Chapter 1

DISTRIBUTED INFORMATION RETRIEVAL

Jamie Callan
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
callan@cs.cmu.edu

Abstract A multi-database model of distributed information retrieval is presented, in which people are assumed to have access to many searchable text databases. In such an environment, full-text information retrieval consists of discovering database contents, ranking databases by their expected ability to satisfy the query, searching a small number of databases, and merging results returned by different databases. This paper presents algorithms for each task. It also discusses how to reorganize conventional test collections into multi-database testbeds, and evaluation methodologies for multi-database experiments. A broad and diverse group of experimental results is presented to demonstrate that the algorithms are effective, efficient, robust, and scalable.

1. INTRODUCTION

Wide area networks, particularly the Internet, have transformed how people interact with information. Much of the routine information access by the general public is now based on full-text information retrieval, as opposed to more traditional controlled vocabulary indexes. People have easy access to information located around the world, and routinely encounter, consider, and accept or reject information of highly variable quality.

Search engines for the Web and large corporate networks are usually based on a *single database* model of text retrieval, in which documents from around the network are copied to a centralized database, where it is indexed and made searchable. The single database model can be successful if most of the important or valuable information on a network can be copied easily. However, information that cannot be copied is not accessible under the single database model. Information that is proprietary, that costs money, or that a publisher wishes to control carefully is essentially invisible to the single database model.

The alternative to the single database model is a *multi-database* model, in which the existence of multiple text databases is modeled explicitly. A central site stores brief descriptions of each database, and a *database selection* service uses these *resource descriptions* to identify the database(s) that are most likely to satisfy each information need. The multi-database model can be applied in environments where database contents are proprietary or carefully controlled, or where access is limited, because the central site does not require copies of the documents in each database. In principle, and usually in practice, the multi-database model also scales to large numbers of databases.

The multi-database model of information retrieval reflects the distributed location and control of information in a wide area computer network. However, it is also more complex than the single database model of information retrieval, requiring that several additional problems be addressed:

Resource description: The contents of each text database must be described;

Resource selection: Given an information need and a set of resource descriptions, a decision must be made about which database(s) to search; and

Results merging: Integrating the ranked lists returned by each database into a single, coherent ranked list.

This set of problems has come to be known as *Distributed Information Retrieval.*

One problem in evaluating a new research area such as distributed IR is that there may be no accepted experimental methodologies or standard datasets with which to evaluate competing hypotheses or techniques. The creation, development, and evaluation of experimental methodologies and datasets is as important a part of establishing a new research area as the development of new algorithms.

This paper presents the results of research conducted over a five year period that addresses many of the issues arising in distributed IR systems. The paper begins with a discussion of the multi-database datasets that were developed for testing research hypotheses. Section 3 addresses the problem of succinctly describing the contents of each available resource or database. Section 4 presents an algorithm for ranking databases by how well they are likely to satisfy an information need. Section 5 discusses the problem of merging results returned by several different search systems. Section 6 investigates how a distributed IR system acquires *resource descriptions* for each searchable text database in a multi-party environment. Finally, Section 7 summarizes and concludes.

2. MULTI-DATABASE TESTBEDS

Research on distributed IR can be traced back at least to Marcus, who in the early 1980's addressed resource description and selection in the EXPERT CONIT system, using expert system technology (Marcus, 1983). However, neither Marcus nor the rest of the research community had access to a sufficiently large experimental testbed with which to study the issues that became important during the 1990's: How to create solutions that would scale to large numbers of resources, distributed geographically, and managed by many parties.

The creation of the TREC corpora removed this obstacle. The text collections created by the U.S. National Institute for Standards and Technology (NIST) for its TREC conferences (Harman, 1994; Harman, 1995) were sufficiently large and varied that they could be divided into smaller databases that were themselves of reasonable size and heterogeneity. NIST also provided relevance judgements based on the results of running dozens of IR systems on queries derived from well-specified information needs.

The first testbed the UMass Center for Intelligent Information Retrieval (CIIR) produced for distributed IR research was created by dividing three gigabytes of TREC data (NIST CDs 1, 2, and 3) by source and publication date (Callan et al., 1995b; Callan, 1999b). This first testbed contained 17 text databases that varied widely in size and characteristics (Table 1.1) (Callan, 1999b). The testbed was convenient to assemble and was an important first step towards gaining experience with resource description and selection. However, it contained few databases, and several of the databases were considerably larger than the databases found in many "real world" environments.

Several testbeds containing $O(100)$ smaller databases were created to study resource selection in environments containing many databases. All were created by dividing TREC corpora into smaller databases, based on source and publication date. One representative example was the testbed created for TREC-5 (Harman, 1997), in which data on TREC CDs 2 and 4 was partitioned into 98 databases, each about 20 megabytes

Table 1.1 Summary statistics for three distributed IR testbeds.

Number of Databases	*Source*	*Number of Documents* *Min*	*Avg*	*Max*	*Megabytes* *Min*	*Avg*	*Max*
17	TREC CDs 1,2,3	6,711	64,010	226,087	35	196	362
100	TREC CDs 1,2,3	752	10,782	39,723	28	33	42
921	TREC VLC	12	8,157	31,703	1	23	31

in size. Testbeds of about 100 databases each were also created based on TREC CD's 1 and 2 (Xu and Callan, 1998), TREC CD's 2 and 3 (Lu et al., 1996a; Xu and Callan, 1998), and TREC CD's 1, 2, and 3 (French et al., 1999; Callan, 1999a).

A testbed of 921 databases was created by dividing the 20 gigabyte TREC Very Large Corpus (VLC) data into smaller databases (Callan, 1999c; French et al., 1999). Each database contained about 23 megabytes of documents from a single source (Table 1.1), and the ordering of documents within each database was consistent with the original ordering of documents in the TREC VLC corpus. This testbed differed from other, smaller testbeds not only in size, but in composition. 25% of the testbed (5 gigabytes) was traditional TREC data, but the other 75% (15 gigabytes) consisted of Web pages collected by the Internet Archive project in 1997 (Hawking and Thistlewaite, 1999). The relevance judgements were based on a much smaller pool of documents retrieved by a much smaller group of IR systems, thus results on that data must be viewed more cautiously.

Although there are many differences among the testbeds, they share important characteristics. Within a testbed, database sizes vary, whether measured by number of documents, number of words, or number of bytes. Databases in a testbed are more homogeneous than the testbed as a whole, which causes some corpus statistics, for example, inverse document frequency (*idf*), to vary significantly among databases. Databases also retain a certain degree of heterogeneity, to make it more difficult to distinguish among them. These characteristics are intentional; they are intended to reduce the risk of accidental development of algorithms that are sensitive to the quirks of a particular testbed. As a group, this set of distributed IR testbeds enabled an unusually thorough investigation of distributed IR over a five year period.

Others have also created resource selection testbeds by dividing the TREC data into multiple databases, usually also partitioning the data along source and publication date criteria, for example (Voorhees et al., 1995b; Viles and French, 1995; Hawking and Thistlewaite, 1999; French et al., 1998). Indeed, there are few widely available alternative sources of data for creating resource selection testbeds. The alternative data used most widely, created at Stanford as part of research on the GlOSS and gGlOSS resource selection algorithms (Gravano et al., 1994; Gravano and García-Molina, 1995), is large and realistic, but does not provide the same breadth of relevance judgements.

3. RESOURCE DESCRIPTION

The first tasks in an environment containing many databases is to discover and represent what each database contains. Discovery and representation are closely related tasks, because the method of discovery plays a major role in determining what can be represented. Historically representation was addressed first, based on a principle of deciding first what is desirable to represent, and worrying later about how to acquire that information.

Resource descriptions vary in their complexity and in the effort required to create them. CIIR research was oriented towards environments containing many databases with heterogeneous content. Environments containing many databases, and in which database contents may change often, encourage the use of resource descriptions that can be created automatically. Resource descriptions that must be created and updated manually (e.g., Marcus, 1983; Chakravarthy and Haase, 1995) or that are learned from manual relevance judgements (e.g., Voorhees et al., 1995a) might be difficult or expensive to apply in such environments.

Environments containing heterogeneous databases also favor detailed resource descriptions. For example, to describe the Wall Street Journal as a publication of financial and business information ignores the large amount of information it contains about U.S. politics, international affairs, wine, and other information of general interest.

A simple and robust solution is to to represent each database by a description consisting of the words that occur in the database, and their frequencies of occurrence (Gravano et al., 1994; Gravano and García-Molina, 1995; Callan et al., 1995b) or statistics derived from frequencies of occurrence (Voorhees et al., 1995a). We call this type of representation a *unigram language model*. Unigram language models are compact and can be obtained automatically by examining the documents in a database or the document indexes. They also can can be extended easily to include phrases, proper names, and other text features that occur in the database.

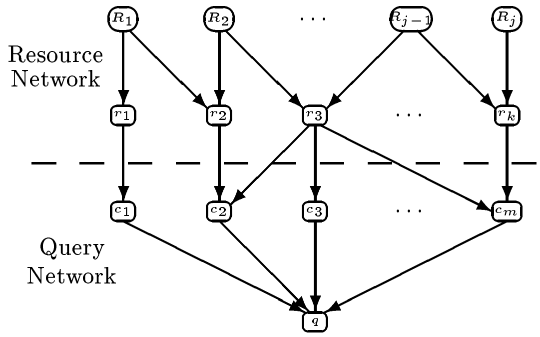


Figure 1.1 A simple resource selection inference network.

Resource descriptions based on terms and their frequencies are generally a small fraction of the size of the original text database. The size is proportional to the number of unique terms in the database. Zipf's law indicates that the rate of vocabulary growth decreases as database size increases (Zipf, 1949), hence the resource descriptions for large databases are a smaller fraction of the database size than the resource descriptions for small databases.

4. RESOURCE SELECTION

Given an information need and a set of resource descriptions, how is the system to select which resources to search? The major part of this *resource selection* problem is ranking resources by how likely they are to satisfy the information need. Our approach is to apply the techniques of document ranking to the problem of resource ranking, using variants of *tf.idf* approaches (Callan et al., 1995b; Lu et al., 1996a). One advantage is that the same query can be used to rank resources and to rank documents.

The Bayesian Inference Network model of Information Retrieval can be applied to the process of ranking resources, as illustrated by Figure 1.1. Each resource R_i is represented by a set of representation nodes (indexing terms) r_j. An information need is represented by one or more queries (q), which are composed of query concepts (c_k) and query operators (not shown in this simple example).

The belief $P(q|R_i)$ that the information need represented by query q is satisfied by searching resource R_i is determined by instantiating node R_i and propagating beliefs through the network towards node q. The belief $P(r_j|R_i)$ that the representation concept r_j is observed given resource R_i is estimated by a variation of *tf.idf* formulas, shown below.

$$T = \frac{df}{df + 50 + 150 \cdot cw/avg\_cw} \tag{1.1}$$

$$I = \frac{\log\left(\frac{C+0.5}{cf}\right)}{\log\left(C + 1.0\right)} \tag{1.2}$$

$$p(r_k|c_i) = b + (1 - b) \cdot T \cdot I \tag{1.3}$$

where:

df	is the number of documents in R_i containing r_k,
cw	is the number of indexing terms in resource R_i,
avg_cw	is the average number of indexing terms in each resource,
C	is the number of resources,
cf	is the number of resources containing term r_k, and
b	is the minimum belief component (usually 0.4).

Equation 1.1 is a variation of Robertson's term frequency (tf) weight (Robertson and Walker, 1994), in which term frequency (tf) is replaced by document frequency (df), and the constants are scaled by a factor of 100 to accommodate the larger df values (Callan et al., 1995b). Equation 1.2 is a variation of Turtle's scaled idf formula (Turtle, 1990; Turtle and Croft, 1991), in which number of documents is replaced by number of resources (C).

Equations 1.1-1.3 have come to be known as the *CORI* algorithm for ranking databases (French et al., 1998; French et al., 1999; Callan et al., 1999b), although the name *CORI* was originally intended to apply more broadly, to any use of inference networks for ranking databases (Callan et al., 1995b).

The scores $p(r_j|R_i)$ accruing from different terms r_j are combined according to probabilistic operators modeled in the Bayesian inference network model. INQUERY operators are discussed in detail elsewhere (Turtle, 1990; Turtle and Croft, 1991), so only a few common operators are presented here. The belief $p(r_j|R_i)$ is abbreviated p_j for readability.

$$\mathrm{bel}_{\mathrm{sum}}(Q) = \frac{(p_1 + p_2 + \ldots + p_n)}{n} \tag{1.4}$$

$$\mathrm{bel}_{\mathrm{wsum}}(Q) = \frac{(w_1 p_1 + w_2 p_2 + \ldots + w_n p_n) w_q}{(w_1 + w_2 + \ldots + w_n)} \tag{1.5}$$

$$\mathrm{bel}_{\mathrm{not}}(Q) = 1 - p_1 \tag{1.6}$$

$$\mathrm{bel}_{\mathrm{or}}(Q) = 1 - (1 - p_1) \cdot \ldots \cdot (1 - p_n) \tag{1.7}$$

$$\mathrm{bel}_{\mathrm{and}}(Q) = p_1 \cdot p_2 \cdot \ldots \cdot p_n \tag{1.8}$$

Most INQUERY query operators can be used, without change, for ranking both databases and documents. The exceptions are proximity, passage, and synonym operators (Callan et al., 1995b), all of which rely on knowing the locations of each index term in each document. Such information is not included in database resource descriptions due to its size, so these operators are all coerced automatically to a Boolean AND operator. Boolean AND is a weaker constraint than proximity, passage, and synonym operators, but it is the strongest constraint that can be enforced with the information available.

The effectiveness of a resource ranking algorithm can be measured with $R(n)$, a metric intended to be analogous to the recall metric for document ranking. $R(n)$ compares a given database ranking at rank n to a desired database ranking at rank n. The desired database ranking is one in which databases are ordered by the number of relevant documents they contain for a query (Gravano and García-Molina, 1995; Lu et al., 1996b; French et al., 1998). $R(n)$ is defined for a query as follows.

$$R(n) = \frac{\sum_{i=1}^{n} rg_i}{\sum_{i=1}^{n} rd_i} \tag{1.9}$$

rg_i : number of relevant documents in the i'th-ranked database under the *given* ranking

rd_i : number of relevant documents in the i'th-ranked database under a *desired* ranking in which documents are ordered by the number of relevant documents they contain

$R(n)$ measures how well an algorithm ranks databases containing many relevant documents ahead of databases containing few relevant documents.

The CORI database ranking algorithm was tested in a series of experiments on testbeds ranging in size from $O(100)$ to $O(1,000)$ databases. Two of the testbeds were developed at the University of Massachusetts (Callan, 1999a; Callan, 1999c); one was developed at the University of Virginia (French et al., 1998). Results were measured using $R(n)$.

Figure 1.2 shows the effectiveness of the resource ranking algorithm with differing numbers of resources (French et al., 1999). The horizontal axis in these graphs is the *percentage* of the databases in the testbed that are examined or considered. For example, for all testbeds, the top 10% of the databases contain about 60% as many relevant documents as the top 10% of the databases in the desired ranking (a ranking in which databases are ordered by the number of relevant documents they contain).

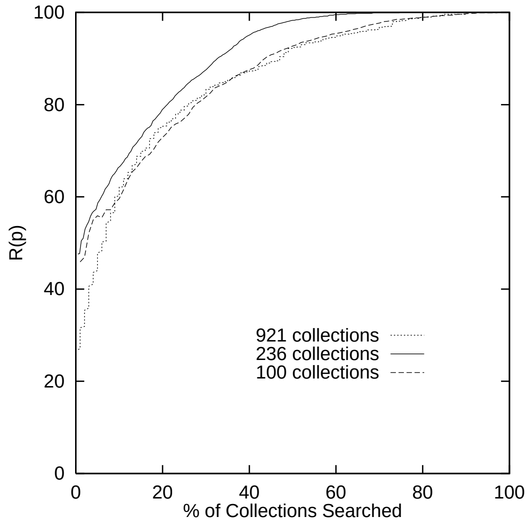


Figure 1.2 Effectiveness of the resource ranking algorithm on testbeds with differing numbers of resources.

The accuracy of the resource rankings was remarkably consistent across all three testbeds when 8–100% of the databases are to be searched. The algorithm was most effective on the testbed of 236 databases, but the differences due to testbed size were small. Greater variability was apparent when 0–8% of the databases are to be searched. In this test, accuracy on the testbed of 921 databases was significantly lower than the accuracy on the other databases. It is unclear whether this difference at low recall (searching 0-8% of the databases) is due to testbed size (100 databases vs 921 databases) or testbed content (produced professionally vs Web pages).

One issue in scaling up this research is that as more databases become available, a smaller percentage of the available data is typically searched for each query. Consequently, as the number of available databases increases, the accuracy of the ranking algorithm must also increase, or else recall will decrease significantly. Some loss of recall is inevitable when many resources contain relevant documents but only a few resources are searched.

Once a set of resources is ranked, resource selection is relatively simple. One can choose to search the top n databases, all databases with a score above some threshold value, or a set of databases satisfying some cost metric (e.g., Fuhr, 1999).

5. MERGING DOCUMENT RANKINGS

After a set of databases is searched, the ranked results from each database must be merged into a single ranking. This task can be difficult because the document rankings and scores produced by each database are based on different corpus statistics and possibly different representations and/or retrieval algorithms; they usually cannot be compared directly. Solutions include computing normalized scores (Kwok et al., 1995; Viles and French, 1995; Kirsch, 1997; Xu and Callan, 1998), estimating normalized scores (Callan et al., 1995b; Lu et al., 1996a), and merging based on unnormalized scores (Dumais, 1994).

The most accurate solution is to normalize the scores of documents from different databases, either by using global corpus statistics (e.g., (Kwok et al., 1995; Viles and French, 1995; Xu and Callan, 1998)) or by recomputing document scores at the search client (Kirsch, 1997). However, this solution requires that search systems cooperate, for example by exchanging corpus statistics, or that the search client rerank the documents prior to their display.

Our goal was a solution that required no specific cooperation from search engines, and that imposed few requirements on the search client. Our solution was to estimate normalized document scores, using only information that a resource selection service could observe directly.

Several estimation heuristics were investigated. All were based on a combination of the score of the database and the score of the document. All of our heuristics favor documents from databases with high scores, but also enable high-scoring documents from low-scoring databases to be ranked highly. The first heuristic, which was used only briefly (Callan et al., 1995b; Allan et al., 1996), is shown in Equation 1.10.

$$\begin{aligned} D'' &= D \cdot (1 + N \cdot \frac{R_i - Avg\_R}{Avg\_R}) \qquad (1.10) \\ N &: \text{Number of resources searched} \end{aligned}$$

The normalized document score D'' is the product of the unnormalized document score D and a database weight that is based on how the database score R_i compares to the average database score Avg_R.

This heuristic was effective with a few databases, but is flawed by its use of the number of databases N and the average database score Avg_R. If 100 low-scoring databases with no relevant documents are added to a testbed, N is increased and Avg_R is decreased, which can dramatically change the merged document rankings.

A second heuristic for normalizing database scores was based on the observation that the *query* constrains the range of scores that the resource ranking algorithm can produce. If T in Equation 1.1 is set to 1.0

for each query term, a score R_{max} can be computed for each query. If T is set to 0.0 for each query term, a score R_{min} can be computed for each query. These are the highest and lowest scores that the resource ranking algorithm could *potentially* assign to a database. In practice, the minimum is exact, and the maximum is an overestimate.

R_{min} and R_{max} enable database scores to be normalized with respect to the query instead of with respect to the other databases, as shown in Equation 1.11. This type of normalization produces more stable behavior, because adding databases to a testbed or deleting databases from a testbed does not change the scores of other databases in the testbed. However, it does require a slight modification to the way in which database scores and document scores are combined (Equation 1.12).

$$R_i{}' = (R_{\text{max}} - R_i)/(R_{\text{max}} - R_{\text{min}}) \tag{1.11}$$

$$D'' = \frac{D + 0.4 \cdot D \cdot R_i{}'}{1.4} \tag{1.12}$$

Equations 1.11 and 1.12 were the core of the INQUERY distributed IR system from 1995-1998. They produced very stable results for most CIIR distributed IR testbeds. However, research projects on language modeling and U.S. Patent data identified an important weakness. Databases that are organized by subject, for example by placing all of the documents about computers in one database, all of the documents about health care in another, etc, produce *idf* scores, and hence document scores, that are very highly skewed. Documents from databases where a query term is common (probably a good database for the query) tend to have low scores, due to low *idf* values. Documents from databases where a query term is rare (probably a poor database for the query) tend to have high scores, due to high *idf* values. When *idf* statistics are very highly skewed, the normalization provided by Equations 1.11 and 1.12 is insufficient.

Equations 1.14 and 1.15 solve the problem of highly skewed document scores by normalizing a document's score by the maximum and minimum document scores that could possibly be produced for the query using the corpus statistics in its database.

$$R_i{}' = (R_{\text{max}} - R_i)/(R_{\text{max}} - R_{\text{min}}) \tag{1.13}$$

$$D' = (D_{\text{max}_i} - D)/(D_{\text{max}_i} - D_{\text{min}_i}) \tag{1.14}$$

$$D'' = \frac{D' + 0.4 \cdot D' \cdot R_i'}{1.4} \tag{1.15}$$

In INQUERY, $D_{\max_i}$ for database R_i is calculated by setting the *tf* component of the *tf.idf* algorithm to its maximum value (1.0) for each query term; $D_{\min_i}$ for database R_i is calculated by setting the *tf* component of the *tf.idf* algorithm to its minimum value (0.0) for each query term. Hence $D_{\max_i}$ and $D_{\min_i}$ are estimates of the maximum and minimum scores any document in database R_i could be assigned for the given query.

Equation 1.14 solves the problem of highly skewed *idf* scores, because it is effective on testbeds with and without highly skewed *idf* scores. However, it requires cooperation among search engines, because $D_{\max_i}$ and $D_{\min_i}$ must be provided by the search engine when it returns document rankings. An independent resource ranking service cannot calculate those values itself (although it could perhaps estimate them, based on observation over time). It is our goal *not* to rely upon cooperation among search engines, because cooperation can be unreliable in multiparty environments. Thus, although this variant of the result-merging algorithm is effective, equally effective algorithms that do not require cooperation remain a research goal.

The two variants of the result-merging algorithm are suitable for different environments. The first variant, expressed in Equations 1.11-1.12, requires no cooperation from resource providers, and is effective when corpus statistics are either homogeneous or moderately skewed among databases. The second variant, expressed in Equations 1.13-1.14, is effective when corpus statistics are homogeneous, moderately skewed, and extremely skewed among databases, but it requires resource providers to cooperate by providing $D_{\max_i}$ and $D_{\min_i}$. The first variant might be appropriate on a wide area network, where cooperation cannot be enforced. The second variant might be appropriate on a local area network within a single organization.

6. ACQUIRING RESOURCE DESCRIPTIONS

Acquiring resource descriptions can be a difficult problem, especially in a wide-area nework containing resources controlled by many parties. One solution is for each resource provider to cooperate by publishing resource descriptions for its document databases. The STARTS protocol, for example, is a standard format for communicating resource descriptions (Gravano et al., 1996). Solutions that require cooperation are appropriate in controlled environments, such as a single organization, but face problems in multi-party environments such as the Internet. If a resource provider can't cooperate, or refuses to cooperate, or is deceptive, the cooperative approach fails.

Even when providers intend to cooperate, different systems, different assumptions, and different choices (e.g., how to stem words) make resource descriptions produced by different parties incomparable. For example, which database is best for the query 'Apple': A database that contains 2,000 occurrences of 'appl', a database that contains 500 occurrences of 'apple', or a database that contains 50 occurrences of 'Apple'? The answer requires detailed knowledge about the tokenizing, stopword, stemming, case conversion, and proper name handling performed by each database. Such detail is impractical to communicate, thus cooperative solutions are most appropriate in environments where all parties use the same software and the same parameter settings.

An alternative solution is for the resource selection service to *learn* what each resource contains by submitting queries and observing the documents that are returned. This technique is called *query-based sampling* (Du and Callan, 1998; Callan et al., 1999a; Callan and Connell, 1999; Callan et al., 1999b). It is based on the hypothesis that a resource description created from a small sample of text is sufficiently similar to a complete resource description. Query-based sampling requires minimal cooperation (only the ability to run queries and retrieve documents), and it makes no assumptions about how each system operates internally. It also allows different resource selection services to make different decisions about how to represent resources, encouraging development of competing approaches to resource description and selection.

Query-based sampling was tested with experiments that investigate it from several different perspectives: Accuracy of learned language models, accuracy of database rankings, and accuracy of document rankings. These experiments are discussed below.

6.1 ACCURACY OF UNIGRAM LANGUAGE MODELS

The first tests of query-based sampling studied how well the *learned* language models matched the *actual* or *complete* language model of a database. A *learned* language model is one created from documents that were obtained by query-based sampling. The *actual* or *complete* language model is one created by examining every document in the database.

Three text databases were used: CACM, 1988 Wall Street Journal, and the TREC-123 databases. CACM is a small, homogeneous database of scientific abstracts. The 1988 Wall Street Journal is a larger, het-

Table 1.2 Test corpora for query-based sampling experiments.

Name	*Size, in bytes*	*Size, in documents*	*Size, in unique terms*	*Size, in total terms*	*Variety*
CACM	2MB	3,204	6,468	117,473	homogeneous
WSJ88	104MB	39,904	122,807	9,723,528	heterogeneous
TREC-123	3.2GB	1,078,166	1,134,099	274,198,901	very heterogenenous

erogeneous database of American newspaper articles (Harman, 1994). The TREC-123 database is a large, very heterogeneous database of documents from a variety of different sources and timespans (Harman, 1994; Harman, 1995). Their characteristics are summarized in Table 1.2. All three are standard IR test databases.

Unigram language models consist of a vocabulary and term frequency information. The *ctf ratio* measures how well the learned vocabulary matches the actual vocuabulary. The Spearman Rank Correlation Coefficient measures how well the learned term frequencies indicates the frequency of each term in the database.

Ctf ratio is the proportion of term occurrences in the database that are covered by terms in the learned resource description. For a learned vocabulary V' and an actual vocabulary V, *ctf* ratio is:

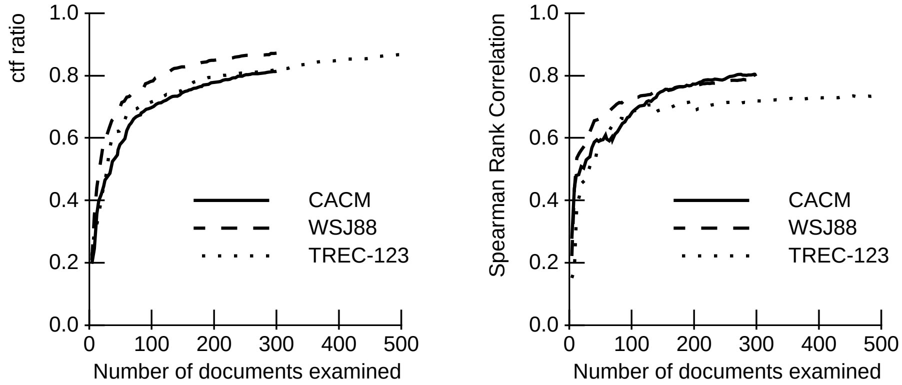
$$\frac{\sum_{i \in V'} ctf_i}{\sum_{i \in V} ctf_i} \tag{1.16}$$

where ctf_i is the number of times term i occurs in the database (database term frequency, or ctf). A *ctf* ratio of 80% means that the learned resource description contains the terms that account for 80% of the term occurrences in the database.

The Spearman Rank Correlation Coefficient is an accepted metric for comparing two orderings, in this case an ordering of terms by frequency. The Spearman Rank Correlation Coefficient is defined (Press et al., 1992) as:

$$R = \frac{1 - \frac{6}{n^3-n}(\Sigma d_i^2 + \frac{1}{12}\Sigma(f_k^3 - f_k) + \frac{1}{12}\Sigma(g_m^3 - g_m))}{\sqrt{(1 - \frac{\Sigma(f_k^3 - f_k)}{n^3-n})}\sqrt{(1 - \frac{\Sigma(g_m^3 - g_m)}{n^3-n})}} \tag{1.17}$$

where d_i is the rank difference of common term i, n is the number of terms, f_k is the number of ties in the kth group of ties in the learned resource description, and g_m is the number of ties in the mth group



(a) **(b)**

Figure 1.3 Measures of how well a learned resource description matches the actual resource description of a full-text database. **(a)** Percentage of database word occurrences covered by terms in the learned resource description. **(b)** Spearman rank correlation coefficient between the term rankings in the learned resource description and the database. (Four documents examined per query.)

of ties in the actual resource description.[1] Two orderings are identical when the rank correlation coefficient is 1. They are uncorrelated when the coefficient is 0, and they are in reverse order when the coefficient is -1.

Prior to comparison with *ctf ratio* and Spearman Rank Correlation metrics, identical stopword lists and stemming algorithms were applied to the learned and actual language models. *ctf ratios* would have been significantly higher if stopwords were retained in the language models.

Query-based sampling supports different sampling strategies, depending upon how query terms are chosen, how many documents are examined from each query, and how often the learned language model is updated with new information. The baseline experiment presented here was based on selecting query terms randomly from the learned language model, examining four documents per query, and updating language models immediately with new information. The initial query term was selected randomly from another convenient resource, in this case, the TREC-123 database.

The choice of the initial query term was a source of bias in these experiments. However, preliminary experiments showed that as long as

[1]Simpler versions of the Spearman Rank Correlation Coefficient are more common (e.g., (Moroney, 1951)). However, simpler versions assume that two elements cannot share the same ranking. Term rankings have *many* terms with identical frequencies, and hence identical rankings.

the initial query term returned at least one document, the choice of the initial query term had little effect on the quality of the language model learned.

Experimental results are summarized in Figure 1.3. Figure 1.3a shows that the sampling method quickly identifies the vocabulary that represents 80% of the non-stopword term occurrences in each database. Figure 1.3b shows that the sampling method also quickly learns the relative frequencies of terms in each database. The rate at which resource descriptions converged was independent of database size and heterogeneity.

The results shown here are based on examining the top 4 documents retrieved for each query, but similar results are obtained when 1, 2, 4, 6, 8, and 10 documents are examined per query (Callan et al., 1999a). Smaller samples, for example 1 or 2 documents per query, produced slightly more accurate language models for heterogeneous databases. Larger samples, for example, 4 or 6 documents per query, produced slightly faster learning for homogeneous databases. The differences were consistent, but not significant. When nothing is known about the contents of a database, the best strategy is to take small samples, trading off speed for guaranteed accuracy.

Several different approaches to query term selection were tested, including selecting terms from the learned language model using frequency criteria, and selecting terms that appear important in other, presumably similar language models (Callan et al., 1999a; Callan and Connell, 1999). Frequency-based selection was rarely a good choice. Selecting query terms from another language model was only a good choice when that other language model was very similar to the database being sampled; in other words, if one has a good guess about what a database contains, the databased can be sampled more efficiently; otherwise, random sampling is best.

The language models for all three databases required about the same number of documents to converge. Database size and heterogeneity had little effect on the rate of convergence. This characteristic is consistent with Zipf's "law" (Zipf, 1949), which states that the rate at which new terms are found decreases with the number of documents examined. Zipf's law places no constraints on the order in which documents in a database are examined. Whether documents are selected sequentially or by query-based sampling, only a relatively small number of documents is required to identify most of the vocabulary in a database of documents.

6.2 ACCURACY OF RESOURCE RANKINGS

One might expect relatively accurate language models to produce relatively accurate resource rankings. However, no prior research indicated how much inaccuracy in a language model could be tolerated before resource ranking accuracy deteriorated. A set of experiments was designed to study this issue.

Resource ranking accuracy was studied using the testbed of 100 databases created from TREC CDs 1, 2, and 3 (Section 2). 100 complete resource descriptions were created (one per database). 100 learned resource descriptions were also created (one per database). The learned resource descriptions were created using query-based sampling, with query terms selected randomly from the learned language model, and 4 documents examined per query. Each databases was sampled with enough queries to yield a specified number of unique documents. Sample sizes of 100, 300, and 700 documents were examined.

Databases were ranked with the CORI database ranking algorithm (Section 4). The CORI algorithm normalizes term frequency statistics ($df_{i,j}$) using the length, in words, of the database (cw_j) (Callan et al., 1995b). It is not known yet how to estimate database size with query-based sampling. In these experiments, term frequency information (df) was normalized using the length, in words, of the set of documents used to construct the resource description.

Queries were based on TREC topics 51-150 (Harman, 1994). The query sets were INQ001 and INQ026, both created at the CIIR (Callan et al., 1995a). Queries in these query sets are long, complex, and have undergone automatic query expansion. The relevance assessments were the standard TREC relevance assessments supplied by the U.S. National Institute for Standards and Technology (Harman, 1994).

The experimental results are summarized in Figure 1.4. The baselines are the curves showing results with the actual resource description ("complete resource descriptions"). This is the best result that the database ranking algorithm can produce when given a complete description for each database.

Resource rankings produced from learned language models were slightly less accurate than rankings produced from complete language models. However, the difference was small when learned language models were created from 700 and 300 documents. The difference was greater when language models were learned from only 100 documents, but the loss is small compared to the information reduction. Accuracy at "low recall" (only 10-20% of the databases searched) was quite good.

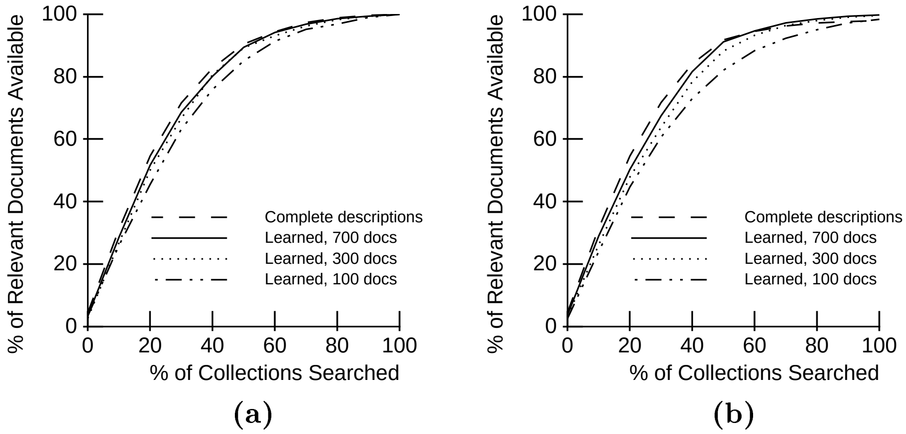


Figure 1.4 Measures of database ranking accuracy using resource descriptions of varying accuracy. **(a)** Topics 51-100 (TREC query set INQ026). **(b)** Topics 101-150 (TREC query set INQ001). (4 documents examined per query. TREC volumes 1, 2, and 3.)

These results are consistent with the results presented in Section 6.1. The earlier experiments showed that term rankings in the learned and actual resource descriptions were highly correlated after examining 300 documents. These experiments demonstrate that the degree of correlation is sufficiently high to enable accurate resource ranking.

6.3 ACCURACY OF DOCUMENT RANKINGS

Relatively accurate database rankings are a prerequisite for accurate document rankings, but the degree of accuracy required in the database ranking was not known. In particular, it was not known whether the minor database ranking errors introduced by learned language models would cause small or large errors in document ranking. A set of experiments was designed to study this issue.

Document ranking accuracy was studied using the testbed of 100 databases created from TREC CDs 1, 2, and 3 (Section 2). 100 complete resource descriptions were created (one per database). 100 learned resource descriptions were also created (one per database). The learned resource descriptions were created using query-based sampling, with query terms selected randomly from the learned language model, and 4 documents examined per query. Each databases was sampled with enough queries to yield 300 unique documents.

The CORI database selection algorithm ranked databases using either the learned resource descriptions or the complete resource descrip-

Table 1.3 Precision of a search system using complete and learned resource descriptions for database selection and result merging. TREC volumes 1, 2, and 3, divided into 100 databases. 10 databases were searched for each query.

	Topics 51-100 (INQ026 queries)		*Topics 101-150 (INQ001 queries)*	
Document Rank	*Complete Resource Descriptions*	*Learned Resource Descriptions*	*Complete Resource Descriptions*	*Learned Resource Descriptions*
5	0.5800	0.6280 (+8.3%)	0.5960	0.5600 (−6.0%)
10	0.5640	0.6040 (+7.1%)	0.5540	0.5520 (−0.3%)
15	0.5493	0.5853 (+6.6%)	0.5453	0.5307 (−2.7%)
20	0.5470	0.5830 (+6.6%)	0.5360	0.5270 (−1.7%)
30	0.5227	0.5593 (+7.0%)	0.5013	0.4993 (−0.4%)

tions, as determined by the experimenter. The 10 databases ranked most highly for each query by the database selection algorithm were searched by INQUERY. The number 10 was chosen because it was used in recent research on distributed search (Xu and Callan, 1998; Xu and Croft, 1999). Each searched database returned its most highly ranked 30 documents. Document rankings produced by different databases were merged into a single ranking by INQUERY's default result-merging algorithm (Section 5). Document ranking accuracy was measured by precision at ranks 5, 10, 15, 20, and 30.

The experimental results indicate that distributed retrieval is about as effective with learned resource descriptions as it is with complete resource descriptions (Table 1.3). Precision with one query set (INQ026, topics 51-100) was 6.6−8.3% higher using learned descriptions. Precision with the other query set (INQ001, topics 101-150) averaged 2.2% lower using learned descriptions, with a range of −0.3% to −6.0%. Both the improvement and the loss were too small for most people to notice.

Experiments were also conducted with shorter queries. Sets of queries were created for TREC topics 51-100 using text from the Title fields (*Title queries*), and sets were created using text from the Description fields (*Description queries*). Summary characteristics for the query sets are shown in Table 1.4.

Table 1.5 summarizes the results of experiments with shorter queries. The shorter queries produce rankings with lower precision than the long queries (INQ026 and INQ001, Table 1.3), which was expected. The difference in precision between searches done with complete language models and with learned language models is larger than in experiments

Table 1.4 Summary statistics for the query sets used with the testbed.

Query Set Name	*TREC Topic Set*	*TREC Topic Field*	*Average Length (Words)*
Title queries, 51-100	51-100	Title	3
Title queries, 101-150	101-150	Title	4
Description queries, 51-100	51-100	Description	14
Description queries, 101-150	101-150	Description	16

Table 1.5 The effects of query-based sampling on the CORI database ranking algorithm, as measured by the precision of the document rankings that are produced. 10 databases searched in a 100 database testbed. **(a)** Title queries. **(b)** Description queries.

Title queries				
Precision	*Topics 51-100*		*Topics 101-150*	
at Rank	*Full*	*Sampled*	*Full*	*Sampled*
5 docs	0.4800	0.4520 (−5.8%)	0.4440	0.4440 (0.0%)
10 docs	0.4400	0.4280 (−2.7%)	0.4100	0.3920 (−4.4%)
15 docs	0.4240	0.4067 (−4.1%)	0.3987	0.3627 (−9.0%)
20 docs	0.4070	0.3870 (−4.9%)	0.3740	0.3470 (−7.2%)
30 docs	0.3913	0.3620 (−7.5%)	0.3560	0.3267 (−8.2%)
100 docs	0.3054	0.2748 (−10.0%)	0.2720	0.2576 (−5.3%)

(a)

Description queries				
Precision	*Topics 51-100*		*Topics 101-150*	
at Rank	*Full*	*Sampled*	*Full*	*Sampled*
5 docs	0.4960	0.4840 (−2.4%)	0.4560	0.4920 (+7.9%)
10 docs	0.4660	0.4540 (−2.6%)	0.4260	0.3980 (−6.6%)
15 docs	0.4520	0.4227 (−6.5%)	0.3973	0.3600 (−9.4%)
20 docs	0.4350	0.4080 (−6.2%)	0.3890	0.3430 (−11.8%)
30 docs	0.4273	0.3860 (−9.7%)	0.3733	0.3327 (−10.9%)
100 docs	0.3128	0.2772 (−11.4%)	0.2702	0.2376 (−12.1%)

(b)

with longer queries (Table 1.5). The drop in precision was $5 - 10\%$ with all but one one query set; in one test, precision actually improved slightly.

These experimental results with short and long queries extend the results of the previous sections, which indicated that using learned resource descriptions to rank databases introduced only a small amount of error into the ranking process. These results demonstrate that the small errors introduced by learned resource descriptions do not noticeably reduce the accuracy of the final search results.

The accuracy of the document ranking depends also on merging results from different databases accurately. The experimental results indicate that learned resource descriptions support this activity as well. This result is important because INQUERY's result merging algorithm estimates a normalized document score as a function of the database's score and the document's score with respect to its database. The results indicate that not only are databases *ranked* appropriately using learned descriptions, but that the *scores* used to rank them are highly correlated with the scores produced with complete resource descriptions. This is further evidence that query-based sampling produces very accurate resource descriptions.

7. SUMMARY & CONCLUSIONS

The research reported in this paper addresses many of the problems that arise when full-text information retrieval is applied in environments containing many text databases controlled by many independent parties. The solutions include techniques for acquiring descriptions of resources controlled by uncooperative parties, using resource descriptions to rank text databases by their likelihood of satisfying a query, and merging the document rankings returned by different text databases. Collectively, these techniques represent an end-to-end solution to the problems that arise in distributed information retrieval.

The distributed IR solutions developed in this paper are effective under a broad set of conditions. The experimental conditions include testbeds with relatively uniform database sizes, testbeds with relatively heterogeneous database sizes, and testbeds ranging in size from $O(10)$ to $O(1,000)$ databases. The solutions scale to at least $O(1,000)$ databases. The experiments presented in this paper are a representative subset of distributed IR experiments done at the CIIR over a five year period. The core algorithms required little adjustment during that time.

The experimental methodology developed as part of this research was intended to reflect conditions in wide area computer networks. These conditions include minimal cooperation among parties, a complete lack of global corpus information (e.g., *idf* statistics), a desire to minimize communication costs, and a desire to minimize the number of interactions among parties. Database ranking algorithms were evaluated by how well they identified databases containing *the largest number* of relevant documents for each query, and by the precision an end-user would see. The intent was to be as "real world" and unforgiving as possible.

In spite of good intentions, weaknesses remain, and these reflect opportunities for future research. The major remaining weakness is the algorithm for merging document rankings produced by different databases. This paper presents two versions of the algorithm. One requires some cooperation among parties; the other does not. Neither algorithm has a strong theoretical basis, and neither algorithm has been tested with document rankings and document scores produced by multiple, disparate search systems, as would be common in the "real world". These weaknesses could be avoided, at some computational cost, by parsing and reranking the documents at the search client. They could also be avoided with a simpler heuristic algorithm, at the cost of a decrease in precision, as in Allan et al., 1996. However, an accurate and efficient solution to this problem remains unknown.

The experimental results with $O(1,000)$ databases demonstrate the need for additional research on "high precision" database ranking algorithms. Few people can or will search 10% of the databases when many databases are available. The most useful algorithms will be those that are effective when 10 out of 1,000 databases (1%), or 10 out of 10,000 databases (0.1%) are searched. None of the prior research has studied this level of accuracy.

The research reported in this paper represents a large first step towards creating a complete multi-database model of full-text information retrieval. A simple distributed IR system can be built today, based on the algorithms presented here. However, many of the traditional IR tools, such as relevance feedback, have yet to be applied to multi-database environments. Query expansion greatly improves the ranking of databases (Xu and Callan, 1998), but this result is of only academic interest until there is a general method for creating query expansion databases that accurately represent many other databases. Nobody has shown how to summarize database contents so that a person can browse in an environment containing thousands of databases. These and related problems are likely to represent the next wave of research in distributed information retrieval.

Acknowledgments

I thank Margie Connell, Zhihong Lu, Aiqun Du, Hongmin Shu, and Yun Mu for their assistance with the research reported here.

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement EEC-9209623; by the U.S. Patent and Trademark Office and Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract F19628-95-C-0235; and by the National Science Foundation, under grant IIS-9873009. Any opinions, findings, conclusions or recommendations expressed in this material are the author's, and do not necessarily reflect those of the sponsor(s).

References

Allan, J., Ballesteros, L., Callan, J. P., Croft, W. B., and Lu, Z. (1996). Recent experiments with inquery. In Harman, D., editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*. National Institute of Standards and Technology Special Publication.

Callan, J. (1999a). Distributed IR testbed definition: trec123-100-bysource-callan99.v2a. Technical report, Language Technologies Institute, Carnegie Mellon University. Available at http://www.cs.cmu.edu/~callan/Data/.

Callan, J. (1999b). Distributed IR testbed definition: trec123-17-bysource-callan99.v1a. Technical report, Language Technologies Institute, Carnegie Mellon University. Available at http://www.cs.cmu.edu/~callan/Data/.

Callan, J. (1999c). Distributed IR testbed definition: trecvlc1-921-bysource-callan99. Technical report, Language Technologies Institute, Carnegie Mellon University. Available at http://www.cs.cmu.edu/~callan/Data/.

Callan, J. and Connell, M. (1999). Query-based sampling of text databases. Technical Report IR-180, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts.

Callan, J., Connell, M., and Du, A. (1999a). Automatic discovery of language models for text databases. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, pages 479–490, Philadelphia. ACM.

Callan, J., Powell, A. L., French, J. C., and Connell, M. (1999b). The effects of query-based sampling on automatic database selection algorithms. Technical Report IR-181, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts.

Callan, J. P., Croft, W. B., and Broglio, J. (1995a). TREC and TIPSTER experiments with INQUERY. *Information Processing and Management*, 31(3):327–343.

Callan, J. P., Lu, Z., and Croft, W. B. (1995b). Searching distributed collections with inference networks. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, Seattle. ACM.

Chakravarthy, A. and Haase, K. (1995). NetSerf: Using semantic knowledge to find Internet information archives. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, Seattle. ACM.

Du, A. and Callan, J. (1998). Probing a collection to discover its language model. Technical Report 98-29, Department of Computer Science, University of Massachusetts.

Dumais, S. T. (1994). Latent semantic indexing (LSI) and TREC-2. In Harman, D. K., editor, *The Second Text REtrieval Conference (TREC-2)*, pages 105–115, Gaithersburg, MD. National Institute of Standards and Technology, Special Publication 500-215.

French, J., Powell, A., Callan, J., Viles, C., Emmitt, T., Prey, K., and Y.Mou (1999). Comparing the performance of database selection algorithms. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245. ACM.

French, J., Powell, A., Viles, C., Emmitt, T., and Prey, K. (1998). Evaluating database selection techniques: A testbed and experiment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Fuhr, N. (1999). A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3):229–249.

Gravano, L., Change, K., García-Molina, H., and Paepcke, A. (1996). STARTS Stanford protocol proposal for Internet retrieval and search. Technical Report SIDL-WP-1996-0043, Computer Science Department, Stanford University.

Gravano, L. and García-Molina, H. (1995). Generalizing GLOSS to vector-space databases and broker hierarchies. In *Proceedings of the 21st International Conference on Very Large Databases (VLDB)*, pages 78–89.

Gravano, L., García-Molina, H., and Tomasic, A. (1994). The effectiveness of GLOSS for the text database discovery problem. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, pages 126–137. ACM. SIGMOD Record 23(2).

Harman, D., editor (1994). *The Second Text REtrieval Conference (TREC2)*. National Institute of Standards and Technology Special Publication 500-215, Gaithersburg, MD.

Harman, D., editor (1995). *Proceedings of the Third Text REtrieval Conference (TREC-3)*. National Institute of Standards and Technology Special Publication 500-225, Gaithersburg, MD.

Harman, D., editor (1997). *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*. National Institute of Standards and Technology Special Publication 500-238, Gaithersburg, MD.

Hawking, D. and Thistlewaite, P. (1999). Methods for information server selection. *ACM Transactions on Information Systems*, 17(1):40–76.

Kirsch, S. T. (1997). Document retrieval over networks wherein ranking and relevance scores are computed at the client for multiple database documents. U.S. Patent 5,659,732.

Kwok, K. L., Grunfeld, L., and Lewis, D. D. (1995). TREC-3 ad-hoc, routing retrieval and thresholding experiments using PIRCS. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD. National Institute of Standards and Technology, Special Publication 500-225.

Lu, Z., Callan, J., and Croft, W. (1996a). Applying inference networks to multiple collection searching. Technical Report 96-42, Department of Computer Science, University of Massachusetts.

Lu, Z., Callan, J., and Croft, W. (1996b). Measures in collection ranking evaluation. Technical Report 96-39, Department of Computer Science, University of Massachusetts.

Marcus, R. S. (1983). An experimental comparison of the effectiveness of computers and humans as search intermediaries. *Journal of the American Society for Information Science*, 34:381–404.

Moroney, M. (1951). *Facts from figures*. Penguin, Baltimore.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical recipies in C: The art of scientific computing*. Cambridge University Press.

Robertson, S. and Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, Dublin, Ireland. ACM.

Turtle, H. (1990). Inference networks for document retrieval. Technical Report COINS Report 90-7, Computer and Information Science Department, University of Massachusetts, Amherst, MA 01003.

Turtle, H. R. and Croft, W. B. (1991). Efficient probabilistic inference for text retrieval. In *RIAO 3 Conference Proceedings*, pages 644–661, Barcelona, Spain.

Viles, C. L. and French, J. C. (1995). Dissemination of collection wide information in a distributed Information Retrieval system. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 12–20, Seattle. ACM.

Voorhees, E., Gupta, N., and Johnson-Laird, B. (1995a). Learning collection fusion strategies. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–179, Seattle. ACM.

Voorhees, E. M., Gupta, N. K., and Johnson-Laird, B. (1995b). The collection fusion problem. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD. National Institute of Standards and Technology, Special Publication 500-225.

Xu, J. and Callan, J. (1998). Effective retrieval of distributed collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–120, Melbourne. ACM.

Xu, J. and Croft, W. (1999). Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 254–261, Berkeley. ACM.

Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology.* Addison-Wesley, Reading, MA.