Information Finding in a Digital Library: the Stanford Perspective

Tak W. Yan and Hector Garcia-Molina
Department of Computer Science, Stanford University, Stanford, CA 94305
{tyan, hector}@cs.stanford.edu

Abstract

In a digital library one of the most challenging problems is finding relevant information. Information finding is the research focus of the Stanford component of the ARPA-sponsored CS-TR Project, and the work has continued as one of the main thrusts in the Stanford Integrated Digital Library project [14]. In this paper we discuss some of the emerging issues in information finding, such as text-database discovery, efficient information dissemination, and copy detection and removal. We also outline our approaches to these issues.

1 Introduction

Computers and networks are rapidly making available vast amounts of electronic information. At the same time, traditional sources of information and entertainment such as books, papers, movies, and music are being provided digitally. Thus, people will have at their fingertips, either at home or at the office, a massive "digital library" that will fundamentally change how they access and process information.

Already today, computers and networks are increasingly used for retrieving information in a wide variety of ways. Conventional library systems, like Folio at Stanford and Melvyl at the University of California, allow users to remotely search online catalogs for bibliographic information. Commercial information providers (e.g., Dialog) offer data such as newspaper and magazine articles, company profiles, library card catalogs, and a lot others. Electronic bulletin board systems like USENET News have readership in millions and daily traffic in tens of megabytes. Topics covered range from software fixes to philosophical discussions, from items for sale to investment tips. Increasingly popular are some new wide-area information systems, notably gopher [19], WAIS [16], and World-Wide Web [1], that allow users to provide as well as to access text, hypertext, image, audio, and video data. With the advance in technologies and further proliferation of communication networks, we will soon have even more sources in greater variety and volume.

It is this emerging massive aggregate of heterogeneous information sources that will evolve into our digital library of the future. However, there is still one catch: before a "needle" of knowledge can be tapped by the end-user, it has to be found in the "haystack" of the world's information sources. To illustrate, consider searching for information hidden among the millions of documents residing in the tens of thousands of gopher, WAIS, and World-Wide Web sources available today. (According to one estimate [18], there are over 4 million pages in the World-Wide Web system as of June 1995.) Say a patent lawyer wishes to access technical reports written on query optimization techniques in relational databases. Suppose he sends his query to *all* sources. The sheer number of available

sources is so large that such a brute force information search is outrageously expensive or even futile. Thus, one of the key questions for the digital library of the future is how to search over large numbers of distributed sources and locate the right sources to direct a search to. We need mechanisms that scale with the number of sources. Since many of the sources available are text databases, an important subproblem is *text-database discovery*, an issue we elaborate on in Section 2.

Complimentary to a retrospective search mechanism, it is also important to provide an *efficient information dissemination* mechanism in the dynamic environment of a digital library. In this case, the user submits a long-term profile, made up of a number of queries, to the library. The profile will be continuously evaluated, and the user will passively receive filtered information. This way, instead of making the user go after the information, the relevant information automatically flows to the user. Again, to support information dissemination, scalable techniques are needed. To motivate this, we may look at some statistics from the SIFT [26] system that we have set up at Stanford. SIFT is an information dissemination system that delivers new, relevant USENET News articles to users. The system has been in operation since February 1994. Initially, the system processed on average some 30,000 documents USENET articles per day, matching them against some 2,000 queries from 800 users. Both the size of the user population and the volume of incoming information have been increasing ever since. As of June 1995, the system has over 10,000 subscribers worldwide, 24,000 long-term queries, and is processing close to 90,000 articles a day. These numbers illustrate that it will be a challenging task to cope with the ever growing user population and data volume. We will outline our approaches to tackle this issue in Section 3.

Whether in the search or in the dissemination scenario, a problematic issue is the proliferation of redundant information. Due to a variety of reasons, duplicate documents in digital media are very common (see Section 4). The existence of redundant information impairs the usefulness of search/dissemination systems. For instance, when searching a bibliographical database, users find duplicate information items intermixed in retrieval results hard to distinguish. In information dissemination, it is important that *new* information is delivered. If a document is delivered to a user, its exact/close copies will certainly reach the user over and over again. In an experiment carried out to investigate the degree of duplication in the articles sent out by SIFT [30], we found that on average some 18% of articles received by a user overlap 80% or more in content (e.g., number of sentences) with some articles seen previously. Understandably, SIFT users have complained emphatically about this. This is a serious problem, not just for SIFT, but for any system containing losely controlled information. The problem only gets worse if one accesses multiple sources at once, since it is likely that information will be replicated across them. Thus, we believe that supporting *copy detection and removal* is another critical issue in information finding. We discuss this further in Section 4.

We have been investigating these issues under the Stanford component of the the ARPA-sponsored CS-TR Project. (Carneigie Mellon University, University of California at Berkeley, Massachusetts Institute of Technology, and Cornell University are also participating in the project. A goal of the CS-TR Project has been to provide a digital library of Computer Science Technical Reports. Such a library is now available; see URL http://elib.stanford.edu.) Our information finding work has continued as one of the main thrusts in the Stanford Integrated Digital Library project [14]. In the rest of this paper we will briefly summarize this work. It represents research done and ideas contributed by Sergey Brin, James Davis, Luis Gravano, Narayanan Shivakumar, Anthony Tomasic, in addition to the authors.

2 Text-Database Discovery

Consider the patent lawyer example again. Say he needs an "exhaustive match," that is, needs to identify every single document in the sources that matches the query. For simplicity, let us view the documents as free text; the discussion generalizes to semi-structured data.

For an exhaustive match, the options are not very attractive. One is to ship the query to all possible information sources. This generates an enormous amount of network traffic and load at the sources. In other words, every information source on the network would be processing every query generated anywhere on the network. Clearly, this approach does not scale to large numbers of sources. Another alternative is to build a complete inverted index [22] of all the documents. By searching in this index, we could match exactly the query to all relevant documents. Unfortunately, this approach does not scale well either. For example, a full text index is usually the same size as the documents it references, so we would need to build a structure of the same size as all the documents in the network.

A third strategy for exhaustive matches is to construct some type of document or index hierarchy. For example, documents on a particular topic would be stored on a fixed source. If say the lawyer in our example has a query for documents on relational databases, the query would be routed to the computer science source. (This source could be physically distributed.) An alternative is to not partition the documents but the index (see for example [24]). Thus, the query would be routed to the computer science index machine. It could in turn provide a list of all the documents that match, even though the documents could be on a variety of machines.

While some form of the hierarchical scheme may be attractive, it seems to require some type of network wide agreement on how to partition the documents or the indexes. Thus, as an alternative to all these exhaustive match strategies, let us explore the approximate match space.

There are at least two ways to interpret an approximate query. One is that the user only wants to find a subset of the matching documents. That is, the user wants some information on a subject, but does not require every single relevant document. A second scenario is to view the approximate query as the first phase of a search process. The result is viewed as a "hint" of the sites that are likely to contain matching documents. The user then directs the query to those sites. If the user wishes additional documents, he can route the query to sites beyond the initial set. The problem of locating potential text sources to direct a search to is called the text-database discovery problem.

There are a variety of strategies to solve this problem. One simple one (used by WAIS) is to provide a "directory of sources." This "master" directory contains a set of entries, each describing (in English) the contents of a source on the network. The user first queries the master directory, and once he has identified potential sources, directs his query to them. One disadvantage is that the user typically needs two queries; e.g., "what sources have computer science documents in them?" and "find documents on relational databases." Also, the master directory entries have to be written by hand to cover the relevant topics, and have to be manually kept up to date as the underlying source content changes. Finally, it is not clear how accurate this scheme is; for example, a document on relational databases in a medical information source may be missed.

The master directory idea can be enhanced if we can use the semantics of queries and sources. In particular, assume we can automatically extract the semantic "concepts" involved in a user query. Also assume that we can extract the semantic concepts appearing in a collection of documents (in a source). Assuming that the number of concepts is much smaller than the number of words appearing in documents, then the concepts can be used for distributed indexing. That is, the user query is processed to extract the

concepts; these are matched against a concept "tree" and the potential sites identified. With the sample query "find documents on relational databases," the processing could extract the concept "computer science" and the index would determine that documents on this concept appear in the computer science and the medical sources.

At Stanford we have been exploring a different, purely syntactic idea for text-database discovery. Each source extracts a "histogram" of its word occurrences. For example, the computer science source could report that the word "relational" occurs 50,000 times, the word "databases" 75,000 times, and so on. This information is orders of magnitude smaller than a full index since it does not report the identities of the documents that have the terms. However, these histograms can still provide very useful information regarding what sites may have relevant documents. Say in our example, the query is Q = ("relational" $\wedge$ "database"); i.e., he is looking for documents with terms "relational" and "database" (a boolean query). Assume the following information is available:

Source A	10,000 occurrences of "relational" 5,000 occurrences of "databases"
Source B	10,000 occurrences of "relational" 0 occurrences of "databases"
Source C	50,000 occurrences of "relational" 75,000 occurrences of "databases"

Clearly, source B would not be a good place to search since we know no documents would match the query. We also know that both A and C could have matching documents. We do not know for sure if they do, since the occurrences of "relational" and "databases" could be in different documents. However, it seems likely that they do, and as a matter of fact, it seems much more likely that source C will have more matching documents since it has more occurrences of the desired terms. If we wanted to provide hints as to where to search, it would be reasonable to suggest source C as the first place to search, and source A as the second.

We refer to this approximate search technique based on term frequencies as GlOSS (Glossary of Servers Server). In [12, 13], we present the GlOSS technique for the boolean retrieval model and evaluate the "accuracy" of GlOSS using six sources available to us at Stanford. (For example, one is INSPEC, a collection of abstracts on Computer Science, Electrical Engineering, and Physics papers; another is PSYCINFO, a database of Psychology articles; a third is ABI, abstracts for Business periodicals.) We also use traces of real queries submitted against these sources. For each query, we can compare the sources suggested by GlOSS to the sources that actually contain the most matching documents (we find the latter by actually running each query against all the sources).

Our results are very encouraging. On the order of 90% of the time GlOSS does "the right thing." That is, in roughly 90% of the queries, GlOSS does give as a hint the sources that have the most matching documents. (There are actually several ways to measure success, so a result like this has to be qualified by the type of metric used; this is detailed in [12].) In addition, we analyze the storage required by GlOSS, and show that it only needs about 2% of the space required by a full-text index over the same data. There are also a variety of improvements possible. For example, it is possible to truncate the histogram, omitting information on terms that occur say one or two times in a source. This reduces the size of the histogram without affecting the accuracy significantly.

In [11], we extend GlOSS to gGlOSS to cover databases using the vector-space retrieval model [22]. The techniques are again evaluated using real-user queries and 53 USENET newsgroup databases. The approach is further generalized to building a hierarchy of gGlOSS brokers.

A GlOSS server has been implemented, keeping information on a number of WAIS-indexed sources. The reader is encouraged to try it out at

the URL http://gloss.stanford.edu in the World-Wide Web system. A collector program is also provided for gathering the necessary information from sources.

3 Information Dissemination

Recent research activities in information dissemination (also known as information filtering and routing) include efforts to improve the accuracy of the filtering process [17, 4, 5, 25], and to support collaborative or social filtering [10]. When used on a large scale, the efficiency (performance) aspect of the dissemination mechanism also needs to be addressed; this is the focus of our research in this area.

Let us first consider the design of a distributed information dissemination system. Some earlier experimental systems also provide distributed information dissemination. For example, the Community Information System [9] broadcasts (via radio channel) all updates to all users, who then apply their filters locally. However, relying solely on user-end filtering is expensive since network bandwidth is wasted to transmit irrelevant information and a lot of wasteful local processing is done. Another extreme is to perform filtering at the source end. In this case, the user will have to locate and replicate his profile at *all* possible sources. This is clearly not a very attractive solution. We focus on the scenario in which we have an intermediate server that collects information from a set of sources and routes it to the user. To scale with the size of user population and the rate of information generation, there can be several servers on the network.

When there are multiple servers, the distribution of profiles and documents becomes an interesting issue. Suppose we make the following assumption: every profile must "see" every document to determine if it is relevant. Then what options do we have to coordinate the servers? One option is to post each profile at only one server. Then every document must be broadcasted to every server, and consequently the document arrival rate at each server is high. Another disadvantage is that if a server goes down, the users it serves will not be able to receive information for the duration. To achieve higher availability, we may replicate the profile at more than one servers. Suppose we take this to the extreme and replicate a profile at all servers. Then a new document need to be sent to only one server, but the number of profiles that a server processes is large. There are immediate solutions between these two. Say we denote the set of servers that a profile P is posted at by S_P, and the set of servers that a document D is sent to by S_D, then to satisfy our assumption, we must guarantee that S_P intersects S_D for every P and D. This bears resemblance to the idea of quorum consensus in replicated data management [7, 8]. In [27], we study this problem in depth. We adopt structured quorum protocols for use in information dissemination and compare them with respect to performance and availability. Our results show that the grid protocol [3] with balanced document and profile quorum sizes is a good distribution scheme.

To scale with the number of users and the information creation rate, it is also important to streamline the processing that a server has to perform; otherwise it will easily become the bottleneck of the filtering network. Hence, we need to devise efficient data structures and algorithms for a single server to use in the filtering process.

To illustrate the situation, let us consider a simple example. Suppose a server has three boolean queries Q_1 = ("bill" $\wedge$ "jogging"), Q_2 = ("bill" $\wedge$ "hillary"), and Q_3 = ("socks"). (Recall the notation Q_1 = ("bill" $\wedge$ "jogging") means that query Q_1 is subscribing to documents that contain both words "bill" and "jogging.") Say a new document D with words "bill", "likes", and "jogging" has arrived. One way to process D is to build a hash table for D that lets us quickly tell if it has a given word. We can then run through all the queries and check them. For example, we check that D has "bill" and "jogging" and so we send D to the person that posted Q_1.

Clearly, this brute force strategy will not scale with the number of queries and the document arrival rate. One way to reduce overhead is to batch documents together for processing, but the timeliness of the documents will be sacrificed.

Inverted indexes have been used by information retrieval systems to speed up conventional retrospective search, namely by building an index of documents. Here, we borrow the idea but instead build a *query index*. The index associates every word with a list of queries that reference it. Hence in our example the word "bill" has a list containing queries Q_1 and Q_2, the list of "jogging" has query Q_1, that of "hillary" has query Q_2, and that of "socks" has Q_3. The situation is symmetrical to conventional retrieval [22]. In that case, we receive a query and check it against an index of documents. In the dissemination case, we get a document and check it against an index of queries. In retrospective search, to process the query against the index, we perform set operations on the lists of the queried words (assuming a boolean query), and the result of the operations is the answer to the query. For example, the $\wedge$ (AND) operator is processed by intersecting the lists; the $\vee$ (OR) operator is processed by merging (union) the lists. However, in processing inverted lists of queries, we cannot use set operations directly. In our example, the intersection of the lists for the words in D (i.e., "bill", "likes", and "jogging") gives us nothing (the list of "likes" is empty). And if we merge the lists, we get both Q_1 and Q_2, but Q_2 does not match D.

Fortunately, all is not lost. We may observe that the result of the union of the lists contains queries that potentially match the document: each shares at least one word with the document. The merged list is in fact a superset of the desired answer, and so we can screen out the superfluous queries by checking them against the document, using the hash table as described above. With this strategy we avoid checking all queries, i.e., queries that do not contain "bill", "likes", or "jogging", (Q_3 in our example) will not be considered.

In our work, we explore this strategy, as well as a number of indexing alternatives, in detail. We consider both boolean queries [29] (the matched document must be an exact match to the query) and vector space queries [28] (documents "similar" to the query are sent to the user). We evaluate their performance and show that the indexing methods are orders of magnitude more efficient than the brute force strategy in terms of processing time.

Selected query indexing techniques have been implemented in the Stanford Information Filtering Tool (SIFT) system [26]. Using SIFT, we have set up the information dissemination server for USENET News articles mentioned in Section 1. This server accepts boolean and vector space model queries from the user, and periodically informs the user of any new, matching articles. (The reader is encouraged to try out the system. For World-Wide Web access, connect to http://sift.stanford.edu. For email access, send an electronic message to netnews@sift.stanford.edu. The message should contain a single line with the word "help." Instructions will be returned automatically.) [1] As mentioned in the introduction, SIFT has become a popular source of information. With the efficient indexing and query processing techniques, SIFT has been shown to scale well to user population and information volume growth.

4 Copy Detection and Elimination

Duplicate documents arise for many reasons. Perhaps the foremost is that digital documents can be reproduced with extreme ease and at almost no cost. For example, in USENET News, a user can cross-post a news article in many newsgroups simultaneously. He may repost it a few

[1] There is also a companion service for Computer Science Technical Reports (CS-TR) at elib@sift.stanford.edu.

days later, again to multiple newsgroups. Further, there is no loss in the quality of the copies, and thus they can be replicated again. For example, a user may retransmit a copy of an article he has received in yet some other newsgroups or channels (e.g., mailing lists), resulting in further propagation.

A duplicate needs not be an exact copy as described above, but in general a document closely similar in content to some other document and not giving the user any extra information. For example, in traditional library catalog system, duplicate bibliographical records referring to the same technical reports are very common. The reasons leading to the existence of duplicate records are mainly human input errors or inconsistencies; e.g., different practice in record creation, translation differences, and typographical errors. The traditional library community has recognized the problem of duplicate detection, especially in systems that provide union catalogs merging multiple bibliographical databases [15, 21, 20].

In modern information systems, documents are not limited to short, structured bibliographical records, but rather full-text documents existing in different media types, with complex inter-document relationships among them. This rich content gives rise to more sources of duplication. One major source is the different media formats in which a document may exist. For instance, a technical report may be written in LaTeX, and converted into DVI and postscript. It may also be converted into plain text or HTML. The hard-copy may be scanned in as images and then converted to text via optical character recognition (OCR). All these may be made available on-line. Another source of duplication is versioning. A document may undergo a number of versions in its life-span. For example, a technical report may have a short and a full version. A user, after reading the short version, may find the full version a duplicate.

As pointed out in Section 1, redundant information is a nuisance to the user of a search/dissemination system. To support duplicate removal, the first step is to detect when two documents are duplicates, and the type or degree of duplication. In [30], we propose a taxonomy of duplicates in digital documents. We classify duplicates into intentional and extensional duplicates. Intentional duplicates are those intended by the creators to be duplicates, even though they may have different content (e.g., different bibliographical records refering to the same technical report). Extensional duplicates are those with overlapping textual content.

In [2], we propose a system for registering documents and then detecting extensional duplicates. A document to be registered is divided into (maybe overlapping) text chunks (e.g., sentences), using some chunking function. The text chunks are hashed into integer values, and the set of hash values forms a representation of the document. (The actual document is not stored in the system.) To check a document D against the registered documents, D is similarly chunked, and the percentage of overlapping hash values is returned as an estimate of the percentage of overlapping chunks. Using this scheme, the storage required to register a collection of documents is a lot less than storing entire documents. The processing is also efficient, as we rely on hashing to compare documents. Sampling techniques (taking a sample of the text chunks) can further reduce the costs. In [2], metrics required for evaluating detection mechanisms (covering accuracy, efficiency, and security) are proposed, and a prototype system called COPS is described and evaluated. Another scheme, called SCAM, based on comparing word occurrence frequencies of documents is proposed in [23].

The second step in supporting duplicate removal is to remove duplicates according to the expectation of the user. Individual users may have different requirements for duplicate elimination. A user, depending on what documents he has read previously, may consider a document a duplicate while another user does not. He may also want to specify how close a document must

be to a seen one for it to be a duplicate. For example, a user may find an article not very interesting and want to remove any duplicate more than 70% similar in content. On the other hand, if a user receives an interesting document, he may want to remove only identical copies. Thus, the removal of duplicates operates on a per user, per document basis – each document read by a user generates a request for duplicate removal, termed a *duplicate restraint.* The scale of global information systems makes the processing of duplicate restraints challenging. For example, in information dissemination, both the number of users and the number of incoming documents are large, so the number of restraints is very large. In SIFT, some 90,000 documents are matched against profiles of over 10,000 users every day; if we keep restraints valid for say ten days, we estimate that more than a million restraints need to be checked for every incoming document. In [30], we consider the design of a duplicate removal module to manage and process a large number of duplicate restraints in information dissemination.

5 Conclusions

In this paper we have briefly outlined some of the critical information finding problems for digital libraries, and have described some of the work at Stanford. There are of course many other important issues beyond information finding and the approaches discussed here. For a good overview of the general area of digital libraries, we refer the reader to [6].

References

[1] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann. World-Wide Web: The information universe. *Electronic Networking: Research, Applications, and Policy*, 1(2):52–8, 1992.

[2] S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. In *Proc. ACM SIGMOD Conference*, 1995.

[3] S.Y. Cheung, M.H. Ammar, and M. Ahamad. The grid protocol: A high performance scheme for maintaining replicated data. In *Proc. International Conference on Data Engineering*, pages 438–45, 1990.

[4] W.B. Croft. The university of massachusetts tipster project. *SIGIR Forum*, 26(2):29–33, 1992.

[5] P.W. Foltz and S.T. Dumais. Personalized information delivery: an analysis of information filtering methods. *Communication of the ACM*, 35(12):29–38, 1992.

[6] E. Fox, R. Akscyn, R. Furuta, and J. Leggett. Editors. Special Issue on Digital Libraries. *Communications of the ACM*, 38(4):22–96, 1995.

[7] H. Garcia-Molina and D. Barbara. How to assign votes in a distributed system. *Journal of the ACM*, 32(4):841–60, 1985.

[8] D. Gifford. Weighted voting for replicated data. In *Proc. Symposium on Operating System Principles*, pages 150–62, 1979.

[9] D. Gifford, R. Baldwin, S. Berlin, and J. Lucassen. An architecture for large scale information systems. In *Proc. Symposium on Operating System Principles*, pages 161–70, 1985.

[10] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communication of the ACM*, 35(12):61–70, 1992.

[11] L. Gravano and H. Garcia-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In *To appear in Proc. 21st VLDB Conference*, 1995.

[12] L. Gravano, H. Garcia-Molina, and A. Tomasic. The effectiveness of GlOSS for the text-database discovery problem. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, 1994.

[13] L. Gravano, H. Garcia-Molina, and A. Tomasic. Precision and recall of GlOSS estimators for for database discovery. In *Proc. of the International Conference on Parallel and Distributed Information Systems*, 1994.

[14] Stanford Digital Libraries Group. The Stanford Digital Library Project. *Communications of the ACM*, 38(4):59–60, 1995.

[15] T. Hickey and D. Rypka. Automatic detection of duplicate monographic records. *J. Libr. Automn*, 12(2):126–42, 1979.

[16] B. Kahle and A. Medlar. An information system for corporate users: Wide Area Information Servers. *Connexions - The Interoperability Report*, 5(11):2–9, 1991.

[17] S. Loeb and D. Terry. Editors. Special Issue on Information Filtering. *Communications of the ACM*, 35(12):26–81, 1992.

[18] Michael Mauldin. *Lycos - the catalog of the Internet.* Lycos, Inc., URL http://www.lycos.com, 1995.

[19] M. McCahill. The internet gopher protocol: A distributed server information system. *Connexions - The Interoperability Report*, 6(7):10–14, 1992.

[20] E. O'Neill, S. Rogers, and W. Oskins. Characteristics of duplicate records in OCLC's online union catalog. *Library Resources & Technical Services*, 37(1):59–71, 1993.

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

[22] G. Salton. *Automatic Text Processing.* Addison Wesley, Reading, Massachusetts, 1989.

[23] N. Shivakumar and H. Garcia-Molina. SCAM: A copy detection mechanism for digital documents. In *Proc. 2nd International Conference in Theory and Practice of Digital Libraries*, 1995.

[24] A. Tomasic and H. Garcia-Molina. Performance of inverted indices in distributed text document retrieval systems. In *Proc. of the International Conference on Parallel and Distributed Information Systems*, pages 8–17, 1993.

[25] M.F. Wyle and H.P. Frei. Retrieval algorithm effectiveness in a wide area network information filter. In *Proc. ACM SIGIR Conference*, pages 114–22, 1991.

[26] T. Yan and H. Garcia-Molina. Sift – a tool for wide-area information dissemination. In *Proc. 1995 USENIX Technical Conference*, pages 177–86, 1995.

[27] T.W. Yan and H. Garcia-Molina. Distributed selective dissemination of information. In *Proc. Parallel and Distributed Information Systems*, pages 89–98, 1994.

[28] T.W. Yan and H. Garcia-Molina. Index structures for information filtering under the vector space model. In *Proc. International Conference on Data Engineering*, pages 337–47, 1994.

[29] T.W. Yan and H. Garcia-Molina. Index structures for selective dissemination of information under the boolean model. *ACM Transactions on Database Systems*, 19(2):332–64, 1994.

[30] T.W. Yan and H. Garcia-Molina. Duplicate removal in information dissemination. In *To appear in Proc. 21st VLDB Conference*, 1995.