NISTIR 5243

Towards Flexible Distributed Information Retrieval

David W. Flater
Yelena Yesha

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

NIST

NISTIR 5243

Towards Flexible Distributed Information Retrieval

David W. Flater
Yelena Yesha

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

August 1993

Towards Flexible Distributed Information Retrieval

David W. Flater* Yelena Yesha†

Abstract

Many years of research into better and more effective information retrieval methods have yielded a collection of good information retrieval techniques. Careful use of these techniques can result in an information retrieval system that can answer high-level queries with surprising accuracy when the domain of discourse is small.

At the same time, the growth of wide-area networks, and the corresponding growth in the amount of information available through them, has caused many archives and information bases to become lost in a great connectionless cloud. Without prior knowledge of site names and access methods, users are often unable to make contact with potentially useful information bases.

To make the best use of networked resources, it is necessary to increase the amount of cooperation between network sites. By allowing sites to share the responsibility of routing requests for data and caching replicas of frequently used data, it is possible to eliminate the inefficiencies resulting from the isolation of individual sites and users. However, the system must remain flexible enough to allow the integration of existing information bases. This chapter will discuss a distributed information system being developed to achieve this goal.

1 Introduction

Information Retrieval is the process of locating the data which are the best answer to a user's query. In the case of text retrieval, it is "leading the user to those documents that will best enable him/her to satisfy his/her need for information[BC92b, Rob81]." Information Retrieval is part of every database, every catalog, and every file system. It is such an integral part of all of these things that it is seldom thought of as an independent process.

This could soon change, however. The sharp increase in the amount of information available to the average user in recent years has drawn attention to the shortcomings of commonly used information retrieval systems. For years, very restrictive and inferior methods have been used to retrieve information. These methods often force the user to know the exact name or other identifier

*Computer Science Department, University of Maryland Baltimore County, Baltimore, MD 21228.

†Supported in part by the Information Systems Engineering Division of the National Institute of Standards and Technology, Gaithersburg, MD 20899.

of the data to be retrieved. Sometimes one is lucky enough to be able to use regular expressions or keywords to locate data which cannot immediately be identified. Nevertheless, locating the desired information is often a frustrating, time-consuming, and even futile task.

Fortunately, information retrieval techniques have been under investigation for years, and a vast improvement could be realized in current information systems merely by implementing known algorithms. Most established techniques are designed for handling textual data, which is in fact where the greatest amount of difficulty is being experienced, but encouraging results for non-textual data are already being reported. In theory, the task of easing the general burden of information retrieval on the user has been well undertaken. In practice, the systems which are widely used every day are still relatively primitive.

If all information retrieval systems were upgraded overnight to use the best techniques available, finding information would be easier – but it would still not be trivial. There is another layer to the problem, which could humorously be called the *network* layer, but whose accepted name is the Resource Discovery Problem[Sch88]. Proliferation of information is not the only difficulty. There is also a proliferation of information bases. In an information retrieval system, one may start a serial search through the entire information base for a particular piece of information. One does not, however, have the option of starting a serial search over the entire Internet for a particular kind of database. In fact, even if one could connect to every site on the network, one would probably only be able to figure out how to access the available databases in the case of well-known archives and public catalogs.

At first glance, it would appear that the problem of finding the information base which would contain the desired information is just another form of information retrieval. One could establish a directory of information bases and use information retrieval techniques to find the correct information base. Later in this chapter, the reasons why this approach is inadequate will be presented.

This chapter is organized as follows. Section 2 provides a classification of information retrieval techniques. Section 3 describes the construction and use of thesauri and dictionaries for information retrieval. Section 4 discusses existing IR techniques and the use of these techniques to achieve fuzzy retrieval. Section 5 investigates the difficulties of extending IR into the distributed domain, and Section 6 describes an information system architecture which handles some of those problems.

2 Information Retrieval Techniques

IR techniques can be divided into three classes: tag-based retrieval, partial content-based retrieval, and full content-based retrieval. In tag-based retrieval, indices are painstakingly built by human beings, and the system uses more or less clever algorithms to search the index for relevant entries. (The term "index" is used loosely here to indicate any form of data keying, whether by means of inversion or signatures.) In partial content-based retrieval, indices are built automatically based on the content of data objects. In full content-based retrieval, queries are executed directly upon the data objects, and no index is required.

2.1 Tag-Based Retrieval

One of the simplest forms of IR is tag-based retrieval. When data objects are added to the information base, a human being has to provide meta-information, such as a subject classification or a set of keywords, for each data object. The information system then uses this meta-information as an index when it is called upon to retrieve data. The data objects themselves are never inspected by the information system; they are merely reproduced verbatim when their meta-information satisfies a user's query.

This form of information retrieval is used today in many applications. In cases where qualified personnel are available to perform the indexing for an information base of limited scope, such as a library of legal documents, these systems can provide extremely efficient information retrieval. However, consider the case of a public library where information of all kinds needs to be indexed. The library workers have the task of maintaining an internally consistent general classification system in the face of a constant influx of new documents. As a result, they will probably not distinguish between, say, distributed databases and distributed information systems, causing a degradation of retrieval precision. The library would require a staff of experts from every field to properly maintain their subject hierarchy.

2.2 Partial Content-Based Retrieval

Consider how much better the situation would be if the process of indexing an information base were automated. In place of the staff of experts needed to maintain a generalized subject hierarchy for a public library would be a library of software. Each program in the library would be maintained with assistance from field experts to perform accurate classification within one particular field of interest. In the best case one would also have a master program which could automatically determine the field of interest a document belongs to and pass it on to the "expert" in that field. The same software could then be distributed to all libraries, and the library workers would have a much simpler job.

It is on exactly this kind of technology that most current research into Information Retrieval focuses. Unfortunately, while there has been some success in the localized indexing problem for limited domains[DRST91, RS92], no one has yet succeeded in integrating many "experts" with a main program capable of solving the higher-level classification problem.

2.3 Full Content-Based Retrieval

An index must contain all the information which is required by the retrieval algorithm, including keyword frequencies or whatever else the algorithm might use. There is thus a conflict between the desire for thorough information retrieval and the desire for tight, efficient access structures. The more one restricts oneself to a fixed set of attributes, the more one is able to use optimized methods such as signature files[WL90]. The less brute-force searching is done or the less overhead is accepted, the less effective is the information retrieval.

The limits of information retrieval are attained by full content-based methods. These methods use no access structures whatsoever. The entirety of every data object in the information base is subject to scrutiny. No information whatsoever is omitted from the search. The benefits in terms of the system's ability to handle queries on obscure topics are phenomenal – but so are the costs.

To perform such a thorough search on a large library in a reasonable amount of time would likely be beyond the ability of even the most powerful parallel processing engines. However, it is possible that these techniques could be used on small databases of limited scope, provided that there is still a global classifier capable of determining the appropriate domain for a data object or query.

3 Thesauri and Dictionaries

One of the most common problems with text retrieval systems is that the user must often express the same query in many different ways until he/she stumbles onto some keywords that are in the index. The information systems are not sophisticated enough to map an arbitrary natural language term onto the set of terms in their indices which have approximately the same semantics, even if the natural language term is an exact synonym of an index term. Since users do not know the set of index terms, they are forced to use trial and error to find keywords that trigger the retrieval of relevant documents. The usefulness of such a system will vary drastically from one user to the next, depending on how close that user's vocabulary is to the index vocabulary. To solve this problem, work is being done to find ways to build thesauri and dictionaries which will enable efficient retrieval of text from natural language queries.

3.1 Thesauri

A thesaurus is a mapping from terms to terms which are semantically close. It may be viewed as a graph where each node contains a term and has links to all of its synonyms. A text retrieval system which is faced with a query term which does not appear in its index should be able to use a thesaurus to try alternate terms in the order of decreasing semantic similarity. If a user specifies a synonym of an index term, the system should be able to determine the index term. On the other hand, if the information base has been indexed inconsistently, a thesaurus should help alleviate the problem by making some terms nearly equivalent.

The popular information retrieval technique of indexing by word stems or trigrams[MK87] can be thought of as a heuristic for detecting synonyms in English text. The thesaurus which it implicitly generates is neither complete nor accurate. A complete and accurate thesaurus would be sufficient to maximize the efficiency of simple keyword-based retrieval systems. Where more complex retrieval methods are used, a dictionary may be needed to provide semantic information to the retrieval or query-parsing algorithms.

The degree of closeness for each mapping from a term to a synonym must somehow be expressed by a thesaurus. To avoid the scenario where every possible combination of terms is assigned a similarity weight, thesauri should be built with the idea in mind that one may navigate more than one link away from the term one is examining if none of its immediate synonyms is satisfactory. To find the synonyms of a word in order of decreasing similarity, a breadth-first search of the graph is performed. If weights are assigned to the links, a more expensive best-first search must be used to insure perfect sorting, but this may be more trouble than it is worth.

Thesauri can be extended to convey more semantic information on how terms are related to each other[Pai91]. For example, there can be a distinguished, directional link type which indicates that one term is a generalization of another. This link type allows an information system to generalize the terms of an overly specific query until some documents can be retrieved. However, it may be undecidable whether or not overspecification was responsible for a null response.

Early attempts at building thesauri automatically using purely statistical methods met with limited success. However, a recent attempt to improve the quality of automatic term association by using linguistic knowledge was successful[Rug92]. Linguistic analysis could be used to improve the signal-to-noise ratio of statistical methods by transforming those sentence structures which cause unrelated terms to co-occur. While reliable parsing of English grammar is exceedingly difficult, a surprisingly high success rate can be achieved using simple methods if one only wishes to recognize a limited subset of the possible sentence structures[Hob]. It should therefore be possible to build sufficiently accurate thesauri using a combination of statistical methods and limited parsing. Another novel approach to automatic thesaurus construction uses the complete link clustering algorithm to cluster the document collection prior to the construction of the thesaurus[Cro90]. This two-stage statistical approach, which has a strong theoretical basis, has also yielded promising results.

The uncertain accuracy of the resulting thesauri is not the only disadvantage of automatic thesaurus construction. While several different kinds of relations between terms can be distinguished by automatic methods[Rug92], it is usually the case that statistical methods point out the existence of relationships without classifying them[Pai91]. This places a restriction on the kinds of retrieval algorithms that can be used with automatically generated thesauri. However, for the retrieval methods which are most commonly used, automatic thesaurus generation provides a way to rapidly create many domain-specific thesauri until more accurate thesauri become available.

If one continues to build more and more semantic information into a thesaurus, at some point it ceases to be a thesaurus and becomes a dictionary. The essence of the distinction is that the possibilities for defining relationships between terms are exhausted and information related to the terms themselves, i.e. "definitions," begin to appear.

3.2 Dictionaries

When one crosses the line from text processing into text understanding, one also needs to trade in one's thesaurus for a dictionary. Any term-related information which would be useful during the retrieval process can be kept in a dictionary. This includes internal semantic representations that can be retrieved from the dictionary and used as building blocks for larger expressions.

As of now, dictionaries are primarily being used in expert systems and knowledge bases. However, for years it has been asserted that large amounts of "world knowledge" are needed for effective natural language understanding. Dictionaries might therefore be needed when thesauri become inadequate to support natural language information retrieval.

4 Fuzzy Retrieval

Thesauri are one of the tools that are being used to enable fuzzy retrieval, the probabilistic retrieval of data objects based on an estimate of their relevance to a query. Fuzzy retrieval is made necessary by the inherent limitations of non-fuzzy methods such as Boolean querying[SM83, BCP91]. Non-fuzzy methods are inflexible and require the user to have prior knowledge of the indexing scheme of the information base if efficient retrieval is to occur.

Most fuzzy retrieval is built upon the notion of a relevance or similarity function. Given a query which is not satisfied by any object in the information base, the similarity function provides an estimate of how similar each object is to the hypothetical object described by the query. The program can then select a small number of objects which are considered to be most similar and list them for the user, along with their similarity (or relevance) estimates.

Consider for example the case of retrieving abstracts with keyword queries. A direct keyword system returns only those abstracts in which every keyword appears at least once. In effect, a Boolean value is assigned (zero or one) to each term to indicate whether or not it appears in the abstract, and then the values are multiplied to determine the relevance of the abstract to the query. This system can be made nominally fuzzy by accumulating, rather than multiplying, the term values, so that the most relevant abstract is the one containing the largest number of query terms. While this nominally fuzzy approach is currently used in many on-line library catalogs, much better techniques are available.

Given a thesaurus, one can immediately improve on the above similarity function by assigning a relevance value which is less than one but greater than zero for any term with a synonym in the document, but which does not appear itself. The value assigned will depend on the thesaural similarity of the term and its synonym. The relevance function will then be fuzzy along the semantic axis as well as the syntactic one, with the result that retrieval will be much more consistent across different query vocabularies.

Even this similarity function is crude compared to many which appear in the literature. For example, it makes no attempt to weight query terms based on how often they appear in a given document, or how often they appear in the document collection as a whole. The best index terms are those which do not appear in so many documents as to be useless for retrieval, yet are not so infrequent that many relevant documents do not contain them. The tf-idf (term frequency – inverse document frequency) term weighting scheme is one way of biasing the search towards those terms which are most useful for retrieval[Cro90, SM83, SB88]. Furthermore, accumulation is only one possible way of combining term coefficients into a document coefficient. It is common practice to arrange term coefficients for queries and documents into vectors (the "vector space model"[SM83]) and use vector operations such as cosine to determine their similarity[Cro90].

Similarity functions do not exist only for text retrieval. A great deal of work is currently being done to allow information retrieval using images. One recent work determines the similarity of images which are comprised of iconic objects[CW92]. Specialized techniques have also been developed for retrieving documented software[MBK91]. However, it is the case that text retrieval has received vastly more attention than other kinds of retrieval, and until sufficient work has been done in this area, textual descriptions must be used to index non-textual data.

One class of fuzzy retrieval algorithms which does not depend exclusively on a similarity function (but which nevertheless may use one) is based on the inference net model of information retrieval[BC92b, TC91]. Information retrieval systems which use these algorithms can look like a combination of hypertext and expert systems. They use Bayesian logic to reason about the properties of data objects and then decide which of them are most likely to satisfy the query. A Bayesian inference network[BC92b, Pea88] is built for the object collection and for the query. The two are then connected so that the probability that each object satisfies the query can be computed. When feedback from the user causes changes in the subnet representing the user's query, the probabilities are recomputed and a new set of pertinent objects is returned. Instead of simply accumulating values, the inference net model deals in detail with the case where the probability that a given object is relevant to a query depends on multiple factors. However, the benefits of such a thorough analysis of relevance may not outweigh the additional complexity for many information retrieval applications.

4.1 Partial Content-Based Methods

The most direct way to adapt existing information retrieval systems to be partially content-based is to automate the process of indexing the data. Using this approach, the same old fuzzy retrieval mechanisms can be used to search the same kind of index. The only difference is that nobody had to go through the effort to build the index manually.

Automatic indexing is most frequently done using methods such as tf-idf to decide which of the terms in a document or document collection would make the best index terms. An index can then be generated which contains only those terms, along with their frequencies in each document. Additional lexical association metrics are the signal-to-noise (concentration), the variance (concentration/diffusion), and the discrimination values of terms[HVB90]. The discrimination value of a term reflects the amount of difference it makes to the document similarity function if that term is removed from consideration.

If an information retrieval system uses summaries or abstracts to index data, all is not lost. Some researchers hope to find ways of piecing together entirely nonanaphoric sentences to form a reasonably informative "abstract" of a document using sophisticated linguistic analysis. An easier approach can be used if something is known about the structure of the documents, as is the case with an information system of limited scope. For example, if all the documents describe analytical experiments, one can search for phrases such as "we conclude" or "we have shown" which often co-occur with important bits of summarized information. However, the problem in general is insidiously difficult, and simple-minded heuristics are unlikely to provide a solution[Pai90]. One approach to automatic keyword indexing[DRST91] employs techniques which have been used for automatic abstracting, but suggests that the extracted phrases simply be used for keyword searches and not regarded as full summaries.

Automatic indexing of images[RS92] is also an immense challenge for researchers, but it is receiving more attention and is likely to reach maturity before automatic abstracting. The cited work points out that "image retrieval by content cannot be attempted in general," but explains that, given a sufficient body of domain-specific patterns and rules for image analysis, one can identify the different objects in an image and build a representation of the image in terms of its objects, their number, their compositions, and their positions. The images themselves can then be indexed using these representations, and retrieval can be achieved by specifying the qualities to look for in the representations.

Lastly, some similar techniques are being used in the realm of hypertext. Attention is now being given to the use of text processing techniques to automatically build semantic network representations of input texts to facilitate hypertext browsing[MN92]. It is more difficult to automatically build a semantic network than to automatically index a data set. For indexing, all the program has to do is extract the most significant terms from a data object, or, depending on what sort of indexing is being used, determine which of a pre-selected list of index terms are descriptive of the datum. To build a semantic net, the program also has to add links connecting the important components of the datum to related concepts. The possible uses of semantic networks to enhance information retrieval capabilities are worthy of investigation.

5 Distributed Approaches to Information Retrieval

There have as of yet been few attempts to expand information retrieval into the world of networked and distributed computing. This is puzzling, considering the vast amount of attention that has been given to distributed databases and client-server architectures in recent years. Somewhat more frequent are applications of parallel processing hardware to cut down on the amount of real time needed to perform searches[CEMW91]. While brute force can no doubt do wonders for the effectiveness of information retrieval algorithms, this is only really helpful when the information base is homogeneous. The larger problem of networked information retrieval remains.

The most notable incursion of information retrieval into the networked computing domain is WAIS[Kah91]. WAIS (Wide-Area Information Server) is technically a client-server system. However, it is the first attempt to connect autonomous information bases over a wide-area network which has achieved a significant following.

When performing a search in WAIS, a user must select one or more servers from a list of servers which is provided. The user then enters a keyword query which is executed on the selected servers. A ranked list of matching data is shown when the search completes. The user may then select individual data to retrieve. WAIS has the ability to handle non-textual data, but this capacity is currently being used to a very limited extent. For example, some images can only be retrieved in WAIS by providing the name of the file in which the image is stored as the search criteria[Gol].

Other distributed approaches to information retrieval[HMTC91] preserve the single homogeneous information base flavor of existing information retrieval methods. This is in conflict with the rising trend of heterogeneous computing which is evident in the database community.

5.1 Current Research Issues

The first problem which must be solved to acquire information using FTP (File Transfer Protocol) is finding the address of an archive site which has the right type of data. Similarly, the first problem which presents itself to a new user of WAIS is that he/she must scroll through a long list of servers to locate a server which contains the desired kind of information. These are just two of many possible manifestations of the Resource Discovery Problem[Sch88], the fundamental problem which must be solved to truly achieve a global integration of information sources.

5.1.1 The Resource Discovery Problem

The problem of locating information on a large network is known as the Resource Discovery Problem. Before a user can interact with a database or archive to retrieve needed information, the user must find out *where* such a database or archive exists.

One approach to solving this problem is to accumulate information to help users find what they need. For this purpose, there currently are a number of on and off-line Internet directory services[Deu92] which let skilled users perform keyword searches to find out which Internet sites have the information they want. Unfortunately, the off-line directories cannot remain up-to-date, and the on-line services have limited effectiveness since they can only handle so much information. They cannot keep track of enough attributes of the data being archived to support general resource discovery. In many cases, one needs the name of the file which contains the desired information to find out the name of the site which owns it. There are also a variety of browsing and information-gathering tools[SHHH91, Rat92] which try to help the inexperienced user, but a browser also can only go so far to make a fragmented and heterogeneous set of resources appear to be unified.

The University of Colorado's Networked Resource Discovery Project has concentrated mainly on developing better directory services for the Internet; however, an early technical report describes their attempt to design a networked resource discovery system based on probabilistic multicasts and resource brokers[Sch90]. More recently, Matsushita Information Technology Laboratory has begun working on an information system as part of the Gold project[BC92a]. Its main improvement over [Sch90] is that the functionality of the resource directories has been specified in greater detail. The brokers, which are the maintainers of the resource directories, now use information retrieval techniques to classify queries and data in order to determine the location of the database capable of answering a query. The system discussed in Section 6 is unique in its brokerless approach to query routing and resource discovery and in its use of cooperative caching.

5.1.2 Large Information Networks

The proliferation of databases in government and industry has focused attention on the problem of integrating these databases. It is no longer practical for data to be disseminated by manually exporting it from one database and importing it into another; neither is it practical to call someone at a remote site each time a piece of information is needed. Since experience has shown that a large, tightly-knit system is too inert and unmaintainable to be a permanent solution[Wie92], work is currently being done to find ways of solving the integration problem by building a system on top of the many databases which are already established. The terms heterogeneous database, federated database, and mediated database all describe this kind of system. Different authors will assign different connotations to these terms, and precise definitions are difficult to achieve. As a group, however, they describe a particular class of systems whose goal is to unify existing databases.

The techniques which are being suggested to allow integration of heterogeneous databases may help to build an integrated information system. The notion of mediators, distinguished pieces of software whose job it is to allow a graceful interface between the outside world and an information base (or other mediators), can help solve the general heterogeneity problem in a modular fashion[Wie92].

6 Architecture for an Integrated Information System

6.1 Preliminaries

The authors have been working on an information system architecture which will eliminate the need for directories of archives, servers, and databases and provide a truly general-purpose information retrieval service[FY92b, FY93a, FY93b, FY93c, FY93d]. They assume that individual sites will host information bases of limited scope – the sorts of archives that existing information retrieval techniques can handle or will be able to handle with sufficient refinement. They then build a distributed information system architecture on top of these information bases to route general queries to the sites which can answer them and to cache replicas of frequently-used data objects.

A data object is a document, an image, a piece of software, or any other self-contained unit which can serve as the answer to a query. For most types of data it will be possible to assign a name, a summary, a list of keywords, or some other textual identifier which can be used by term similarity algorithms to determine whether cached objects are appropriate answers to queries which are received. Particularly strange types of data might require a specialized descriptor which only certain "specialist" sites can interpret; the generic textual descriptor of such data would merely indicate that the data required special processing. For classes of data which cannot be assigned any kind of identifier, all queries will have to be forwarded to the sites which host those classes of data, and replicas will not be created.

The information system is constructed by forming a virtual network of sites. The links in the virtual network are given time-varying costs which reflect the observed delay on the underlying wide-area network, or any other costs which must be minimized. The sites in the virtual network do not all have to own archives; many of them can simply cache replicas of data whose authoritative copies are elsewhere. Each network site will:

- Answer queries on data which are in its cache by sending a copy back towards the querying site;
- Forward queries on other data over the link which is believed to lead to the nearest copy of the desired data;
- Forward response messages along the least costly path to the querying site;
- Cache replicas of data contained in response messages;
- Replace older versions of data with newer versions when a response message is forwarded which contains a newer version of a datum in cache, or vice-versa;
- Process queries on data for which this site owns an archive (if applicable).

6.2 Query Routing

In order to know where to send a query, the information system must be able to determine the class of data which would satisfy a query. For example, a query asking for PC software should be sent to a site which archives PC software, and a query asking about the Civil War should be sent to a site hosting historical documents. Thus, it is expected that the queries received will contain sufficient information to allow this classification. Having to type "US history Civil War" instead of just "Civil War" is the price of general-purpose information retrieval. "Civil War" is only really sufficient if the information base is already restricted to historical documents. Of course, one can always type "encyclopedia Civil War" to find out that the Civil War falls under the category of US history. This last command would access a site which contains the "encyclopedic" class of data, i.e. superficial text on a wide variety of topics.

Queries are routed in a point-to-data fashion, rather than a point-to-point one. Instead of maintaining a routing table which tells which way to send a message bound for a particular site, a table is kept which is indexed by classes of data. Each row represents one class of data; each column represents one outbound link which could be used to forward the query. The entries themselves are values indicating the believed usefulness of each link for locating the class of data in question. When a response message passes through which contains a particular class of data, the row in the routing table for that class of data is usually updated to reflect the fact that a source of that data class can be reached in the direction from which the message arrived. In some cases, the row will be updated to point in the direction in which the message is forwarded with the belief that a cached replica of the data may be found in that direction.

Each time a row in the routing table is updated, the existing values are "aged" so that the newer information takes precedence. However, if a query which is forwarded comes back "empty handed," the older information will be used as the next most likely alternative is tried. The old information also comes in handy if a query arrives over the link which was thought to lead to the desired information. Sites avoid sending queries back to sites which have already seen them.

6.3 Cooperative Caching

Cooperative caching is distinguished from "selfish" caching by the fact that sites will cache data which do not directly benefit those sites. If one particular site is generating a huge volume of queries, the surrounding sites through which that site's queries are forwarded will accumulate copies of the data which that site is using. Idle sites donate their cache space to further the common good.

It is not easy to arrange this kind of behavior in a fully distributed manner without resorting to expensive measures. However, it can be done inexpensively using a heuristic decision function that determines whether or not a site will cache a copy of a datum which it receives. By maintaining counters in response messages, the number of hops taken and the amount of link cost incurred by a response message between the last site which had a replica of the datum in the response and the current site can be determined. Similarly, it is possible to keep track of the amount of cache space owned by the intervening sites. Using this information along with the information available locally, i.e. the costs of the links attached to this site, the following decision function can be implemented:

$$\text{rnd} < \frac{\text{Hops so far}}{\text{Hops so far} + \text{tcf}} \times \frac{\text{Running link cost}}{\text{tcf} \sum \text{neighboring link costs}} \times \frac{\text{Running cache sum}}{\text{tcf}}$$

"Rnd" represents a random number in the interval [0,1). Nondeterminism is used to prevent anomalies such as global looping behavior from persisting if they ever arise. "Tcf" is the Turnover Control Factor for the current site. Each site maintains its own tcf in an attempt to maintain a constant level of cache turnover. Turnover can be estimated as an incrementally smoothed function of the difference between the current time and the hint-adjusted timestamps of the items being replaced in cache. If necessary, a different tcf can be used in each of the three factors in the decision function to provide more control.

The first factor, hops over hops plus tcf, discourages the creating of replicas when the last observed replica is very close. The second factor, running link cost over tcf times the sum of the neighboring link costs, encourages replicas to be created when the cost of reaching existing replicas becomes great. The neighboring link costs are used to get an idea of how large the costs incurred by the response message are relative to an average link cost for the local area. The last factor, running cache sum over tcf, keeps a site from being stingy with its cache space when it has plenty of it.

Cache replacement is done using a modified Least Recently Used scheme which allows cached replicas to have hint values[Gla86] which are added to their last access times to bias the replacement. This feature is used to make it less likely that a replica of a datum which is expensive to requery will be replaced than a replica whose source is nearby.

If the decision function is not triggered by a response message, all is not lost. If the site happens to have enough idle cache space to create a replica without replacing any existing replicas, it can create a replica with a low hint value. That way, all available cache space can be utilized without jeopardizing replicas which are more important. Naturally, if the new replica is reused, its hint value is upgraded. Note that idle cache space results from fragmentation since data objects are cached as atomic units.

6.4 Simulation Results

The query routing and caching methods have been studied and developed with the aid of a simulation. The simulation has gone through several generations as the methods have been revised and assumptions have been generalized. Overall, the results indicate that an efficient distributed information system is feasible and could achieve a level of performance which averages around 40% of that which could be achieved optimally, where the distribution of data replicas would be perfect and every query would be routed directly to the nearest available replica.

To give meaning to the results which follow, several terms must be defined. The term "link delay" will indicate the time consumed by transmitting a single message from one site to another over a link. The term "response delay" will indicate the total time consumed in the transmittal of a query and its response through any number of links. The "connectivity" of a sub-network, or cluster, is defined as the number of links between nodes in the cluster divided by the number of links required for the cluster to be fully connected.

6.4.1 First Generation

In the first generation, a cluster of identical sites in which all links had costs in the same order of magnitude was simulated. A simple replica placement strategy was used which depended only on the accumulated link cost since the last observed replica on the return path. All sites had the same size cache, and the set of primary copies did not grow once it was created. Data were classified statically based on the sites at which their primary copies lay. As was the case with all the simulations, the system was started with empty caches and empty routing tables, allowing them to self-stabilize over time. Network topologies were generated at random, testing everything from lines and trees to fully connected clusters. What happened was that the caching and routing heuristics interacted in such a way that a stable arrangement of data replicas was created, and correspondingly stable routing tables guaranteed that queries were routed to nearby replicas. As link cost were perturbed, the system made minor adaptations to continue using the minimal cost paths.

The main focus of the first generation was to establish that routing and caching heuristics could be found which would achieve the authors' goals. Most of the data collected in this early simulation study consisted of traces of the routes taken by messages and descriptions of the topologies through which the messages were being routed. A representative excerpt from such a trace is shown in Figure 1. The failure of the point-to-point router to use the shortest path (the path of least delay, to

Figure 1: Early Algorithm Development

```
Query path from 16 (3.091741):  16 25 24 1 6 18 2
Shortest from 2 to 16 (1.286241):  2 18 20 15 16
Actual from 2 to 16 (1.350579):  2 18 20 4 12 26 5 7 16
Time is now 615
```

be precise) is a "feature" which was added later to generate more realistic results; an implemented router is only likely to approximate the shortest path since the actual delays will be constantly changing.

6.4.2 Second Generation

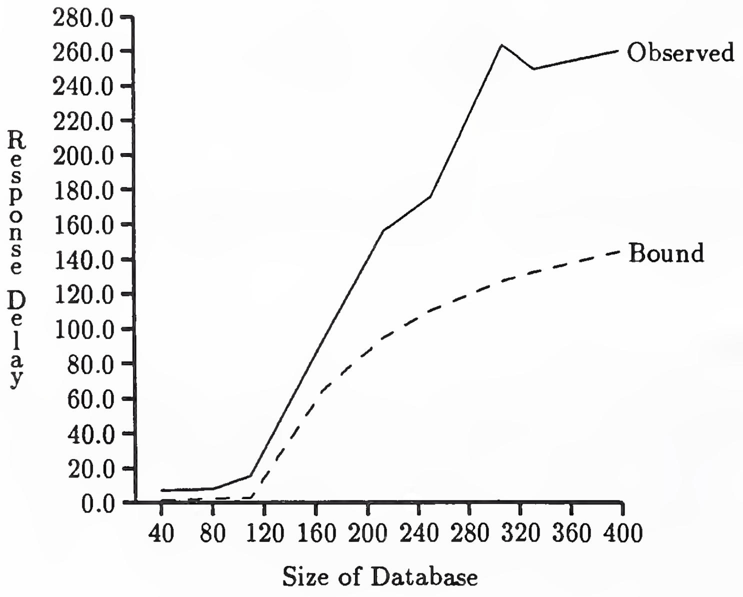
In the second generation the study began to concentrate on networks in which there were two classes of links, namely intercluster and intracluster links. A number of clusters connected by low-cost links were generated, then pairs of clusters were connected by choosing random sites within each cluster to host the high-cost links. This topology has been investigated in detail firstly because it simulates the structure of real-world networks, and secondly because the first generation algorithms had the weakness of making too much use of expensive intercluster links. The second generation algorithms made exceptions to deal with radically expensive links, and the result was that sites within a cluster succeeded in cooperatively caching all the information which was needed rather than repeatedly fetching it from overseas, as it were.

The second generation simulation took the number of data (primary copies), the minimum datum size, the maximum datum size, the mean datum size, the number of sites, the number of clusters, and the connectivity as inputs. The set of authoritative primary copies would be generated randomly at the start of each run, but remain fixed thereafter. The network topology was created in a similar manner, but intracluster link delays were perturbed during the run to simulate changing network conditions. Later simulations would perturb all link delays.

As has been the case with all the simulations, the sizes of data were generated using a Poisson distribution. The actual average datum size is usually higher than the specified average due to the removal of data of size 0 from the distribution. Since the simulation does not try to estimate the overhead involved in caching data, each site could cache all the null data without incurring any cost; it is therefore pointless to include them in the simulation.

Figure 2 shows the results of increasing the number of data in the system while keeping the number of nodes constant. The simulation reported periodically on the mean of the response delays taken over just those queries which were issued since the previous report. The curve labeled "Bound" is an optimistic lower bound on response delay. The authors were encouraged by the resemblance between the curves formed by the observed data and the optimal bound. The response delays shown in the graph are the means of all the values which were reported after steady state was reached. A minimum connectivity of 0.4, an average datum size of 2, and a cache size of 10 were used. Intracluster links ranged in cost from 0 to 2; intercluster links ranged from 50 to 200[FY92b].

Figure 2: Response Delay vs. Database Size

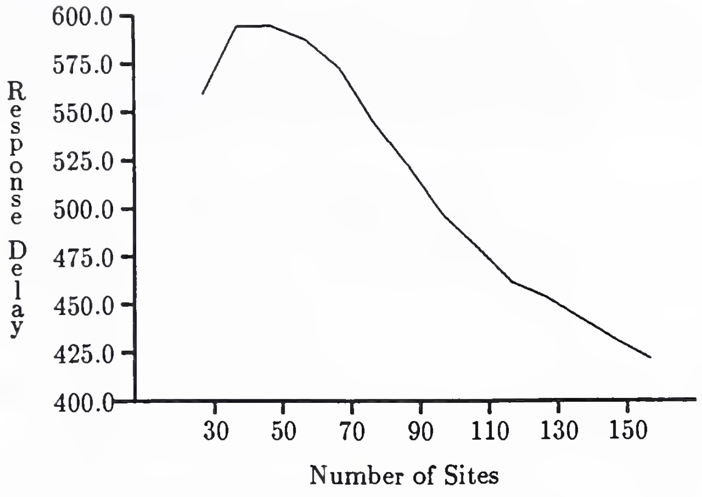


6.4.3 Third Generation

The third generation algorithms improved on the second generation algorithms to the extent that most special case exceptions were removed. The simulation began growing the database dynamically over time and placing the primary copies of data at random sites within the cluster designated as the home cluster for that class of data. The static classification scheme induced by this arrangement is "lazier" than the previous scheme, which assigned a different class to data with different primary sites, since it is coarser and provides less precise information to the query router. Furthermore, sites were allowed to have arbitrary sized caches, including null caches. The realization that it is necessary maintain a consistent level of cache turnover across sites for cooperative algorithms to be at their best was the main inspiration for this generation and is primarily responsible for the generalization of the model. A series of tests was run to try to determine the best level of turnover to maintain, only to discover that the effect of varying turnover on efficiency is partially periodic, making it difficult to analyze numerically. A theoretical analysis of this effect is on the agenda for future work.

Using the new caching heuristic and the new "lazy" classification scheme, a study similar to the second generation study was performed. This time, the sizes of sites' caches were randomized as well. Data were collected for a series of scenarios in which an ever increasing number of sites was arranged into ten clusters. Ten basic data classes were used, but their home cluster assignments were allowed to overlap. A minimum connectivity of 0.8, an average datum size of 5, and an average cache size of 25 (varying uniformly from 0 to 50) were used throughout. Intracluster links ranged in cost from 0 to 2; intercluster links ranged from 50 to 200. All sites were configured to attempt

Figure 3: Sites vs. Response Delay



to maintain turnover close to 600, meaning that the sites adjusted their caching to keep the average hint-adjusted age of data objects being uncached around 600 time units.

Figure 3 shows that response delay decreases as nodes are added. The delays used were those experienced with a constant influx of new data. Figure 4 shows the progress of response delay as the database is created, the system stabilizes, and then the long onslaught of new data begins. Note how delay is initially high since routing tables and caches start out empty, rapidly attains its steady state minimum, then as new data are constantly introduced, again begins to level off. The noise on the right side of the graph results from there being less simulation runs which took that long to complete. Finally, Figure 5 shows that the performance remains constant as the network becomes larger[FY93a]. Performance is the quotient of the calculated optimal bound on response delay and the observed response delay; constant performance indicates that response delay changed at the same rate as the optimal bound.

6.5 Implementation

Using the methods developed during the simulation study, implementation of a networked information system will soon begin. The "prototype" will be constructed so that an incremental growth of the prototype will yield a fully implemented system for public use.

Although it will not take long to explain, the implementation of the system will require a significant amount of work. The heterogeneity of the information bases will necessitate the construction of several mediators, not the least of which will be a package capable of automatically utilizing anonymous FTP to access known Internet archives. Until such archives "join" the information system themselves, a member site must act as surrogate owner of each archive.

The information system will be built in two layers. The top layer will handle the routing of

Figure 4: Time (DB Growth) vs. Response Delay

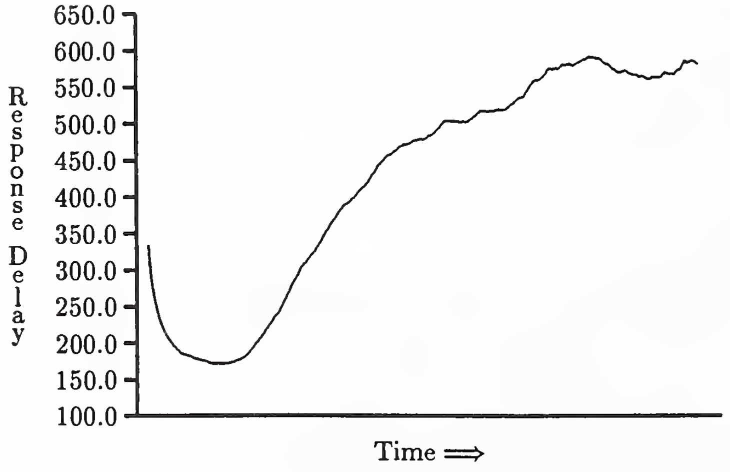
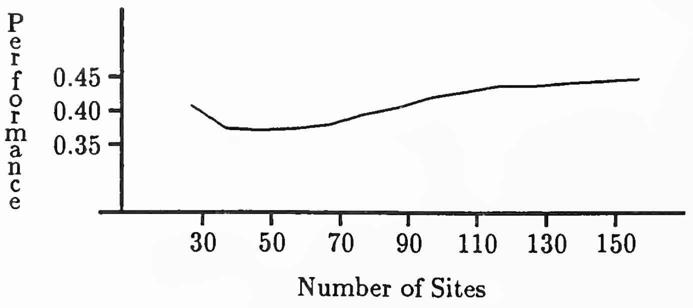


Figure 5: Sites vs. Performance



/

queries and the caching of data in a distributed manner. It will contain coarse grained classification software to determine the best way to route a query and general descriptor matching heuristics to decide when to try to answer a query with cached data. The bottom layer will consist of mediators and the local archive managers at each site owning an archive. It may be formed by an existing DBMS or by a specialized archive manager. Each will contain some combination of fine grained data classification, query processing, automatic indexing, and archive maintenance software. The bottom layer will be characterized more precisely in the following section.

6.5.1 Data Types and Classification Heuristics

When the prototype comes on line, it should include at least the following in order to provide a demonstration of its capacity to handle heterogeneity:

1. A corpus of ASCII text documents;
2. a collection of source code in several different programming languages;
3. a database of relational tables or other objects which is indirectly accessed through a DBMS;
4. a collection of bitmapped images with differing formats;
5. an FTP archive containing executables and other data.

Most of the workstations in the system should at least be able to determine which major data type a query requests. This coarse grained classification will be accomplished by inspection of the query for key terms. For example, source code would be flagged by the presence of the names of one or more programming languages or the word "source" in a favorable context. To avoid repetition of coarse grained classification, the general data type required by a query will be coded by the client at the querying site whenever possible. The most common data types will be assigned special codes. Queries which cannot be parsed in this way will be forwarded without the special type coding.

Finer grained classification for each major data type will be accomplished as follows:

1. WAIS should serve as the main textual archive manager. Any additional text documents will be analyzed and indexed using an existing text processing algorithm, and matches between queries and documents will be made using an existing fuzzy retrieval method. Boolean querying may also be supported.
2. Source code will be classified by language and by purpose. The language is easily determined by scanning the file for constructs unique to different languages. The purpose of each source file will be kept in an index which is initially built automatically from comments found in the source, but modified as necessary by a human operator. Indexing solely by scanning comments is unreliable since commenting styles vary drastically from programmer to programmer, some of which may not comment at all. Matches between queries and index entries will be done as before.

3. When a DBMS is involved, an information system query might encapsulate a query to be presented to the DBMS or it might contain an unstructured question as usual. The encapsulated query case is trivial since it only needs to be identified as such; this is on the same level as determining the language of a source code file. However, unstructured questions must be translated into a query which is meaningful to the DBMS. Finding ways of translating arbitrary natural language queries into a database query language is beyond the scope of this work, as research is still actively being done in this area. However, a simple system may be implemented by which a user may ask how to access on line help for the specific DBMS and for the names and attributes of database objects which contain data relevant to some topic. It should also be possible to create a simple macroscopic mapping of database objects onto information system objects, such as to consider an entire relational table to be a single object[FY92a, BC92a]. This simplifies the interface enough that the information system should be able to reliably construct queries for the desired objects and to cache replicas of them as usual. The drawback is that the user must accept the object in its entirety and send an encapsulated query to the DBMS if more selectivity is desired – but the simplified system is much more likely to succeed than an attempt to translate natural language into arbitrarily complex queries.

4. The format of an image can almost always be determined by scanning the beginning of the image file for signatures which are unique to different formats. As was discussed earlier, automatic determination of the content of images is just beginning to be actively researched and is in its extremely early stages. With the exception of automatically generated images such as satellite weather maps, the collections of images the system includes will almost certainly be manually indexed.

5. Many FTP archives provide informal indices for their available files. The system will interface with these archives by utilizing the existing textual indices and anonymous FTP in a fully automatic manner.

While it is small, this initial set of interfaces and data managers will provide a demonstration of the capabilities of the system. The FTP interface alone will provide a much needed service, removing the need to navigate unfamiliar file systems and reducing the drain on central archive sites and the network through cooperative caching.

7 Concluding Remarks

While the information retrieval field was growing, the need for good information retrieval techniques was growing even faster. Some good information retrieval techniques are now available for small domains – but the current information domain is vastly larger than any of these techniques can handle. Just as small-scale information systems are becoming capable of dealing with natural language queries, the major information networks have grown so large that even a skilled user is sometimes unable to locate needed information.

To find a desirable solution to the Resource Discovery Problem, it is necessary to adapt and integrate information retrieval techniques for the large network environment. The fragmented and heterogeneous resources of the network must be used to support distributed information retrieval in a fair and efficient manner. Specialized information retrieval techniques must be allowed to operate within limited domains while researchers work to solve the greater problem of generalized retrieval. In this chapter, algorithms and architecture were presented for a distributed information system which can solve the following problems:

- The Resource Discovery Problem. By automatically routing queries to the sites which can answer them, the system removes the burden of resource discovery from the user.
- The heterogeneity problem. By treating all data as objects, it supports the integration of any information base whose data can be divided into units. By giving access to all these information bases simultaneously, a simplified global view is provided.
- The retrieval problem. Information retrieval techniques and data classification can easily be used by the system to handle queries expressed in natural language.

Acknowledgements

Information Retrieval is indeed a very rich field which has been under investigation for many years. The authors would like to recognize the very many researchers who worked long and hard to bring Information Retrieval to the level it has attained. They sincerely apologize to all those who could not be recognized due to space limitations.

References

[BC92a] Daniel Barbará and Chris Clifton. Information brokers: Sharing knowledge in a heterogeneous distributed system. Technical Report MITL-TR-31-92, Matsushita Information Technology Laboratory, 182 Nassau Street, Princeton, NJ 08542, October 1992.

[BC92b] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, December 1992.

[BCP91] G. Bordogna, P. Carrara, and G. Pasi. Query term weights as constraints in fuzzy information retrieval. *Information Processing & Management*, 27(1):15–26, 1991.

[CEMW91] Janey K. Cringean, Roger England, Gordon A. Manson, and Peter Willett. Network design for the implementation of text searching using a multicomputer. *Information Processing & Management*, 27(4):265–283, 1991.

[Cro90] C. J. Crouch. An approach to the automatic construction of global thesauri. *Information Processing & Management*, 26(5):629–640, 1990.

[CW92] Chin-Chen Chang and Tzong-Chen Wu. Retrieving the most similar symbolic pictures from pictorial databases. *Information Processing & Management*, 28(5):581–588, 1992.

[Deu92] Peter Deutsch. Resource discovery in an internet environment–the Archie approach. *Electronic Networking*, 2(1):45–51, Spring 1992.

[DRST91] James R. Driscoll, David A. Rajala, William H. Shaffer, and Donald W. Thomas. The operation and performance of an artificially intelligent keywording system. *Information Processing & Management*, 27(1):43–54, 1991.

[FY92a] David W. Flater and Yelena Yesha. An efficient management of read-only data in a distributed information system. Technical Report CS-92-04, University of Maryland Baltimore County, Baltimore, MD 21228, March 1992.

[FY92b] David W. Flater and Yelena Yesha. Query routing and object caching in a large distributed information system. In *Proceedings of the ISMM First International Conference on Information and Knowledge Management*, pages 525–534, Baltimore, MD, U.S.A., November 1992. The International Society for Mini and Microcomputers.

[FY93a] David W. Flater and Yelena Yesha. An efficient management of read-only data in a distributed information system. *International Journal of Intelligent and Cooperative Information Systems, Special Issue on Information and Knowledge Management*, 1993. To appear.

[FY93b] David W. Flater and Yelena Yesha. An information retrieval system for network resources. In *Proceedings of the International Workshop on Next Generation Information Technologies and Systems*, 1993. To appear.

[FY93c] David W. Flater and Yelena Yesha. Properties of networked information retrieval with cooperative caching. Technical Report CS-93-08, University of Maryland Baltimore County, Baltimore, MD 21228, April 1993.

[FY93d] David W. Flater and Yelena Yesha. A robust and efficient strategy for the distributed caching of read-only data in a large networked information system. *International Journal of Intelligent and Cooperative Information Systems*, 1993. Submitted.

[Gla86] Henry M. Gladney. A model for distributed information networks. Technical Report RJ5220, IBM Almaden Res. Lab, 650 Harry Road, San Jose, California 95120-6099, July 1986.

[Gol] Jonny Goldman. In response to questions sent through e-mail.

[HMTC91] Donna Harman, Wayne McCoy, Robert Toense, and Gerald Candela. Prototyping a distributed information retrieval system that uses statistical ranking. *Information Processing & Management*, 27(5):449–460, 1991.

[Hob] Jerry Hobbs. Presentation on the use of a finite state machine (FASTUS) to perform information extraction from natural language texts given in a panel session at the First International Conference on Information and Knowledge Management.

[HVB90] G. David Huffman, Dennis A. Vital, and Royal G. Bivins, Jr. Generating indices with lexical association methods: Term uniqueness. *Information Processing & Management*, 26(4):549–558, 1990.

[Kah91] Brewster Kahle. think.com:/public/wais/README, September 1991.

[MBK91] Yoëlle S. Maarek, Daniel M. Berry, and Gail E. Kaiser. An information retrieval approach for automatically constructing software libraries. *IEEE Transactions on Software Engineering*, 17(8):800–813, August 1991.

[MK87] Arnold C. Meltzer and Gerald Kowalski. Text searching using an inversion database consisting of trigrams. In *IEEE Proceedings of the Second International Conference on Computers and Applications*, pages 65–69, 1987.

[MN92] James Mayfield and Charles Nicholas. SNITCH: Augmenting hypertext documents with a semantic net. In *Proceedings of the ISMM First International Conference on Information and Knowledge Management*, pages 146–152, Baltimore, MD, U.S.A., November 1992. The International Society for Mini and Microcomputers.

[Pai90] Chris D. Paice. Constructing literature abstracts by computer: Techniques and prospects. *Information Processing & Management*, 26(1):171–186, 1990.

[Pai91] Chris D. Paice. A thesaural model of information retrieval. *Information Processing & Management*, 27(5):433–447, 1991.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[Rat92] Lee Ratzan. Building an internet browser. *UNIX Review*, 10(1):25–, January 1992.

[Rob81] S. E. Robertson. The methodology of information retrieval experiment. In K. Sparck Jones, editor, *Information Retrieval Experiment*, pages 9–31. Butterworths, 1981.

[RS92] F. Rabitti and P. Savino. Automatic image indexation to support content-based retrieval. *Information Processing & Management*, 28(5):547–565, 1992.

[Rug92] Gerda Ruge. Experiments on linguistically-based term associations. *Information Processing & Management*, 28(3):317–332, 1992.

[SB88] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(3):513–524, 1988.

[Sch88] Michael F. Schwartz. The networked resource discovery project: Goals, design, and research efforts. Technical Report CU-CS-387-88, University of Colorado, Boulder, Colorado 80309, May 1988.

[Sch90] Michael F. Schwartz. A scalable, non-hierarchical resource discovery mechanism based on probabilistic protocols. Technical Report CU-CS-474-90, University of Colorado, Boulder, Colorado 80309, June 1990.

[SHHH91] Michael F. Schwartz, Darren R. Hardy, William K. Heinzman, and Glenn C. Hirschowitz. Supporting resource discovery among public internet archives using a spectrum of information quality. In *11th International Conference on Distributed Computing Systems*, May 1991.

[SM83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[TC91] H. R. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 3:187–222, 1991.

[Wie92] Gio Wiederhold. Intelligent integration of diverse information. In *Proceedings of the ISMM First International Conference on Information and Knowledge Management*, pages 1–7, Baltimore, MD, U.S.A., November 1992. The International Society for Mini and Microcomputers.

[WL90] Wai Yee Peter Wong and Dik Lun Lee. Signature file methods for implementing a ranking strategy. *Information Processing & Management*, 26(5):641–653, 1990.