Information Retrieval with Distributed Databases: Analytic Models of Performance

Robert M. Losee and Lewis Church Jr.

Abstract—The major emphasis of this paper is on analytical techniques for predicting the performance of various collection fusion scenarios. Knowledge of analytical models of information retrieval system performance, both with single processors and with multiple processors, increases our understanding of the parameters (e.g., number of documents, ranking algorithms, stemming algorithms, stop word lists, etc.) affecting system behavior. While there is a growing literature on the implementation of distributed information retrieval systems and digital libraries, little research has focused on analytic models of performance. We analytically describe the performance for single and multiple processors, both when different processors have the same parameter values and when they have different values. The use of different ranking algorithms and parameter values at different sites is examined.

Index Terms—Collection fusion, information retrieval, metasearch engines, distributed processing, analytic performance models, digital libraries.

1 Introduction

Information retrieval systems have been used for decades, with more recent search engines using parallel processing to achieve the desired level of throughput. Analytic models of retrieval performance have become available for evaluating system performance without conducting retrieval experiments [1], [2]. Experimental methods still predominate in the study of retrieval and are required of participants in the TREC program [3] administered by the US National Institute of Standards and Technology (http://trec.nist.gov). However, analytic models of retrieval performance may be more appropriate tools to use when one is trying to understand the relationships between variables, such as performance compared to the number of processors, density of terms in a given relevance class, number of documents with query features, etc.

Many retrieval systems today reference documents from disparate sources or collections. These sources may reside on the same computer as the retrieval system, some of them may be dispersed over a local or wide area network, whereas others may reside at various Internet locations. A common task for a retrieval system, in response to a query, is to retrieve documents that satisfy the query from these sources (which may have different document ranking schemes, different stemming algorithms, different stopword lists, etc.), rerank these documents according to its own ranking scheme, and return the result to the entity that posed the query. In essence, the retrieval system acts as a metasearch engine [4].

There are a multitude of strategies and techniques that a metasearch engine can use in its goal of obtaining the qualifying documents. Associated with these various ways, though, can be vastly different levels of performance which can be demonstrated by various normalized measures such as precision, recall, mean reciprocal rank [5], etc. The key to good performance, from both effectiveness and/or efficiency viewpoints, is a mechanism that enables the engine to predict the performance of various strategies and use that knowledge to guide it in efficiently obtaining high quality results. This paper is primarily concerned with the effectiveness aspect of performance.

2 Retrieval and Distributed Databases

The effectiveness of information retrieval systems is often described in terms of relevant and nonrelevant documents being retrieved or not retrieved. Most retrieval performance measures are based to some extent on the ordering of documents of varying degrees of expected relevance. Relevant documents may be loosely described as those documents that address the user's needs. The relationship between a query and a document may be described as relevant if it is objectively determined to be relevant or it may be viewed as a subjective characteristic, based on the individual's characteristics [6], [7], [8], [9]. Relevance is viewed most commonly as binary; that is, a document is either relevant or nonrelevant. It may also be viewed as many valued [10], [11], or it may be viewed as a continuous phenomena [2]. Below, we always treat relevance as a binary phenomenon to simplify our analysis of the retrieval process.

Similarly, we simplify our analysis by using univariate methods, rather than using multivariate methods. Focusing on single terms or phrases, treated as single units [12], and treating documents and queries as though they all have or do not have this single term or phrase, will simplify our analysis. While many queries have more than one feature, and many systems would examine more than one feature when making retrieval decisions, limiting our scope to a univariate model where we focus on single terms allows one to examine complex relationships; treating this as a multivariate problem can lead to a level of complexity that

• *The authors are with the University of North Carolina-Chapel Hill, Manning Hall, CB#3360, Chapel Hill, NC 27599-3360. E-mail: {losee, churchl}@ils.unc.edu.*

Manuscript received 17 Sept. 2002; revised 2 Apr. 2003; accepted 12 June 2003.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 117403.

makes it difficult to understand retrieval in a distributed environment.

The focus of this paper is on *collection fusion* [13], [14], [15]. More specifically, the emphasis is on analytic techniques that predict the performance of systems that enable the retrieval of documents from distributed databases.

The proliferation of sites on the World Wide Web (WWW) during the past decade has been wonderful for enhancing the availability and accessibility of information. Information is stored, often redundantly across sites, at many locations on the WWW. A significant portion of this information resides in text documents. These documents can be stand-alone or they can reside in a digital library [16]. Each site on the WWW can be considered, at least conceptually, as a data-containing processor. Documents are accessible via these processors.

Information requests are typically made via a search engine. For broader coverage, it is often advised that a metasearch engine be used for searching, as the WWW is too large for a single search engine [17]. Many complications would arise if a human tried to take on the tasks of determining what sites to search, which databases to investigate at each site, gathering any retrieved document(s), and ranking them [4].

A metasearch engine takes an information request and employs many other search engines to assist in the searching. After this searching is complete, its task is to merge the results obtained into a unified collection of answers. Much more often than not, the documents are presented to the user ranked in descending order of similarity to the user's information request. For each document, the ranking assigned to it is based on an estimation of the probability that it is relevant to the information request. The rankings, as a whole, are consistent with those that would be consistent with the *optimal document retrieval principle* [18], [19].

2.1 Challenges

The key challenge for a metasearch engine is to take an information need, expressed as a query, submit it to the other search engines, and attempt to obtain the same results that would be generated if the documents accessible via those other search engines were *all* in a *single* collection. This is the essence of the distributed information retrieval problem.

In order to be useful, this process must perform well (e.g., be both effective and efficient). This process should 1) provide answers similar to those that could be attained with all the information being in a single collection and 2) the use of system resources (e.g., communication time, bandwidth, the number of sites polled, etc.) should be minimized. Unfortunately, there is tension between 1) and 2). This tension is typically resolved by making various trade offs which impact the strategy used by the metasearch engine [20], [21], [22].

The WWW, as it exists today, can be viewed as a distributed heterogeneous federated entity [23], with each site enjoying a substantial degree of autonomy [24]. It is those characteristics that make the WWW such a potent information resource; however, it is precisely because of those characteristics that the area of distributed information retrieval (DIR) poses many challenges.

The challenges are due primarily to: distribution of control, site autonomy, and heterogeneity. In practice, these challenges cause many headaches, as well as pose many interesting and challenging problems to both practitioners and researchers.

On autonomy, Meng et al. [4] states:

> Component search engines that participate in a metasearch engine are often built and maintained independently. Each search engine decides the set of documents it wants to index and provide search service to. It also decides how documents should be represented/indexed and when the index should be updated. Similarities between documents and user queries are computed using a similarity function. It is completely up to each search engine to decide what similarity function to use. Commercial search engines often regard the similarity functions they use and other implementational decisions as proprietary information and do not make them available to the general public.

Many heterogeneities arise due to this autonomy [4]. Because of this, there may be differences between search engines in: the indexing method, the document term weighting scheme, the query term weighting scheme, the similarity function, the document database, the document version, and how results are presented.

These differences represent just the tip of the iceberg. In addition to the areas listed above, Baumgarten [25] details many other problem areas. Some of those include:

1. the coordination problem (i.e., there may be difficulties in coordinating ranking and indexing methods),
2. the intellectual property problem (i.e., document providers might be unwilling to release plain documents for indexing purposes),
3. the capacity problem (i.e., the processing, communication, and storage requirements for the sites may vary enormously, thus leading to network capacity issues), and
4. the up-to-dateness problem (i.e., how up-to-date is the metadata available to the metasearch engine?).

Meng et al. [4] claims that, in addition to the heterogeneities just enumerated above that may be present in the component search engines, there quite likely are heterogeneities present between the metasearch engine and the component search engines. How can this happen? Very easily, as the global similarity function (GSF) used by the metasearch engine is not likely to be the one used locally by any of the component engines. Furthermore, these component engines are not even likely to be using the same similarity function. Even if they use the same local similarity function (LSF), it is possible that a particular document, replicated at each of these sites' collections, could receive vastly different rankings due to characteristics of the collections where the documents appear [21].

Gravano and Garcia-Molina [26] provide insight into what can happen when the GSF differs, even slightly, from the LSFs—which may differ even among themselves. Due to this GSF/LSF difference, it is very possible that a document ranked high by an LSF may be ranked much lower, or even low, by the GSF. If the local search engine is one that limits the number of documents returned, say to 50, it is entirely possible that the merged results may be missing some relevant documents from the perspective of the metasearch engine.

This variance in similarity functions and other information that may be returned by the local search engines complicate the metasearch engine's task of merging the results. The STARTS protocol proposal [20], developed at Stanford, is an effort to provide guidelines as to the information that a search engine should provide to enable metasearchers to obtain better results. A followup to that protocol, SDARTS [27], combines "two complementary protocols (e.g., STARTS and SDLIP [28]) for searching over distributed collections" to form a protocol and toolkit for metasearching.

The distribution, autonomy, and heterogeneity discussed above have stimulated distributed information retrieval research in an area known as the *collection fusion problem*. This problem, as defined by Voorhees et al. [13], can be viewed as consisting of the following four subproblems: database selection, query translation, document selection, and results merging.

Database selection [29], [30], [31], [21] is concerned with determining which database(s) to use. Some factors that may hinder database selection are that some sites may be uncooperative, some may charge a fee, and sites may differ in how much information they make available about their collections.

Query translation [32], [21] is concerned with translating the metasearch query into a form that the site databases understand. This arises since the interfaces and capabilities of these sources may vary dramatically [21]. Some typical problems are:

1. the retrieval models may differ;
2. even if two databases support the same model, their query syntax may differ;
3. different stemming algorithms;
4. different stop-word lists; and
5. different vocabularies.

Document selection [33], [34] is the determination, for each selected database, of the types and number of documents to ask for. Problems here include the same document appearing in the collections of distinct sites and being ranked vastly differently and the ranking being influenced by site-specific factors.

Results merging [4] focuses on the question of how to combine the results, if any, from the component search engines into a coherent whole. Complicating factors are that different sites will probably be using different similarity functions, some may return results that are not ranked, and even when ranked results are returned, some sites may not return the numeric values associated with these rankings.

2.2 Past Approaches

Various methodologies and techniques [35], [36], [37], [38], [39] have been proposed and implemented to solve, ameliorate, or gain insight into database selection, document selection, and results merging. Many of these have been "(at least in parts) heuristic in nature" [40]. Of the three areas cited above, the ones that a metasearch engine has the most influence over are database selection and results merging [25].

Individually and collectively, all three of the areas affect *performance*. Each of them can be examined from either a qualitative or a system resource perspective. In practice, these perspectives are often inversely related (e.g., the usage of more resources may enable better collection selection, which may result in better document selection, which may result in a better collection of ranked documents to present to the user; on the other hand, not having the resources available to do better collection selection may cause a poor collection of ranked documents to be presented to the user).

2.3 Predicting Performance

What this paper will focus on is the quality aspect of DIR (e.g., effectiveness). It provides formulas for *predicting* the performance of various aspects of DIR via analytic means akin to cost formulas [41], [42] developed in the relational database management community (and used for analysis, prediction, and as an aid to query optimization) over approximately the last 25 years. More specifically, this paper makes the following contributions in predicting system retrieval performance:

1. methods for estimating the precision of the system and
2. methods for predicting retrieval performance measured as Average Search Length.

A key contribution of our techniques for prediction is the use of Average Search Length (ASL), the expected position of a relevant document in an ordered list of documents [2], which may be predicted analytically.

This predictive capability primarily provides two benefits: 1) it enables a better understanding of the factors that contribute to effective distributed retrieval and 2) it provides objective criteria that should enable metasearch engines to make better decisions.

Distributed information retrieval and distributed data retrieval are different, yet have much in common. Some techniques and formulas developed in the database research community should be adaptable for use in the information retrieval (IR) community. Many similar concepts exist in both but may be known by different names. For example, the term "generality" [43], a misnomer according to the reference just cited, in the information retrieval community has an analog (e.g., "selectivity" [41]) in the data retrieval community. Both have values in the closed interval [0, 1] and can be thought of as the percentage of documents/records that meet a certain criteria. In information retrieval, the criteria is relevance; in data retrieval, it is satisfying the conditions of a predicate/filter. However, in these communities, the perspectives are opposite, i.e., a value of 0.95 would be considered "highly specific" (Korfhage [43]) in the information retrieval community since it indicates a very large percentage of relevant documents whereas that same value, in the database community, would be considered "low selectivity" because, in the context of filtering, it excludes very few records. Hence, it has little discrimination power.

It is hoped that, over time, more analytic tools will be developed for use in the IR community and that these can be used, similar to formulas developed in the data retrieval community, to analytically model performance and enhance the processing of user information requests, both qualitatively and also in terms of the usage of system resources.

3 Retrieval Performance

Retrieval performance is often measured using *precision* and *recall*. Precision is the probability that a document is relevant given that it is retrieved. Less formally, it serves as an indicator of the quality of the retrieved set of documents. Recall is the probability that a document has been retrieved given that it is relevant. A *high recall* search is one in which most of the relevant documents have been retrieved. This is desirable in environments such as law or academia where the cost of missing a relevant document may be very high. On the other hand, *high precision* searches are those where the searcher desires a few good documents and does not want to wade through a lot of documents. Most quick searches on the Internet are high precision searches.

Notationally, N represents the total number of documents, R represents the total number of relevant documents, $n_{i,j}$ represents the number of documents in processor i with feature frequency j, $r_{i,j}$ represents the number of relevant documents in processor i with feature frequency j, and $g_i = \frac{r_{i,1}+r_{i,0}}{n_{i,1}+n_{i,0}}$ represents the generality for processor i.

When we have a small set consisting of the first b documents from the ordered list, we may compute the precision for this set and denote it as P_b, which may be computed as:

$$P_b = \big(\Pr(rel|d)\min(n_{1,1}, b) + \Pr\big(rel|\bar{d}\big)\big(b - \min(n_{1,1}, b)\big)\big)/b. \tag{1}$$

The variables d and $\bar{d}$ represent the feature occurring with frequency 1 and 0, respectively. The $\min(x, y)$ function represents the minimum of the two values, x or y. The probabilities may be estimated using Bayesian methods and with relevance feedback [2].

If we consider the first b documents in the merged set of documents, when we find that b is smaller than $n_{1,1}$, which occurs in many situations for terms that are not extremely rare or for very small b, we may estimate P_b as the probability that a document with the feature is relevant. Thus,

$$P_b \approx \Pr(rel|d). \tag{2}$$

In many circumstances, using this estimate may be simpler and more satisfying than the exact solution.

Note that we are not examining the measure often used as a complement of precision, *recall*, the percent of relevant documents that have been retrieved. Most search engines address high precision searches rather than high recall searches.

More problematic is that we are not looking at the retrieval of different aspects of a topic. If one were obtaining recipes for apple pie, for example, it might be helpful if the system attempted to provide recipe diversity for the first documents displayed, perhaps by displaying some recipes with cinnamon and some without cinnamon, some with sweet apples and some with tart apples, etc.

We begin modeling retrieval performance by suggesting a measure for performance: the Average Search Length (ASL), the expected position of a relevant document in the ordered list of documents. For example, given documents strongly ordered and having binary relevance values of *Relevant, Relevant, Nonrelevant, Relevant*, the average position of a relevant document is the average of positions 1, 2, and 4, or 2.333. In the case where the documents are weakly ordered and a set of documents has the same weight (or the same profiles) vis-à-vis the query, we will treat the location of each of the documents in the subset of the ordered set as the average of the locations of this subset.

While the ASL measure is effective at measuring the placement of relevant documents in an ordered list of documents, other measures may be more effective at showing the quality of the documents at the extremes of the distribution of ordered documents.

4 Analytic Retrieval

The material in this section addresses the issue of a single processor. It can be viewed as a recap of some of the work in [12] and appears here for the convenience of the reader.

The ASL may be predicted analytically as well as measured after a set of relevant documents have been obtained, as in the example in the preceding section. We continue to focus on single feature systems, where we only consider the single term in the query. This will simplify our analysis and allow us to focus on the variables of interest that effect performance.

We begin with the measure of the quality of a ranking method, $\mathcal{Q}$. This is computed as the probability the ranking is optimal. Because ranking methods exist that appear to be very good, and some approach optimality [12], we will assume for most of this work that $\mathcal{Q} = 1$.

This is combined with the scaled expected position of a relevant document that is associated with the best-case (optimal) ranking (denoted as $\mathcal{A}$) and the worst case ranking, $\overline{\mathcal{A}}$, to compute the Average Search Length (ASL), the expected position of a relevant document in the ranked list of documents. Then, given the characteristics of a query, a database, and of the ranking algorithm itself, the expected performance of the system may be computed.

Documents may be presented to the user in one of two different orders. Documents with a binary query feature with frequency d may be followed by those with frequency $\bar{d} = 1 - d$, which we refer to as the *optimal ranking*. It is assumed that the term weight for d is greater than that for $\bar{d}$; this is the case when query terms are positive discriminators. If this is not the case, the values may be switched (reparameterized) so that the weight for d is greater than or equal to the weight for $\bar{d}$. Thus, we assume that the features are reparameterized so that feature frequency d multiplied by the term weight has a higher value than feature frequency $\bar{d}$ multiplied by the term weight. This is best-case or optimal ranking. The worst-case ranking retrieves documents with feature frequency $\bar{d}$ before documents with frequency d.

The average location of a relevant document in the optimal ranking may be computed based on a scale from 0 to 1, a unit scale with an $\mathcal{A}$ of 0 representing the average position of relevant documents being at the beginning of the search process, and an $\mathcal{A}$ of 1 being at the end of the search process. The parameter $\mathcal{A}$ is then the expected proportion of

documents examined in an optimal ranking if one examines all the documents up to the document in the average position of a relevant document. It is the expected position of a relevant document, scaled from 0 to 1.

The variable $\mathcal{A}$ is computed by noting that documents with feature frequency d are at the low end of the $\mathcal{A}$ spectrum (good performance), and those with feature frequency $\bar{d}$ at the high end of the spectrum (poor performance). The middle (average) position for each of the profiles, when they are arranged in order, is such that $\Pr(d)/2$ is the average position for documents with feature frequency d. The mean position is $1 - \Pr(\bar{d})/2$ for the documents with feature frequency $\bar{d}$; that is, when the feature frequency is 0. Documents of each feature frequency may be weighted by the probability of having the frequency of the feature given that they are relevant. This is the percent of relevant documents that have the particular frequency. Thus, we may estimate $\mathcal{A}$ as

$$\mathcal{A} = \Pr(d|rel)\Pr(d)/2 + \Pr(\bar{d}|rel)(1 - \Pr(\bar{d})/2). \quad (3)$$

Using a similar technique, we find that $\overline{\mathcal{A}}$, the $\mathcal{A}$ value for the worst-case ranking, is $1 - \mathcal{A}$.

Equation (3) may be simplified algebraically to:

$$\mathcal{A} = \frac{1 + \Pr(d) - \Pr(d|rel)}{2}.$$

If, for notational simplicity, we let $p = \Pr(d|rel)$ and $t = \Pr(d)$, we find that

$$\mathcal{A} = \frac{1 - p + t}{2}. \quad (4)$$

Even though the primary purpose of this paper is formulating the theory concerning ASL, and not on building a system, it might be helpful at this point to say something briefly about how to estimate $\Pr(d)$ and $\Pr(d|rel)$ when exact values are not available. $\Pr(d)$ can be estimated rather easily using standard statistical sampling techniques [44]. On the other hand, though, $\Pr(d|rel)$ is much more difficult to estimate. The major problem with its estimation is that, typically, relevance information is not available. Probabilistic models [45], [46], [47], [48] and Bayesian techniques [49] provide some guidance with relevance estimation techniques when very little or no relevance information is available.

As an example of the computation of $\mathcal{A}$, consider a sequence of five documents, the first two having the feature in question (and both of these documents being relevant) and the remaining three documents not having the feature (and one of these being relevant). We find that $\Pr(d|rel) = 2/3$ and $\Pr(d) = 2/5$. Applying (3), we compute the weighted average position of a relevant document: $(2/3)(1/5) + (1/3)(7/10) = 11/30$. Alternatively, using (4), we find that $(1 - p + t)/2 = (1 - 2/3 + 2/5)/2 = 11/30$. It is most helpful to view this process as computing the position of the average relevant document, normed to being between 1 and 0, or it is the proportion of documents in the data set retrieved when retrieving the conceptual document at the average position of a relevant document.

Determining the relative level of performance, compared to the best level of performance obtainable and the random performance, allows the comparison between somewhat different situations. For example, examining the retrieval performance given different numbers of documents on different processors, or different numbers of documents on a single processor at different times, may be difficult.

A normalized form of the $\mathcal{A}$ measure is

$$\mathcal{A}_{norm} = 1 - \frac{\mathcal{A}_{Best} - \mathcal{A}}{\mathcal{A}_{Best} - \mathcal{A}_{Random}}.$$

We may interpret $\mathcal{A}_{norm}$ as the degree to which optimal ranking is obtained for a query. Here, one obtains 1 when performance is optimal, 0 when performance is random, and -1 when performance is worst-case. This measure is not directly sensitive to the number of documents being considered; instead, it shows the relative performance of an already normalized measure ($\mathcal{A}$) compared to its best case and worst case.

How does one predict the Average Search Length? Here, we mean not working through an existing ranked list of documents, as one does after retrieving documents, but how to predict the ASL probabilistically, as one predicts performance using queueing theory [50]. Given N documents, $\mathcal{A}$ as defined in (4), and $\mathcal{Q}$, then the Average Search Length (ASL) when using a single query term is

$$ASL = N\big(\mathcal{Q}\mathcal{A} + \overline{\mathcal{Q}}\,\overline{\mathcal{A}}\big) + 1/2. \quad (5)$$

Estimating the ASL requires the computation of the the weighted average of $\mathcal{A}$ and $\overline{\mathcal{A}}$, where $\overline{\mathcal{A}}$ is defined as $1 - \mathcal{A}$, with $\mathcal{A}$ being the performance for best-case ranking and $\overline{\mathcal{A}}$ representing the comparable value for worst-case ranking. The weighting factors are the percent of orderings that are optimal ($\mathcal{Q}$) and the percent of rankings that are worst-case ($\overline{\mathcal{Q}}$), where $\overline{\mathcal{Q}} = 1 - \mathcal{Q}$.

The ASL may be estimated more simply if optimal retrieval, i.e., $Q = 1$, is assumed. The ASL for optimal ranking is $N(1 - p + t)/2 + 1/2$. Please note that optimal retrieval is just a simplifying assumption for this paper as, in the real-world, there is much variability in results from one search engine to the other [51]. Future work may explore relaxing this optimality assumption and/or using multiple terms.

The ASL, as one can discern from the formula above, is based on the total number of documents at a processor. In many situations, for various reasons (e.g., bandwidth limitations, protection of a valuable resource, etc.), a processor may impose a limit on the number of documents it returns [4]. This limit may be much less than the number of documents available at that processor. That situation, while less than ideal, can be handled by using, for the value of N, the actual number of documents returned rather than the total number available.

One can view the analytic model of retrieval more generally, moving beyond binary terms and binary relevance [2]. The expected term frequency for a term in the relevant documents, $E(d|rel) = \mu_r$, is the relative average position (scaled from 0 to 1) for a document in the distribution of relevant documents. By looking at the documents with frequencies at or above this position in all the documents, the $\mathcal{A}$ component may be computed as follows: $\mathcal{A} = \int_{\mu_r}^{\infty} \Pr(j)\,dj$, where $\Pr(j)$ is the probability of a document having feature frequency j.

We use two distributions in further developing this model. We refer to f_{all}, the distribution of feature frequencies in all documents, while f_{rel} refers to the distribution of feature frequencies in relevant documents. The mean of the distribution f_{rel} is μ_r and the mean of the distribution of all documents, f_{all}, is μ, denoted without a subscript. We denote the sum of the probabilities above frequency x for the distribution of all terms as the *survival function*, $\mathcal{S}(x, all)$. The survival function is one minus the cumulative distribution function for the appropriate distribution.

For continuous term distributions and binary relevance, $\mathcal{A} = \mathcal{S}(\mu_r, all)$. This follows from the definitions of $\mathcal{A}$ and the survival function. When one computes the survival function of the binary model for a single term, we arrive at (4) for $\mathcal{A}$.

5 Retrieving Documents from Distributed Databases

The performance of distributed retrieval systems may be studied analytically using some of the methods described above for predicting $\mathcal{A}$ and Average Search Length. The performance of each separate processor may be studied separately, or we may examine the performance of an external ranking applied to the merged set of documents from all the data-containing processors. We are not focusing on network communication in these performance measures, although communication time and cost are certainly important factors in the performance of such a system.

Conceptually, the metasearch engine constructs the merged list as follows: Visit each processor exactly once, remove all of the documents that have a feature frequency of d, and append those to the end of an initially empty list named A. Note that, since the ordering is optimal, all the documents with the aforementioned frequency will be at the head of each processor's list. Any remaining documents (e.g., those with a feature frequency of $\bar{d}$) are appended to an initially empty list named B. At the conclusion of the processor visits, append the elements of B to A. The resultant list is the merged list.

The performance of a single processor with an optimal ranking algorithm may be measured as $ASL = N\mathcal{A}_1 + 1/2$, where $\mathcal{A}_1 = (1 - \Pr(d|rel) + \Pr(d))/2$. Here, $\mathcal{A}_1$ represents $\mathcal{A}$ for a single processor.

For the convenience of the reader, here is a recap of some of the notation defined earlier, but that has not been used in the last several pages. Notationally, N represents the total number of documents, R represents the total number of relevant documents, $n_{i,j}$ represents the number of documents in processor i with feature frequency j, $r_{i,j}$ represents the number of relevant documents in processor i with feature frequency j, and $g_i = \frac{r_{i,1}+r_{i,0}}{n_{i,1}+n_{i,0}}$ represents the generality for processor i.

Below, we will develop expressions for generalizing both $\Pr(d_i|rel)$ and $\Pr(d_i)$ to multiple processors with the processor number designated as i. For the one processor case, $\Pr(d_i|rel)$ is equivalent to $\frac{n_{1,1}\Pr(d_1, rel_1)}{n_{1,1}g_1}$; likewise, $\Pr(d_i)$ has the same value as $\frac{n_{1,1}\Pr(d_1)}{n_{1,1}}$.

With two processors having identical parameter sets and no duplicated documents, the performance of the merging system may be measured as

$$\mathcal{A}_{2,\epsilon,\infty,0} = \frac{1}{2}\left[1 - \frac{n_{1,1}\Pr(d_1, rel_1) + n_{2,1}\Pr(d_2, rel_2)}{n_{1,1}g_1 + n_{2,1}g_2} + \frac{n_{1,1}\Pr(d_1) + n_{2,1}\Pr(d_2)}{n_{1,1} + n_{2,1}}\right]. \tag{6}$$

When referring to $\mathcal{A}$, the first subscript always represents the number of processors, the second subscript indicates that the parameters for all processors are equal (ϵ) or unequal ($\bar{\epsilon}$), the third subscript indicates the number of documents for each processor that are included in the merged output of the system, and the fourth subscript indicates the probability that a document in a processor is duplicated in another processor (probability of redundancy). In most cases, we drop the third and fourth subscripts when the context makes them unnecessary.

We may generalize this result from two processors to the performance of the merging of results from π processors so that

$$\mathcal{A}_{\pi,\bar{\epsilon}} = \frac{1}{2}\left[1 - \frac{\sum_{i=1}^{\pi} n_{i,1}\Pr(d_i, rel_i)}{\sum_{i=1}^{\pi} n_{i,1}g_i} + \frac{\sum_{i=1}^{\pi} n_{i,1}\Pr(d_i)}{\sum_{i=1}^{\pi} n_{i,1}}\right]. \tag{7}$$

When the parameters have the same values for each processor's database, performance becomes

$$\mathcal{A}_{\pi,\epsilon} = \frac{1}{2}\left[1 - \frac{\pi n_{1,1}\Pr(d_1, rel_1)}{\pi n_{1,1}g_1} + \frac{\pi n_{1,1}\Pr(d_1)}{\pi n_{1,1}}\right] = \frac{1}{2}\left[1 - p + t\right], \tag{8}$$

the same as (4). Clearly, when the same parameter values exist in each processor, n_i and g_i may arbitrarily be taken from the first processor so that the $\mathcal{A}_\pi$ value is the same as $\mathcal{A}_1$. The variables t and p represent the values as discussed earlier, when the population is the union of all the documents in the system.

6 Retrieving the Best b Nonredundant Documents from Each Processor

A metasearch engine may provide the information seeker the option to ask for only the k best unique documents to be presented to the end user, since many searches on the Internet are high precision searches. Generally, when retrieving documents from each separate database, it may be desirable to retrieve the best b documents from each database, rather than retrieving all the documents from each database. The major issue is determining how many documents to choose from each database. This number could vary easily from one database to another as two major factors affecting this are the perceived quality of each of the databases with regard to the query [33] and the probable variation in the ranking functions [26] among the various sites. These two issues are very interesting and important but are beyond the scope of this paper.

No matter how the documents are chosen, the retrieved documents from individual processors are merged at a central site and then presented to the end user using the ranking function in effect at the central site. We assume here that the ranking methods (e.g., the popular IDF weighting that serves as the basis for document ranking in many search engines) and information available are the *same* for each site. We will describe methods for determining performance with different ranking methods toward the end of this paper.

Developing a model consistent with (1), we find that, in the single processor case,

$$P_b = P_{1,b} = \frac{1}{b}\left[\frac{r_{1,1}}{n_{1,1}}\min(n_{1,1}, b) + \frac{r_{1,0}}{n_{1,0}}\big(b - \min(n_{1,1}, b)\big)\right]. \quad (9)$$

This equation finds the percentage of the initial b documents that are relevant. We do this by first computing the expected number of documents with a 1 that are relevant, with a maximum of b. If there are documents without a 1 in the first b documents, we then also examine the expected number of relevant documents from those with a zero that would occur within the first b documents.

Among the first b documents, assuming that $N \geq b$, we may estimate $t = \Pr(d)$ for these b documents as $\min(n_{1,1}, b)/b$ and estimate $p = \Pr(d|rel)$ for the first b documents as $\frac{\min(b, r_{1,1})(r_{1,1}/n_{1,1})}{\min(R,b)}$. We may then compute $\mathcal{A}$ for the first b documents retrieved from a single processor as

$$\mathcal{A}_{1,b} = \frac{1}{2}\left(1 - \frac{\min(b, r_{1,1})(r_{1,1}/n_{1,1})}{\min(R, b)} + \frac{\min(n_{1,1}, b)}{b}\right). \quad (10)$$

We can thus see that single processor performance, when measured as $\mathcal{A}$ or P, may be computed for the first b documents retrieved.

We may measure the performance of each of the individual processors, as well as the performance of the combined set. For the case where the parameters are the same for each different processor, the precision for the first b documents from each individual processor will be the precision for the entire retrieved set consisting of πb documents.

The precision for the merged set of documents is

$$P_{\pi,b,\bar{\epsilon}} = \frac{1}{\pi b}\sum_{i=1}^{\pi}\left[\frac{r_{i,1}}{n_{i,1}}\min(n_{i,1}, b) + \frac{r_{i,0}}{n_{i,0}}\big(b - \min(n_{i,1}, b)\big)\right] \quad (11)$$

when there are π processors. Here, the parameters may vary from one processor to the next.

Similarly, we may compute $\mathcal{A}$ for the same group of π processors.

$$\mathcal{A}_{\pi,b,\bar{\epsilon}} = \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{\pi}\min(b, r_{i,1})(r_{i,1}/n_{i,1})}{\sum_{i=1}^{\pi}\min(r_{i,1} + r_{i,0}, b)} + \frac{\sum_{i=1}^{\pi}\min(n_{i,1}, b)}{\pi b}\right). \quad (12)$$

If we wish to simplify our assumptions and treat each processor as having the same parameters regarding the density of documents for *relevant* and for *all* documents, we can compute the precision for the merged set as

$$P_{\pi,b,\epsilon} = \frac{1}{\pi b}\pi\left[\frac{r_{i,1}}{n_{i,1}}\min(n_{i,1}, b) + \frac{r_{i,0}}{n_{i,0}}\big(b - \min(n_{i,1}, b)\big)\right] = P_{1,b,\epsilon}, \quad (13)$$

where the subscript i can be used to represent the parameter for any processor. This is the same as $P_{1,b}$, (9).

Similarly, we may compute the $\mathcal{A}$ for the merged set, given equivalent parameters across processors, as

$$\mathcal{A}_{\pi,b,\epsilon} = \frac{1}{2}\left(1 - \frac{\pi\min(b, r_{i,1})(r_{i,1}/n_{i,1})}{\pi\min(r_{i,1} + r_{i,0}, b)} + \frac{\pi\min(n_{i,1}, b)}{\pi b}\right) = \mathcal{A}_{1,b,\epsilon}, \quad (14)$$

where the subscript i can be used to represent the parameter for any of the π processors. This is the same as $\mathcal{A}_{1,b}$, (10).

For very large b; that is, as b equals N, P_b becomes the generality of the database, and the ordering of the documents becomes irrelevant. This is one of the reasons for using the $\mathcal{A}$ measure rather than the precision measure, which focuses on the quality of the retrieved set, regardless of the ordering of these retrieved documents.

There often will be duplicated documents when moving from one database to another. When the same document occurs in multiple systems, it might be retrieved by each of the individual processors; however, the redundant documents will be filtered out by the merging processor. We assume that the probability a document is redundant remains the same across both relevance classes and across different document profiles.

To determine performance with redundant documents, we need to estimate the performance with the reduced number of documents in the merged set, as compared to the larger number of unduplicated documents that would occur given the same parameters but with no redundancy. Equations (13) ($P_{\pi,b,\epsilon}$) and (14) ($\mathcal{A}_{\pi,b,\epsilon}$) can be used to evaluate performance in these circumstances.

7 Local and Global Estimates Used in Ranking

Different sites may use different values for parameters, affecting different ranking. Here, we consider this in terms of the probability that a ranking method, including the parameter values available to it, produce optimal ranking. This can be generalized to allow us to model the performance of any ranking algorithm, given the assumptions of our models.

Estimating the values of parameters used in document ranking may result in poor estimates when the documents available for use in the estimation are not representative of the characteristics of the entire system. While there may be an optimal ranking, given all the knowledge from the entire system, individual processors may use suboptimal rankings for their documents if they do not have knowledge of the universe, but are instead limited to local knowledge, which we will assume to be knowledge of only the processor in question. When documents are sent to the merging processor, they may be accompanied by counts of terms and documents such as we have used here. When the merging processor receives documents for merging and reranking, the proper parameters for the universe may be

formulated by summing together individual parameter values from the submitting sites.

We may study the different performance obtained with ranking algorithms, differing by the different parameter estimates that are made, by comparing the performance at each individual site associated with ranking algorithms using local parameters with the optimal ranking available, given omniscience. We assume that optimal ranking, $\mathcal{R}_o$, is obtained by ordering documents so that the expected precision of the set of ordered documents weakly decreases as we move through the ordered set.

Assume that the set of available documents $\boldsymbol{d}$ is weakly ordered by a ranking algorithm $\mathcal{R}_x$ and there are $N = |\boldsymbol{d}|$ elements in the set of documents $\boldsymbol{d}$. The ranking $\mathcal{R}_x$ produces an ordering of the documents $\boldsymbol{d}_x = d_1 \succeq d_2 \succeq \cdots \succeq d_N$; that is, d_1 is weakly ordered before d_2 which is weakly ordered before d_3, and so forth to d_N. When studying the ranking of a set of documents and the resulting performance, it may be necessary to compute $\Pr(\boldsymbol{d}_x|\mathcal{R}_x)$ the probability that one will obtain a particular document ordering $\boldsymbol{d}_x$ when ranking procedure $\mathcal{R}_x$ is used.

One ranked set of documents is the best, or optimal (or tied for optimality), denoted as $\mathbf{v}_o$, and the probability that we will have this when the optimal ranking ($\mathcal{R}_o$) procedure is used is $\Pr(\boldsymbol{d}_o|\mathcal{R}_o) = 1$. This best-case ranking is the same as the ranking obtained with

$$\Pr(rel|d) = \frac{\Pr(d|rel)\Pr(rel)}{\Pr(d)}.$$

Because the $\Pr(rel)$ component is constant for each document, it may be dropped without affecting ranking; thus, ranking documents by $\Pr(d|rel)/\Pr(d)$ produces the same ranking. Given binary features, and using a weighting scheme similar to that of the Binary Independence Model [52], the weight for a feature is

$$w = \log\left(\frac{p/(1-p)}{t/(1-t)}\right), \tag{15}$$

where we still use $p = \Pr(d = 1|rel)$ and $t = \Pr(d = 1)$.

The probability that ranking methods $\mathcal{R}_x$ and $\mathcal{R}_o$ produce the same ranking, computed over the set of rankings, is

$$\mathcal{Q}_x = \Pr(\mathcal{R}_o, \mathcal{R}_x) = \sum_{i \in Perm(\boldsymbol{d})} \Pr(\boldsymbol{d}_i, \mathcal{R}_x, \mathcal{R}_o), \tag{16}$$

where $Perm(\boldsymbol{d})$ is the set of distinct weak orderings possible, rearrangements of the document profiles in vector $\boldsymbol{d}$.

To apply (16) to the case of optimal retrieval, we must compute the probability that a document with a feature value of 1 is ranked ahead of or equal to a feature with value 0:

$$\begin{aligned} 1w &> 0w \\ \log\left(\frac{p/(1-p)}{t/(1-t)}\right) &> 0. \end{aligned} \tag{17}$$

This is the case only when $p > t$. When this condition is met, the number in the left-hand side of (17) is positive. Similarly, we may compute the probability that documents without the feature will be ranked ahead of the feature as $\Pr(p \leq t)$.

The probability that optimal ranking is obtained when the ranking function $\mathcal{R}$ is $\mathcal{R}_o$ is

$$\mathcal{Q}_{opt}(p,t) = \Pr(p > t, p > t) + \Pr(p \leq t, p \leq t) = 1. \tag{18}$$

Inverse document frequency (IDF) weighting is an effective feature weighting system when no information about relevant documents is available, other than the fact that a term is or is not in the query [53], [54], [55]. Using, as a weight,

$$w = -\log t,$$

we find that the weight increases as t decreases. This provides higher weights to rarer and more specific terms.

One finds that $1 \succ 0$ if and only if $-\log t > 0$; that is, when $t > 0$, which is the case for all valid probabilities, all values of t. For the other ordering, $1 \preceq 0$ is found if and only if $-\log t \leq 0$, which never occurs, for any valid probability t.

We may estimate

$$\mathcal{Q}_{IDF}(p,t) = \Pr(p > t, t > 0) + \Pr(p \leq t, t \leq 0) = \Pr(p > t). \tag{19}$$

To estimate $\mathcal{Q}$, it is necessary to estimate probabilities for p, t and, sometimes, q (defined as $\Pr(d|\overline{rel})$, the probability that the term appears in a nonrelevant document). While these probabilities may be point probabilities such that either $p \geq t \geq q$ or the reverse holds, the probabilities may also be estimated as distributions. In some situations, our knowledge about these parameters for binary distributions may best be represented by beta distributions, describing the probability that the parameter p of the binary distribution has a certain value. We can often treat t as a point probability, since we can always know precisely what percent of documents have a particular term. The parameters p and q are not known beforehand, although for large databases, we may often make the approximation $q \approx t$ and, thus, treat q as a point probability. Using these probabilities, we may estimate $\mathcal{Q}$ for each processor. Using the local $\mathcal{Q}$ value, we may estimate $ASL = N(\mathcal{Q}\mathcal{A} + \overline{\mathcal{Q}}\,\overline{\mathcal{A}}) + 1/2$. As $\mathcal{Q}$ decreases because of the decreasingly accurate estimates, the ASL increases and the search quality worsens. Any processor with a lower $\mathcal{Q}$ will have a weaker ASL, and techniques that extract the best b documents will clearly drop in performance as fewer relevant documents are situated near the front of the ordered set of documents. The exact performance is predicted by using the earlier formulas based on parameters $n_{i,j}$ and $r_{i,j}$.

8 Conclusion

Distributing documents and media for later retrieval may have advantages for users and system administrators such as security, local administrative and financial controls, ease of update, load balancing, as well as system fault tolerance. Many existing document-based systems are inherently distributed; material in offices may be in several different locations in a given office or in any of a number of different offices. Similarly, text and media on the Internet are distributed over large numbers of servers.

Information retrieval performance may be modeled analytically, using the probability of optimality of a ranking, denoted here as $\mathcal{Q}$, and the expected position of a relevant document in an optimally ordered list of documents, with the positions scaled from 0 to 1, denoted as $\mathcal{A}$. While one can implement systems that use various retrieval algorithms, the study of analytic models of retrieval system performance allows one to understand the relationships between variables that are directly supplied by the model.

We have proposed methods for estimating the precision of the set of documents retrieved from distributed databases. Systems retrieving all documents from the distributed systems are analyzed, as are systems that only retrieve a small percentage of the documents for transmission to the merging system.

Two major domains for the application of this research are document collections distributed over the World Wide Web and digital libraries. In the digital library (DL) case, it is possible that the collection(s) a DL accesses could reside at the same site as itself. This is possible as the key point is distribution of control, not necessarily the distribution of data. In both of these situations, for a particular query, there will, quite likely, be more than one strategy available. Performance prediction models could be a great aid in providing guidance to the metasearch engine as to which strategy or strategies to use.

REFERENCES

[1] R.M. Losee, "Evaluating Retrieval Performance Given Database and Query Characteristics: Analytic Determination of Performance Surfaces," *J. Am. Soc. for Information Science,* vol. 47, no. 1, pp. 95-105, 1996.

[2] *Text Retrieval and Filtering: Analytic Models of Performance.* Boston: Kluwer, 1998.

[3] D.W. Harman, *The First Text REtrieval Conf. (TREC-1),* Nov. 1992, *Information Processing and Management,* vol. 29, no. 4, pp. 411-414, July-Aug. 1993.

[4] W. Meng, C. Yu, and K.-L. Liu, "Building Efficient and Effective Metasearch Engines," *ACM Computing Surveys,* vol. 34, no. 1, pp. 48-89, Mar. 2002.

[5] "The TREC-8 Question Answering Track Report," *Proc. Eighth Text REtrieval Conf. (TREC-8),* E.M. Voorhees and D.K.H., eds., Nat'l Inst. of Standards and Technology, 2000.

[6] A. Bookstein, "Relevance," *J. Am. Soc. for Information Science,* vol. 30, no. 5, pp. 269-273, 1979.

[7] D.R. Swanson, "Subjective versus Objective Relevance in Bibliographic Retrieval Systems," *Library Quarterly,* vol. 56, no. 4, pp. 389-398, Oct. 1986.

[8] L. Schamber, M. Eisenberg, and M.S. Nilan, "A Re-Examination of Relevance: Toward a Dynamic, Situational Definition," *Information Processing and Management,* vol. 26, no. 6, pp. 755-776, 1990.

[9] R. Tang and P. Solomon, "Toward an Understanding of the Dynamics of Relevance Judgment: An Analysis of One Person's Search Behavior," *Information Processing and Management,* vol. 34, nos. 2/3, pp. 237-256, 1998.

[10] R. Tang, J.L. Vevea, and W.M. Shaw, "Towards the Identification of the Optimal Number of Relevance Categories," *J. Am. Soc. for Information Science,* vol. 50, no. 3, pp. 254-264, 1999.

[11] K.L. Maglaughlin and D.H. Sonnenwald, "User Perspectives on Relevance Critera: A Comparison Among Relevant, Partially Relevant, and Not-Relevant Judgments," *J. Am. Soc. for Information Science and Technology,* vol. 53, no. 5, pp. 327-342, 2002.

[12] R.M. Losee and L.A.H. Paris, "Measuring Search Engine Quality and Query Difficulty: Ranking with Target and Freestyle," *J. Am. Soc. for Information Science,* vol. 50, no. 10, pp. 882-889, 1999.

[13] E.M. Voorhees, N.K. Gupta, and B. Johnson-Laird, "The Collection Fusion Problem," *Proc. Third Text REtrieval Conf. (TREC-3),* pp. 95-104, 1995.

[14] J. Savoy, A.L. Calve, and D. Vrajitoru, "Report on the TREC-5 Experiment: Data Fusion and Collection Fusion," *Proc. Fifth Text REtrieval Conf. (TREC-5),* pp. 489-502, 1997.

[15] G.G. Towell, E.M. Voorhees, N.K. Gupta, and B. Johnson-Laird, "Learning Collection Fusion Strategies for Information Retrieval," *Proc. Int'l Conf. Machine Learning,* pp. 540-548, 1995.

[16] J.C. French, A.L. Powell, and W.R. Creighton, "Efficient Searching in Distributed Digital Libraries," *ACM Digital Library,* pp. 283-284, 1998.

[17] Y. Rasolofo, "Approaches to Collection Selection and Results Merging for Distributed Information Retrieval," *Proc. Conf. Information and Knowledge Management,* pp. 191-198, Nov. 2001.

[18] W.S. Cooper, "The Formalism of Probability Theory in IR: A Foundation or an Encumbrance," *Proc. 17th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 242-248, 1994.

[19] C.T. Yu and W. Meng, *Principles of Database Query Processing for Advanced Applications.* Calif.: Morgan Kaufmann Publishers, Inc., 1998.

[20] L. Gravano, C. Chang, H. Garcia-Molina, and A. Paepcke, "STARTS: Stanford Proposal for Internet Meta-Searching," *Proc. ACM SIGMOD Conf.,* pp. 207-218, 1997.

[21] L. Gravano and Y. Papakonstantinou, "Mediating and Metasearching on the Internet," *Data Eng. Bull.,* vol. 21, no. 2, pp. 28-36, 1998.

[22] D. Hawking, "Efficiency/Effectiveness Trade-Offs in Query Processing," *ACM SIGIR Forum,* vol. 32, no. 2, pp. 16-22, 1998.

[23] B. Chidlovskii and U.M. Borghoff, "Query Translation for Distributed Information Processing on the Web," *Proc. Int'l Database Eng. and Application Symp.,* pp. 214-223, 1998.

[24] D. Kim, J. Lee, S. Lee, and C. Chung, "Heterogeneous Multimedia Database Selection on the Web," Korean Advanced Inst. of Science and Technology, Taejon, Korea, Technical Report CS/TR-2000-147, Feb. 2000.

[25] C. Baumgarten, "Probabilistic Information Retrieval in a Distributed Heterogeneous Environment," PhD dissertation, Dresden Univ. of Technology, 1999.

[26] L. Gravano and H. Garcia-Molina, "Merging Ranks from Heterogeneous Internet Sources," *Proc. 23rd Very Large Databases Conf.,* pp. 196-205, 1997.

[27] N. Green, P.G. Ipeirotis, and L. Gravano, "SDLIP + STARTS = SDARTS A Protocol and Toolkit for Metasearching," *Proc. ACM/IEEE Joint Conf. Digital Libraries,* pp. 207-214, 2001.

[28] A. Paepcke, R. Brandriff, G. Janee, R. Larson, B. Ludaescher, et al. "Search Middleware and the Simple Digital Library Interoperability Protocol," *D–Lib Magazine,* vol. 6, no. 3, Nov. 2000.

[29] J.C. French, A.L. Powell, J.P. Callan, C.L. Viles, T. Emmitt, K.J. Prey, and Y. Mou, "Comparing the Performance of Database Selection Algorithms," *Research and Development in Information Retrieval,* pp. 238-245, 1999.

[30] J.C. French, A.L. Powell, and J. Callan, "Effective and Efficient Automatic Database Selection," Univ. of Virginia, Technical Report CS-99-08, 1999.

[31] J.C. French and A.L. Powell, "Metrics for Evaluating Database Selection Techniques," Univ. of Virginia, Technical Report CS-99-19, 1999.

[32] N. Craswell, "Methods for Distributed Information Retrieval," PhD dissertation, Australian Nat'l Univ., 2000.

[33] W. Meng, K.-L. Liu, C.T. Yu, W. Wu, and N. Rishe, "Estimating the Usefulness of Search Engines," *Proc. Int'l Conf. Data Eng.,* pp. 146-153, 1999.

[34] K. Liu, C. Yu, W. Meng, W. Wu, and N. Rishe, "A Statistical Method for Estimating the Usefulness of Text Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 14, no. 6, pp. 1422-1437, Nov./Dec. 2002.

[35] N. Fuhr, "A Decision-Theoretic Approach to Database Selection in Networked IR," *ACM Trans. Information Systems,* vol. 17, no. 3, pp. 229-229, 1999.

[36] D. Hawking and P. Thistlewaite, "Methods for Information Server Selection," *ACM Trans. Information Systems (TOIS),* vol. 17, no. 1, pp. 40-76, 1999.

[37] W. Meng, Z. Wu, C. Yu, and Z. Li, "A Highly Scalable and Effective Method for Metasearch," *ACM Trans. Information Systems,* vol. 19, no. 3, pp. 310-335, July 2001.

[38] Z. Wu, W. Meng, C.T. Yu, and Z. Li, "Towards a Highly-Scalable and Effective Metasearch Engine," *World Wide Web,* pp. 386-395, 2001.
[39] J. Xu and J. Callan, "Effective Retrieval with Distributed Collections," *Proc. 21st Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 112-120, 1998.
[40] C. Baumgarten, "A Probabilistic Solution to the Selection and Fusion Problem in Distributed Information Retrieval," *Proc. 22nd Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 246-253, 1999.
[41] H. Garcia-Molina, J. Ullman, and J. Widom, *Database Systems: The Complete Book.* Prentice-Hall, Inc., 2002.
[42] M. Ozsu and P. Valduriez, *Principles of Distributed Database Systems,* second ed. Prentice-Hall, Inc., 1999.
[43] R.R. Korfhage, *Information Storage and Retrieval.* New York: John Wiley and Sons, Inc., 1997.
[44] R.E. Walpole, R.H. Myers, and S.L. Myers, *Probability and Statistics for Engineers and Scientists.* Saddle River, New Jersey: Prentice Hall, 1998.
[45] W.B. Croft and D.J. Harper, "Using Probabilistic Models of Document Retrieval Without Relevance Information," *J. Documentation,* vol. 35, no. 4, pp. 285-295, 1979.
[46] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* New York: Addison Wesley, 1999.
[47] S.E. Robertson and K. Sparck Jones, "Relevance Weighting of Search Terms," *J. Am. Soc. for Information Science,* vol. 27, pp. 129-146, 1976.
[48] C. Van Rijsbergen, *Information Retrieval,* second ed. London: Butterworths, 1979.
[49] J.W. Pratt, H. Raiffa, and R. Schlaifer, *Introduction to Statistical Decision Theory.* Mass.: MIT Press, 1995.
[50] L. Kleinrock, *Queueing Systems, Volume I: Theory.* New York: Wiley Interscience, 1975.
[51] D. Hawking, N. Craswell, P. Bailey, and K. Griffiths, "Measuring Search Engine Quality," *Information Retrieval,* vol. 4, no. 1, pp. 33-59, 2001.
[52] C.T. Yu and G. Salton, "Precision Weighting—An Effective Automatic Indexing Method," *J. ACM,* vol. 23, no. 1, pp. 76-88, 1976.
[53] K. Sparck Jones, "A Statistical Interpretation of Term Specificity and Its Application in Retrieval," *J. Documentation,* vol. 28, no. 1, pp. 11-21, 1972.
[54] C.T. Yu and G. Salton, "Effective Information Retrieval Using Term Accuracy," *Comm. ACM,* vol. 20, pp. 135-142, 1977.
[55] W.B. Croft and D. Harper, "Using Probabilistic Models of Document Retrieval without Relevance Information," *J. Documentation,* vol. 35, no. 4, pp. 285-295, Dec. 1979.

Robert M. Losee received the PhD degree from the University of Chicago in 1986. He is a professor in the School of Information and Library Science at the University of North Carolina-Chapel Hill. His research interests include information retrieval, information, reasoning systems, organizing information, and decision making. Dr. Losee is the author of two books: *Text Retrieval and Filtering: Analytic Models of Performance* (Kluwer) and *The Science of Information: Measurement and Applications* (Academic Press). He is also the coauthor of another book, *Research and Evaluation for Information Professionals* (Academic Press).

Lewis Church Jr. received the BS degree in computer science from Virginia Tech, the MS degree in applied science from the College of William and Mary, and the ScM degree in computer science from Brown University. He is a PhD student in the School of Information and Library Science at the University of North Carolina-Chapel Hill. His research interests include distributed and parallel information retrieval (IR), analytic models of performance, machine learning and soft computing, optimal query processing strategies, and the computational complexity of IR algorithms.