performed searches automatically, retrieved the first 10 pages returned, and searched the pages for citations that were similar to the document being sought. Another service interposed a Web proxy server between the user that tracked the user's examination of the search results and recorded what page the paper was actually found on. The proxy let the user navigate the Web as normal, but added a header to the top of each page that showed the author and title of the paper being sought; the proxy header also contained a link for the user to follow when the paper had been found. Neither of the experimental services were robust enough or successful enough to include in the current interface, but our brief use of them suggests that they would be interesting areas for future work.

Chapter 7

Conclusions and Future Directions

7.1 Future directions

7.1.1 Performance, portability, and production

The underlying database and environment for storing and comparing bibliographic records, in particular the n-gram string comparison, were not the primary focus of this thesis. If the system is going to support a very large collection (millions of records) or many simultaneous users, its performance needs to be improved. The system also needs a few other implementation changes to allow long-term use in a production environment.

The prototype system was implemented primarily in Perl 5, which allowed rapid prototyping at the cost of execution-time effficiency. The n-gram string comparison is particularly slow in the current implementation; during the second round of cluster creation, loading records from disk and performing the detailed comparison proceeds at less than 10 records per second (on a 25 MHz RS/6000).

One consequence of the implementation decisions is that it is difficult to identify the bottlenecks. The Perl implementation of, for example, n-gram comparisons is clearly slow, but implementing it in a different language might speed it up enough that some other part of the system becomes the bottleneck. The remaining comments on performance should be considered with this constraint in mind.

It appears that loading bibliographic records into the system is costly. The records are stored in their original text format, and parsed each time the record is loaded. The primary cost of loading a record is parsing the Bibtex, so it may be profitable to develop a more easily parse intermediate format.

The cost of performing full-text queries during the construction of potential match pools appears to be the next mostly costly part of clustering, after the n-gram comparisons. In addition to optimizing the internal workings of the search engine, there may be an opportunity for global query optimizations. Three three-word queries are created for each record in the collection; it seems probable that the same query would be generated more than once, both because of duplicate records and because authors are likely to generate different works with some of the same title words. If performing a query is a signficant bottleneck, the queries could be re-ordered to take advantage of repeated queries.

One important limitation, independent of performance considerations, is that it does not record the source of a bibliographic record. When a particular bibliography is integrated into the main collection, there is no way to record that it was originaly part of, say, the USENIX bibliography. As a result, it is difficult to keep the collection up-to-date and incorporate changes and additions from a bibliography that has already been included.

The production system should record the source in a consistent way, so that the main collection can continuously incorporate changes from external sources. A "data pump" could be set up to monitor sources of bibliographic records and add new records or update modified records.

Recording the source of a record enables other value-added services, such as judging the quality of a record based on its source, which are described below.

7.1.2 Improving the quality of records

One approach to improving the quality of bibliographic information, the one described in this thesis, is to locate related bibliographic records and merge them in a way that improves the quality of information. A different approach is to use authority control.

In a library catalog, authority control describes the process of identifying each of the unique names in the catalog—usually names of authors and names of subject headings—and finding all of the variant forms of the name within the catalog. An authority record describes the authoritative form of the name along with any variants.

Authority records can be integrated into the catalog, but more often they are used by librarians to help in the preparation of the catalog. New entries in the catalog can be checked against the authority records to determine the proper form of the name, and old records can be updated to use the authoritative form.

Using authority control to regularize the use of certain fields, notably author, journal, and publisher, would improve the quality of the records visible to the user and, in the case of the author field, the quality of the clustering algorithm. (Recall that problems parsing and comparing author lists accounted for most of the missed matches during clustering.) A system for authority control, however, would have to deal with some of the same problems the clustering algorithm handles now; it needs to identify as many variant entries as possible without being so aggressive that truly different entries are conflated.

Authority control for journals, publishers, and conferences would not affect the creation of author-title clusters, but would make it easier to produce union records and would improve the value of the "field consensus" and "source match" ratios. Creating the authority records, however, would be a labor-intensive process, requiring a human cataloger to generate a list of authoritative names and review possible variations to determine if they in fact refer to the same object. It should be possible to automate much of the process by looking for plausible variations on and abbreviations of the authoritative name, but some variations would be virtually impossible: The *Journal of Library Automation*, for example, changed its name to *Library Resources and Technical Services*. On the other hand, it is possible that journals in different fields could be abbreviated the same way; possible conflicts in abbreviations should be reviewed by a human cataloger.

This observation about the need for human supervision of authority control applies to library cataloging in general. The identification of basic bibliographic information—the author and title of the work, the pages it appears on, etc.—is a largely clerical process. (Fully automated cataloging is an active area for research, but little progress has been made [44].) Instead, cataloging should focus information that is more difficult to obtain—whether two authors with similar names are in fact the same person or whether two papers with similar but different titles actually represent the same work. Heaney makes the same case in his argument for an object-oriented cataloging standard[15].

7.1.3 Identifying other bibliographic relationships

The clustering algorithm identifies author-title clusters, in part because identifying equivalence and derivative bibliographic relationships has the most advantage for users and in part because they can be reliably identified in the presence of mixed-quality records. Identifying other bibliographic relationships would also be useful; if authority control (or some other mechanism) is used to improve the quality and consistency of field values, this problem would be easier to tackle.

The hierarchical relationship holds between a composite work and its parts—

between a journal issue and the articles it containes or between a conference proceedings and the papers it contains. The wide variation in the journal and booktitle fields makes this relationship hard to identify in the current collection, but authority control could make possible comparisons. The relationship could be stored as journal issue clusters or proceedings clusters that contain all of the articles from a particular issue of a journal or all the papers presented at a conference.

The hierarchical relationship would be a useful addition to the current search interface. When a user finds an interesting paper, he could examine the proceedings clusters to see if any similar work was presented or check the journal cluster for an accompanying article. Clusters could also identify the sequential relationship by linking together the journal issue or proceedings clusters, which would provide a three-level hierarchy from browsing; users could move between clusters for individual articles, clusters for issues, and clusters for entire journals.

The referential relationship is interesting because it cannot, in general, be identified using bibliographic information alone. References that involve critique or review, e.g. a *Computing Reviews* article, might be identifiable, but citations do not contain enough information to determine that one paper is cited by another paper.

7.1.4 Integrating non-bibliographic information

The referential relationship could be identified if an information dossier contained information in addition to bibliographic records—in particular, if it contained the citation list or the entire text of the document. The information dossier is a particularly useful notion because it can include information of all sorts, such as the full text of the document or information on how to order it.

Non-bibliographic information can be included in a dossier if the author and title can be identified and matched with an existing author-title cluster. Extending the dossier allows a much richer set of interactions with the library collection. For example, a user browses a document and discovers a reference to another work that is potentially of interest. The user highlights the reference with his or her mouse and clicks a button, and the document that was referenced appears in a new window on the user's screen.

Including the full-text of a document (including the citations) enables many other applications as well. Users can perform queries across the entire text of a document, which creates more opportunities for discovery relavant documents. Abstracts and summaries can be automatically generated for the documents [32], which can help the user quickly establish the relevance of a document to the current search. Citation indexes and graphs can be created that show how often and how widely a particular paper or conference proceedings is cited.

One related issue that does not seem to be well-understood is machine processing of citations intended to be read by humans. It is difficult to design a general purpose processor that can identify the distinct parts of a citation; possible problems include identifying the individual authors names and distinguishing between different numeric values, like years, page numbers, and volume/issue numbers. Some leverage on the problem can be gained by looking for bibliographic records that are "similar" to the citation, using the structured information contained in bibliographic records to try and understand the unstructerd citation. Eytan Adar and I [3] proposed one scheme for linking the two.

7.1.5 Enabling librarianship and human input

The automatic processes for identifying and merging bibligraphic records work quite well in general, but human intervention would be helpful for correcting the errors that do occur. In general, the system should allow users to make corrections and changes to the bibliographic records and to the author-title clusters.

There are at least two different actions that a librarian might want to perform. First, the librarian should be able to change the contents of an author-title cluster by explicitly labelling a pair of records as related or not related. Marking two records as related would cause the author-title cluster to contain all records that have the same author and title fields as one of the two records.

Second, librarians should be able to label the quality of a source record or a particular collection of source records. Even a simple quality control scheme that allowed records to be marked as high quality, low quality, or mixed quality would improve the creation of composite records.

The library collection would also benefit from other kinds of human interaction. The collection of bibliographic records and a system for managing information dossier provides a basic infrastructure for supporting collaborative and cooperative work. An annotation service that allowed users to share reviews and critiques of documents is an example of such a service.

7.2 Conclusions

The two primary conclusions to draw from this work are that bibliographic relationships can be automatically identified in mixed-quality source records and that freely available bibliographic information can provide the basis for a useful and relatively complete index of the computer science literature.

The author-title clustering algorithm, described in Chapter 3, successfully identifies related bibliographic records that describe the same work. The algorithm tolerates errors in the records and variability in the cataloging practices, but maintains a tolerably low error rate; testing the algorithm with a small, controlled sample showed that it identified more than 90 percent of the related records and mistakenly linked records for two different works less than 1 time in 100. The effects of mistaken links are mitigated by presenting the user with information about the amount of variation in the underlying records.

The clustering algorithm uses a full-text index of the source records to limit the number of inter-record comparisons and to overcome errors in the author and title fields have have caused other algorithms to fail.

The Digital Index for Works in Computer Science demonstrates that it is possible to create a useful information discovery service from heterogenous sources of bibliographic information. It uses the clustering algorithm to integrate records from many sources and in multiple formats without any more coordination between sources than now exists. The system can automatically incorporate records from other collections and from individual citation lists without requiring that the creators of those records change their current practice.

The 240,000-record DIFWICS collection is broad in scope: It covers a large part of the computer science literature, including most areas of speciality and a large percentage of the total literature cataloged by the ACM between 1977 and 1993.

The DIFWICS catalog identifies individual works, linking together duplicate records and different documents with the same author and title, and helps users find online documents. The work-centered catalog reduces redundancy in search results and makes inter-document relationships clearer, and the preliminary automatic linking work speeds the process of searching for papers on the Web and suggests that the process could be fully automated.

Bibliography[1]

[1] Alf-Christian Achilles. bibmerge. [Program available via WWW], 1994. URL <http://liinwww.ira.uka.de/bibliography/tools/bibmerge> (version 28 Feb. 1995).

[2] Alf-Christian Achilles. A collection of computer science bibliographies. [WWW document], 1995. URL <http://liinwww.ira.uka.de/bibliography/index.html> (visited 10 Feb. 1995).

[3] Eytan Adar and Jeremy Hylton. On-the-fly hyperlink creation for pages images. In Shipman et al. [34], pages 173–176.

[4] Deborah Lines Andersen, Thomas J. Galvin, and Mark D. Giguere, editors. *Navigating the Networks: Proceedings of the ASIS Mid-Year Meeting*, Medford, NJ, 1994. American Society for Information Science (ASIS), Learned Information.

[5] Eva Bertha. Inter- and intrabibliographical relationships: A concept for a hypercatalog. In Helal [16], pages 211–223.

[1]Several documents cited in this thesis are available only in electronic form, via the World-Wide Web. It is difficult, however, to provide long-lasting citations for these documents. I have chosen to include the current URLs for the Web page and either the date of the most recent update to the page or the date of the last time I checked that the page was available (if the page was not dated).

[6] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, and Michael F. Schwartz. The harvest information discovery and access system. In Committee [10]. [WWW document] URL <ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/Harvest.Conf.ps.Z> (visited 10 Feb. 1995).

[7] C. Mic Bowman, Peter B. Danzig, Udi Manber, and Michael F. Schwartz. Scalable internet resources discovery: Research problems and approaches. *Communications of the ACM*, 37(8):98–107, August 1994.

[8] Michael K. Buckland, Mark H. Butler, Barbara A. Norhard, and Christian Plaunt. Union records and dossiers: Extended bibliographic information objects. In Andersen et al. [4], pages 42–57.

[9] Michael J. Carey and Donovan A. Schneider, editors. *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, May 1995. Also published as SIGMOD Record 24(2), June 1995.

[10] International World Wide Web Conference Committee, editor. *Proceedings of the 2nd International Conference on the World-Wide Web*, Chicago, December 1994. [WWW document, labeled "These documents are no longer being supported."] URL <http://www.ncsa.uiuc.edu/SDG/IT94/IT94Info.html>.

[11] Walt Crawford. *MARC for library use.* G. K. Hall, Boston, second edition, 1989.

[12] James R. Davis. Creating a networked computer science technical report library. *D-Lib Magazine [Online journal]*, September 1995. URL <http://www.dlib.org/dlib/september95/09davis.html>.

[13] Ahmed K. Elmagarmid and Calton Pu. Introduction to the special issue on heterogeneous databases. *ACM Computing Surveys*, 22(3):175–178, September 1990.

[14] M. C. Harrison. Implementation of the substring test by hashing. *Communications of the ACM*, 14(12):777–779, December 1971.

[15] Michael Heaney. Object-oriented cataloging. *Information Technology and Libraries*, 14(3):135–153, September 1995.

[16] Ahmed H. Helal, editor. *Opportunity 2000: understanding and serving users in an electronic library.* Essen University Library, 1993.

[17] Mauricio A. Hernández and Salvatore J. Stolfo. The merge/purge problem for large databases. In Carey and Schneider [9], pages 127–138. Also published as SIGMOD Record 24(2), June 1995.

[18] Thomas B. Hickey and David J. Rypka. Automatic detection of duplicate monographic records. *Journal of Library Automation*, 12(2):125–142, June 1979.

[19] Robert E. Kahn. An introduction to the cs-tr project. [WWW document], December 1995. URL <http://www.cnri.reston.va.us/home/cstr.html> (version 11 Dec. 1995).

[20] Karen Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377–439, December 1992.

[21] Carl Lagoze and James R. Davis. Dienst: An architecture for distributed digital libraries. *Communications of the ACM*, 38(4):47, April 1995.

[22] Leslie Lamport. *Latex: a document preparation system.* Addison-Wesley, 2nd edition, 1994.

[23] Rebecca Lasher and Danny Cohen. A format for bibliographic records, June 1995. Internet Engineering Task Force, RFC 1807.

[24] David M. Levy and Catherine C. Marshall. Going digital: A look at assumptions underlying digital libraries. *Communications of the ACM*, 38(4):77–84, April 1995.

[25] Clifford Lynch and Hector Garcia-Molina, editors. *Interoperability, Scaling and the Digital Libraries Research Agenda*. HPCC/IITA Working Group, August 1995. A Report on the May 18-19, 1995 IITA Digital Libraries Workship.

[26] Clifford A. Lynch, Avra Michelson, Craig Summerhill, and Cecilia Preston. The nature of the nidr challenge. Technical report, Coalition for Networked Information, 1995. URL <http://www.cni.org/projects/nidr/www/toc.html> (visited 12 Feb. 1995).

[27] Keith D. MacLaury. Automatic merging of monographic data bases–use of fixed-length keys derived from title strings. *Journal of Library Automation*, 12(2):143–155, June 1979.

[28] Edward T. O'Neill, Sally A. Rogers, and W. Michael Oskins. Characteristics of duplicate records in oclc's online union catalog. *Library Resources and Technical Services*, 37(1):59–71, 1993.

[29] Edward T. O'Neill and Diane Vizine-Goetz. Quality control in online databases. In Williams [46], pages 125–156.

[30] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In Yu and Chen [50], pages 251–260.

[31] M. J. Ridley. An expert system for quality control and duplicate detection in bibliographic databases. *Program*, 26(1):1–18, January 1992.

[32] Gerard Salton, James Allan, Chris Buckley, and Amit Singhal. Automatic analysis, theme generation, and summarization of machine-readable text. *Science*, 264:1421–1426, June 1994.

[33] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*, chapter 6.2 Vector Similarity Functions, pages 201–204. McGraw-Hill, 1983.

[34] Frank M. Shipman, III, Richard Furuta, and David M. Levy, editors. *Proceedings of Digital Libraries '95*, Department of Computer Science, Texas A&M Universty, College Station, TX 77843, June 1995. Hypermedia Research Laboratory.

[35] Narayanan Shivakumar and Hector Garcia-Molina. Scam: A copy detection mechanism for digital documents. In Shipman et al. [34].

[36] Narayanan Shivakumar and Hector Garcia-Molina. The scam approach to copy detection in digital libraries. *D-Lib Magazine [Online journal]*, November 1995. URL <http://www.dlib.org/dlib/november95/scam/11shivakumar.html>.

[37] Richard P. Smiraglia and Gregory H. Leazer. Toward the bibliographic control of works: Derivative bibliographic relationships in the online union catalog. In *OCLC Research Bulletin*, pages 56–59. 1994.

[38] Guy L. Steele, Jr. *Common LISP*, chapter 6.3 Equality Predicates, pages 103–110. Digital Press, 1990.

[39] Elaine Svenonius, editor. *The Conceptual Foundationgs of Descriptive Cataloging*. Academic Press, San Diego, Calif., 1989.

[40] Barbara B. Tillett. Bibliographic structures: The evolution of catalog entries, references, and tracings. In Svenonius [39], pages 149–166.

[41] Barbara B. Tillett. A taxonomy of bibliographic relationships. *Library Resources & Technical Services*, 35(2):150–158, 1991.

[42] Stephen R. Toney. Cleanup and deduplication of an international bibliographic database. *Information Technology and Libraries*, 11(1):19–28, March 1992.

[43] Marc Van Heyningen. The unified computer science technical report index: Lessons in indexing diverse resources. In Committee [10]. [WWW document] URL <http://www.cs.indiana.edu/ucstri/paper/paper.html> (visited 10 Feb, 1995).

[44] Stuart Weibel. Automated cataloging: Implications for libraries and patrons. In F. W. Lancaster and Linda C. Smith, editors, *Artificial Intelligence and Expert Systems: Will They Change the Library?*, pages 67–80. University of Illinois, 1992.

[45] Gio Weiderhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, mar 1992.

[46] Martha E. Williams, editor. *Annual Review of Information Science and Technology*, volume 23. Elsevier Science Publishers B.V., 1988.

[47] Martha E. Williams and Keith D. MacLaury. Automatic merging of monographic data bases–identification of duplicate records in multiple files: The IUCS scheme. *Journal of Library Automation*, 12(2):156–168, June 1979.

[48] Patrick Wilson. The second objective. In Svenonius [39], pages 5–16.

[49] Tak W. Yan and Hector Garcia-Molina. Duplicate detection in information dissemination. In *Proceedings of 21st International Very Large Database Conference (VLDB)*, September 1995. Zurich, Switzerland.

[50] P. S. Yu and A. L. P. Chen, editors. *Proceedings of the 11th International Conference on Data Engineering*, Taipei, Taiwan, March 1995. IEEE Computer Society Press.