Distributed Text Retrieval From Overlapping Collections

Milad Shokouhi **Justin Zobel** **Yaniv Bernstein**

School of Computer Science and Information Technology
RMIT University,
PO Box 2476V, Melbourne, 3001, Australia
Email: {milad,jz,ybernste}@cs.rmit.edu.au

Abstract

In standard text retrieval systems, the documents are gathered and indexed on a single server. In distributed information retrieval (DIR), the documents are held in multiple collections; answers to queries are produced by selecting the collections to query and then merging results from these collections. However, in most prior research in the area, collections are assumed to be disjoint. In this paper, we investigate the effectiveness of different combinations of server selection and result merging algorithms in the presence of duplicates. We also test our hash-based method for efficiently detecting duplicates and near-duplicates in the lists of documents returned by collections. Our results, based on two different designs of test data, indicate that some DIR methods are more likely to return duplicate documents, and show that removing such redundant documents can have a significant impact on the final search effectiveness.

Keywords: Distributed information retrieval, duplicate and near-duplicate detection, similarity measurement, search engines

1 Introduction

In standard text retrieval systems, the collection of documents is indexed at a single location and made available to users through a search interface. Such centralized information retrieval (IR) is efficient and effective when the documents can be gathered together, but such consolidation is not always possible. On the web, for example, many documents are not crawlable and can only be accessed by that particular site's search interface. Such documents form the *hidden web*, which has been reported to be many times larger than the fraction of the web that can be crawled (Bergman 2001). Also, crawling the web can be prohibitively slow; some documents are held on servers whose delivery speed makes crawling impractical.

A solution is provided by distributed information retrieval (DIR), in which documents held on multiple servers are made available through a single interface. Distributed search over multiple collections—also known as federated search—has been an active area of research for over a decade. In addition to making hidden-web documents centrally accessible, there are other gains; for example, with DIR the costs of crawling can be avoided, and distributed indexes can be updated more easily than the alternatives.

In DIR, queries are submitted to a central component, known as the *broker*. For efficiency, the broker sends the query to only a limited number of collections—creating the problem of collection selection. In order to select suitable collections for each query, the broker should acquire information about each collection in advance—creating the problem of collection representation. A collection's *representation set* is the broker's knowledge about each collection. Queries are compared with representation sets, and collections that are more likely to return relevant documents are selected (Callan, Lu & Croft 1995, Gravano, Garcia-Molina & Tomasic 1999). The selected collections return their answers to the broker, which merges the results and presents them to the user (Callan et al. 1995, Si & Callan 2003*b*)—creating the problem of result merging.

In *cooperative* environments, collections provide the broker with comprehensive information about their indexes, typically comprised of data such as lexicon statistics and collection size, allowing the broker to effectively select suitable collections for a query (Allan et al 2003, Meng, Yu & Liu 2002). In *uncooperative* environments, collections do not provide such information. To identify likely collections, the broker must first create sample surrogates by downloading a limited number of documents from collections (Callan, Connell & Du 1999, Callan & Connell 2001, Ipeirotis & Gravano 2004, Ipeirotis & Gravano 2002). In *semi-cooperative* environments, the broker has incomplete knowledge about the collections. However, it might receive supplementary information such as document scores from collections, allowing more accurate results merging (Callan et al. 1995, Si & Callan 2003*b*).

Many collection representation techniques (Callan et al. 1999, Callan & Connell 2001, Ipeirotis & Gravano 2004, Ipeirotis & Gravano 2002, Shokouhi, Scholer & Zobel 2006), collection selection techniques (Callan et al. 1995, Gravano et al. 1999, Nottelmann & Fuhr 2003, Si & Callan 2003*a*, Si & Callan 2004, Yuwono & Lee 1997, Zobel 1997), and result merging techniques (Callan et al. 1995, Kirsch 2003, Si & Callan 2003*b*) have been investigated. However, in almost all reported experiments, it is assumed that collections do not overlap, so that any given document is only available in one collection.

In practice, however, this assumption is frequently valid. For example, many research articles are indexed by both the ACM portal[1] and the IEEE Xplore[2] digital libraries. Therefore, a broker that sends queries to both online collections may receive many duplicate documents. Detecting such duplicates is far from straightforward, as they may be delivered from different URLs and have slight changes, such as in their metadata or presentation, that do not

[1] http://portal.acm.org
[2] http://ieeexplore.ieee.org

alter the content but mean that the documents are not identical. Retrieval of all candidate documents by the broker to check for duplication imposes significant expense—in standard DIR the broker need not inspect the documents—while, as we show, failure to eliminate duplicates can significantly degrade the quality of answers returned.

Although distributed search among overlapped collections has been suggested as one of the major issues in DIR (Allan et al 2003, Meng et al. 2002), the problem has not been thoroughly explored. We have previously proposed that *grainy hash vectors* (GHVs) can be used for removing duplicate and near-duplicate documents from result lists (Bernstein, Shokouhi & Zobel 2006). However, these experiments used a centralized index. Moreover, in this work the performance of DIR techniques in the presence of duplicates and near-duplicates is not comprehensively investigated.

In this paper, we use GHVs on distributed testbeds to remove duplicate and near-duplicate documents from the results. We also analyze the behavior of well-known DIR collection selection and result merging strategies on overlapped collections. As a basis for this work, we propose two mechanisms for creating experimental testbeds from standard test collections. One, based on windowing, is somewhat artificial but allows precise setting of the level of duplication, thus allowing us to explore the impact of parameters in a controlled way; the other, based on query result sets, generates collections with some consistency of topic. To our knowledge, such a study on overlapped collections has not previously been undertaken.

In our experiments, we find that GHVs are effective at identifying duplicates in answer sets, and that removal of such duplicates can have a substantial impact on effectiveness. We also find that relative performance of the collection selection and result merging methods depends to some extent on the testbed used, demonstrating that care is required in the development of resources and interpretation of outcomes for such research questions.

2 Distributed information retrieval

In DIR, the broker receives each query, distributes it to selected servers, and merges results to give a consolidated list to return to the user. Before the broker can begin processing queries, it must gather a representation set for each collection. In cooperative protocols such as STARTS (Gravano, Chang, Garcia-Molina & Paepcke 1997), collections provide the broker with comprehensive information about their indexes including the lexicon statistics and the number of documents. However, access to such a representation set for each collection may not be possible; in uncooperative environments, the broker must download a limited number of documents from each collection and use them as a representation set.

Query-based sampling (QBS) was proposed by Callan et al. (1999) for uncooperative environments. In this approach, an initial query is selected from a list of frequent keywords and is submitted to a collection. The top N documents returned by the collection are downloaded by the broker and the next query is picked, usually at random, from the contents of these documents. This process continues until a sufficient number of documents (k) has been downloaded. Callan et al. (1999) claimed that $N = 4$ and $k = 300$ are suitable values; in contrast, we have suggested that the rate of discovery of new terms in the downloaded documents is a suitable indicator of when sampling should stop (Shokouhi et al. 2006). In this paper, we use the values suggested by Callan et al. (1999), to make our results comparable to related work in this area (Nottelmann & Fuhr 2003, Si & Callan 2003*b*, Si & Callan 2003*a*, Si & Callan 2004).

Once representation sets are created, the broker can use them to pass queries to collections that are deemed as likely to contain relevant documents. In some collection selection techniques, the broker uses variants of standard IR measures to compute the similarity of a query with collection representation sets. CORI (Callan et al. 1995), GlOSS (Gravano et al. 1999) and CVV (Yuwono & Lee 1997) are a few well-known examples of such approaches. Among these, CORI was reported to produce the highest precision (Craswell, Bailey & Hawking 2000, Powell & French 2003, Rasolofo, Abbaci & Savoy 2001).

In CORI, the belief $P(t|c)$ for observing a query term t in collection c is computed as below:

$$P(t|c) = d_b + (1 - d_b) \times T_c \times I_c$$

$$T_c = d_t + (1 - d_t) \cdot \frac{\log(f_{t,c} + 0.5)}{\log(max_c(f_t) + 1.0)}$$

$$I_c = \frac{\log(\frac{N+0.5}{cf})}{\log(N + 1.0)}$$

where $f_{t,c}$ is the document frequency of t in c and cf is the number of collections containing the term t. d_t is the minimum term frequency component while d_b is the minimum belief component when t occurs in c. d_t and d_b are respectively set to 0.4 and 0.5 by default. N represents the total number of collections while $max_c(f_t)$ is the maximum document frequency in c. The final weight of a collection c for query Q is calculated by summing up the computed beliefs $P(t|c)$ for all query terms $t \in Q$.

Although more recent studies suggest that CORI is generally worse than other methods on many testbeds (D'Souza, Zobel & Thom 2004, Si & Callan 2003*a*), it is still used frequently for collection selection research (Si & Callan 2004, Si & Callan 2005, Avrahami, Yau, Si & Callan 2006). Thus we use CORI as one of our collection selection methods in this paper.

In recent years, new collection selection algorithms have been shown to produce better results than CORI (Hawking & Thomas 2005, Si, Jin, Callan & Ogilvie 2002, Si & Callan 2003*a*, Si & Callan 2004, Si & Callan 2005) on some testbeds. In HARP (Hawking & Thomas 2005), the anchor texts available in a large crawled repository are used to create the representation sets, which for each collection consists of the anchor texts of URLs available in the crawled data that are targeting the collection. This is similar to Q-pilot (Sugiura & Etzioni 2000).

ReDDE (Si & Callan 2003*a*) ranks collections according to the estimated number of relevant documents. The broker creates a central index of all sampled documents, then each submitted query is evaluated on this index before being sent to collections. The number of relevant documents in each collection is estimated from the contribution of collections in the top-ranked documents. UUM (Si & Callan 2004) estimates the probability of relevance of documents inside each collection using training queries and their related relevance judgments. UUM was reported to produce better results than ReDDE (Si & Callan 2004).

RUM (Si & Callan 2005) is a variant of UUM that also considers the search effectiveness of collections. As in ReDDE, RUM maintains a central index of all sampled documents on the broker. In the training stage, the documents returned by collections for queries are compared to those ranked by the central index. In addition, the broker downloads a few

top-ranked documents from each collection and calculates their weights in the central index. Based on the weights of downloaded documents for the training queries, the broker then approximates the search effectiveness of each collection. Si & Callan (2005) suggest that RUM can slightly outperform UUM.

RUM and UUM have significant drawbacks: both require a set of training queries and relevance judgments. RUM also downloads documents from each collection for the training queries. In practical situations, relevance judgments may be costly and downloading documents might be infeasible. Therefore, we use ReDDE (Si & Callan 2003*a*) as a practical representative of recent collection selection algorithms.

Selected collections return their answers to the broker. The broker then merges the results and represents them to the user.

The document score values reported by collections are not comparable as they are computed by different retrieval models and rely on inconsistent lexicon statistics. The goal of result merging algorithms is to calculate a global score for each document that is comparable to the scores of documents returned by other collections.

In CORI result merging (Callan et al. 1995), the global score D_G of a document d returned by a collection (c) is calculated according to its normalized document score (D') and collection score (C'). The former is the *collection-specific* weight of d reported by its original collection (c) and the latter is the weight of c calculated by the broker.

$$C' = \frac{(C - C_{min})}{(C_{max} - C_{min})}$$

$$D' = \frac{(D - D^c_{min})}{(D^c_{max} - D^c_{min})}$$

$$D_G = \frac{D' + 0.4 \times D' \times C'}{1.4}$$

where D^c_{min} and D^c_{max} are respectively the minimum and maximum document scores reported by collection c, and C_{min} and C_{max} are the minimum and maximum weights that can be assigned to any collection by the broker during collection selection.

SSL (Si & Callan 2003*b*) uses a semi-supervised learning method to create a model for each collection that maps document scores into global scores. SSL creates a central index of all sampled documents. For a query, some of the documents returned by any collection c might be already available in the central index. SSL compares the central weights of such overlap documents with the reported scores from c and then approximates the weights of other documents that are not available in the central index.

When collections use an identical retrieval model, all overlap documents can be used to train a single linear model that maps collection-specific scores into approximated global scores. In such a scenario, for an overlap document d returned from any selected collection c, two scores are available: D is the score reported by the original collection and D_S is the weight of document computed by the central, sample-based index.

Using the values for D and D_S, SSL trains a linear model that converts the reported scores by collections to their approximated central scores, with parameters a and b used to combine scores as follows:

$$D_G = a \times D + b \times D_S \times C$$

where C is the weight of collection c from which d is drawn. Given this information, SSL can accurately approximate the central scores of documents. The central index can be assumed to be representative of the global information, so the weights of documents in the central index are likely to be representative of their global scores.

For simplicity, we assume that all collections in our experiments are using INQUERY (Callan, Croft & Harding 1992) as their retrieval model. Thus, we use SSL single-model as one of our merging methods.

CORI and SSL are intended for semi-cooperative environments where document scores are broadcast by collections. In the absence of document scores, they calculate the pseudoscores of documents as described by Si et al. (2002). Other merging strategies (Kirsch 2003, Si et al. 2002) typically follow the same strategy but are not as well-known as CORI and SSL. Therefore, in our experiments we use CORI and SSL for merging.

DIR merging is not equivalent to data fusion or the merging problem in metasearch (Croft 2000, Fox & Shaw 1994, Lee 1997). In data fusion, different ranking functions are applied to the same collection (Meng et al. 2002). In the presence of multiple collections, queries are usually sent to all of them without collection selection. Therefore, in data fusion or metasearch merging, collection representation sets are not usually required. Also, documents are merged solely based on their ranks or reported scores by collections.

3 Overlapping collections

Distributed retrieval from overlapped collections has been described as one of the current challenges in IR (Allan et al 2003, Meng et al. 2002). However, all of the methods discussed above assume that collections do not have overlapped documents or that the number of duplicates is negligible.

Metasearch engines such as ProFusion (Gauch, Wang & Gomez 1996), MetaCrawler (Selberg & Etzioni 1997), and Grouper (Zamir & Etzioni 1999) remove duplicate documents from the results by aggregating those that point to the same URL. The rank of a duplicate document in the final result is calculated according to its position in the ranklists of different search engines. Typically, this involves use of methods such as CombSUM or ComMNZ (Fox & Shaw 1994), or other similar approaches (Lee 1997). Some researchers argue that such methods cannot be defined in the context of DIR and should be described in the broader category of metasearch (Si & Callan 2003*b*).

These metasearch methods have two major drawbacks; they cannot detect and remove near-duplicates and they are unable to distinguish exact duplicate documents with different URLs (mirror URLs). Moreover, a recent study (Wu & McClean 2006) suggests that when the rate of overlap between the final results of collections is less than 60%, the performance of such methods significantly degrades.

To our knowledge, the only discussion of removal duplicates and near-duplicates during merging in DIR is that given by Bernstein et al. (2006). This approach is discussed in detail in the following sections.

Duplicate documents can also be avoided by using an overlap-aware collection selection method. Hernandez & Kambhampati (2005) introduced COSCO, which considers the rate of overlap among collections during collection selection. For a query, COSCO does not select two collections that are likely to return many identical documents. The approach, although interesting, has defects. COSCO requires a large number of training queries to learn the rates of overlap between collections for each topic. In addition, COSCO was not tested for detection of near-duplicate

documents, which, as discussed below, is a much more challenging problem (Zobel & Bernstein 2006).

We argue that removal of duplicates during merge is more appropriate because overlap-aware collection selection methods may be lossy. That is, ignoring a collection that includes unvisited relevant documents can affect the final precision. In contrast, if duplicate removal occurs during merging, all relevant documents can be retrieved by the broker. We now explore methods for detection and removal of duplicates.

4 Detection of duplicates and near-duplicates

There are many sources of duplication in text collections. For example, a crawl of documents harvested from the web may yield duplicates due to factors including URL aliasing; copies of the same document held at several mirrors; each author of a paper placing a copy on their website; republication of news stories by multiple online newspapers; and commercial websites presenting local copies of public documents.

Exact copies are easy to detect, but many duplicates are not exact copies. A newspaper that republishes a newswire article may edit it to reflect local knowledge, and the page context such as advertising and 'related story' links is likely to be different. Some stories are regularly recycled, such as birthday notices and Groundhog Day features. Policy documents and legislation may differ little from jurisdiction to jurisdiction, as policymakers adopt models from elsewhere. Ultimately, the question of whether two documents are duplicates is highly application-dependent—for example, in some contexts two documents that differ only in publication date may be regarded as having a significant difference. However, we have found that, in the context of search, mechanisms for detecting duplication such as those described in this section are consistent with user judgements as to whether documents are duplicates or near-duplicates. (Bernstein & Zobel 2005, Zobel & Bernstein 2006).

In the context of search, duplicate detection can take place during either indexing or retrieval. At indexing time, duplicate detection involves processing the entire collection to find pairs of documents that appear to be duplicates or near-duplicates (Manber 1994, Brin, Davis & García-Molina 1995, Broder, Glassman, Manasse & Zweig 1997, Fetterly, Manasse & Najork 2003, Bernstein & Zobel 2004). In a DIR system, such an approach is unlikely to be feasible, particularly in uncooperative environments. Thus we must focus on methods that can be used to eliminate duplicates from answer lists. However, we wish to avoid adding significant costs to the query evaluation mechanism. A DIR duplicate-detection mechanism that involved fetching all of the answer documents from each collection is not attractive.

As we have described elsewhere (Bernstein et al. 2006), the most suitable methods described in previous literature are based on computing a brief descriptor of each document. In a semi-cooperative environment, each collection could return these descriptors, allowing efficient duplicate detection. We now summarize our previous analysis of prior methods.

A descriptor-based approach that is superficially attractive is to use deterministic term extraction to identify terms in each document that are deemed likely to be indicative of duplication (Chowdhury, Frieder, Grossman & McCabe 2002, Ilyinski, Kuzmin, Melkov & Segalovich 2002, Cooper, Coden & Brown 2002, Conrad, Guo & Schriber 2003, Kolcz, Chowdhury & Alspector 2004). The assumption is that near-duplicates will share an exact set of such terms, so that hashing them will produce a descriptor that can be matched extremely fast. This term extraction process must take place at index time; otherwise, query processing would be much more expensive.

However, such approaches can only succeed if they are effective at identifying indicative terms. A problem is that a common design principle in these approaches is to select terms that are significant within the document, that is, are relatively rare across the corpus. In DIR, there are no cross-corpus statistics, and even in centralized retrieval the corpus is not static; thus the same term extraction method will produce different term sets in different collections. It is not at all clear that such term extraction methods can be reliable in the absence of global statistics. Another problem is that it is not known how robust these techniques are, as a single difference between the term sets means that near-duplication is not detected. A proposed solution is to return multiple hashes per document (Pugh & Henzinger 2003, Kolcz et al. 2004), but in the absence of global statistics it is still unclear that this can be effective.

The other widely-investigated approach to duplicate detection is to use *chunks* (Hoad & Zobel 2003), an approach that has been proposed for a variety of applications (Manber 1994, Lyon, Malcolm & Dickerson 2001, Brin et al. 1995, Conrad et al. 2003, Broder et al. 1997, Bernstein & Zobel 2004, Bernstein & Zobel 2005). In chunking, each document is parsed into strings of text, each typically of some fixed length or some fixed number of words. A pair of documents is deemed to be duplicated if they share a sufficient number of chunks. Chunk-extraction can be selective or exhaustive; some methods use only a few chunks per document, while in others every word occurrence is the start of a new chunk. Given a set of chunks, the *resemblance* between two documents (Broder et al. 1997) can be defined as:

$$R(d, d') = \frac{|\hat{d} \cap \hat{d'}|}{|\hat{d} \cup \hat{d'}|}$$

where $\hat{d}$ is the set of chunks extracted from document d.

A complete set of chunks is not a particularly useful document descriptor in DIR. However, the method of Fetterly et al. (2003), which we call *minimal-chunk sampling*, can be used for chunk selection. By use of an appropriate class of hash functions, by use of ρ functions from this class, chunks can be sampled in an unbiased way. Each chunk is hashed with each function, and for each of the ρ functions the smallest observed hash value yielded by any chunk is kept. These hash values can be used as proxies for chunks in determining resemblance. Fetterly et al. (2003) use $\rho = 84$, giving 84 32-bit hashes, a total of 336 bytes. For DIR, this is a significant data volume to transmit per document. It is therefore attractive to seek a more compact alternative.

5 Grainy hash vectors

A grainy hash vector (GHV) (Bernstein et al. 2006) is a form of minimal-chunk sampling method. However, it has features derived from deterministic term extraction techniques, and the vectors are only a few bytes, making it suitable for DIR. Bernstein et al. (2006) list the major benefits of GHVs as below:

- Efficiency; the hash vectors are designed to fit into a single machine word of 32 or 64 bits.
- Theoretical foundation; GHVs are based on mathematical principles that are amenable to analysis.

- Fast comparison; thousands of document vectors can be compared in a few milliseconds using bit-parallelism techniques.
- Robust comparison; GHVs do not change significantly in the presence of small differences in documents.

A GHV of n bits consists of ρ w-bit hashes and is represented as follows:

$$\rho(n, w) = \left\lfloor \frac{n}{w} \right\rfloor$$

Each of the w-bit hashes is produced by a separate minimal-chunk sampling technique. Therefore, for a 64-bit GHV with $w = 2$, there are 32 2-bit hashes that are merged into the vector. The value of w should be a factor of n, to avoid having unused bits.

For small values of w, there is a significant probability of collision between the outputs of different minimal hashes. For $w = 2$, for example, it is likely that most of the minimal hashes will be the bit pair 00 (hash values for different chunks are sorted and the 2 least significant bits of the smallest hash value are selected). On the other hand, using large values for w may be computationally expensive, and reduces the number of hashes per vector. Therefore, GHV initially uses minimal sampling to produce ρ 32-bit hashes for each document. Then the least significant w bits of each hash value are used to give the GHV.

For a pair of documents with resemblance r ($0 \leq r \leq 1$), the probability of having the same hash value at any given position of their GHVs is:

$$\phi(r; w) = r + (1 - r)(2^{-w})$$

If two documents have resemblance r, they will at least have r of their hash vectors in common. Thus, the probability of a hash match between their hash vectors is r. The second component calculates the probability of having the same hash value while the two source chunks are different.

Assuming that for each bit the in a vector the probability of being 0 or 1 is independent the other bits, we can conclude that the number of matches between two vectors with resemblance r follows a binomial distribution $Bi(\rho(n, w), \phi(r; w))$, and thus

$$P(X = k) = \binom{\rho}{k} \phi^k (1 - \phi)^{\rho - k}$$

where $P(X = k)$ is the probability of having k matches between the vectors of two documents with resemblance r.

For each query, collections send a GHV per document they return to the broker. The broker detects and removes duplicates or near-duplicates according to the number of mismatches between any given pair of vectors. If $w = 1$, this amounts to counting the number of bits that match in the vectors; if the number is sufficiently high, the documents are deemed to be likely to be duplicates or near-duplicates.

A critical question, then, is what is the minimum number of mismatches between two GHVs if the documents they represent are not near-duplicates? That is, a threshold for the number of mismatches needs to be set to minimise the number of false misses. (False misses are much more acceptable than false matches in this application; the former means that duplicate information is presented, while the latter means that novel information is lost.) Bernstein et al. (2006) compared the accuracy of GHVs with the duplicate detection method DECO (Bernstein & Zobel 2004), for different values of n and w. They found that GHVs can effectively detect duplicates or near-duplicates using $n = 64$, $w = 2$, and a threshold of 8 mismatches, without significant false misses. We apply the suggested values by Bernstein et al. (2006) and use 32 minimal-chunk hashes to create 32 2-bit hash values for each document.

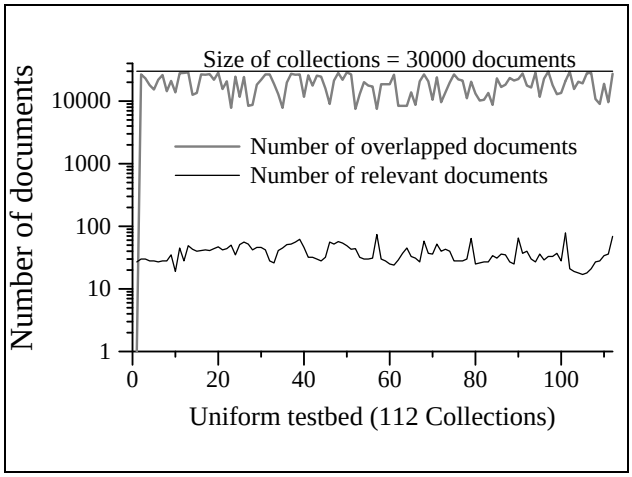

Figure 1: *Collection sizes, distribution of relevant documents and the amount of overlap between consequent collection pairs in the uniform testbed.*

We now compare the performance of collection selection and result merging algorithms on collections that overlap. We also analyze the impact of removing duplicates and near-duplicates from the list of results in terms of search effectiveness, using GHVs to determine whether documents are likely to be duplicates.

6 Testbeds

In current well-known DIR testbeds such as trec123-100col-bysource or trec4-kmeans,[3] the degree of overlap between collections is intentionally set to zero. Therefore, current testbeds are not suitable for evaluating DIR methods in the presence of overlap. We create three testbeds using the documents available in the TREC GOV data (Craswell & Hawking 2002). These testbeds are as follows:

uniform-112col-dups (uniform): This testbed is comprised of 112 collections, each containing 30 000 documents, created using a sliding window on the TREC GOV documents. The first 30 000 documents comprise the first collection. Then a random percentage R ($R \geq 25\%$) is picked. The second collection is created from the last $R\%$ of the first collection and the next documents from the GOV corpus to a total size of 30 000 documents. The rest of the collections are generated in the same sliding window manner.

Figure 1 shows the distribution of relevant documents for TREC topics 551–600 among the collections in this testbed. As was expected, relevant documents are spread uniformly among collections. The figure also shows the number of documents in each collection that are shared with the previous collection. The rate of overlap varies from 25% to 99%.

skewed-115col-dups (skewed): Collection selection algorithms show variable performance on different testbeds. Some approaches such as CORI are found to be less effective when the distribution of collection sizes is skewed (D'Souza et al. 2004, Si & Callan 2003*a*). To create such a testbed, we adapt the approach Si & Callan (2003*a*) used for deriving

[3] Available at `www.cs.cmu.au/~callan/Data`.

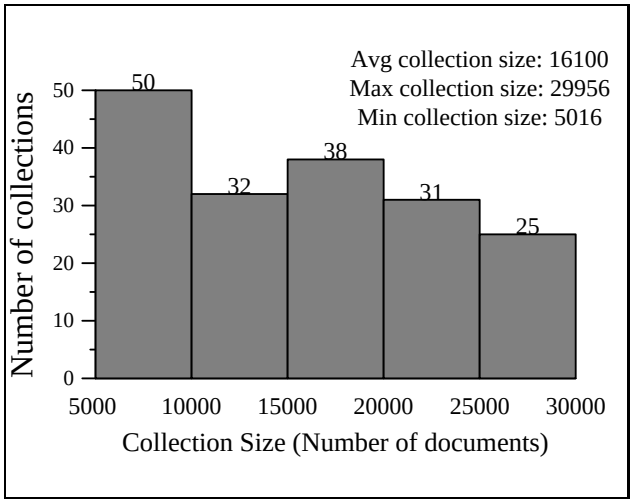

Figure 2: *Collection sizes in the Qprobed testbed.*

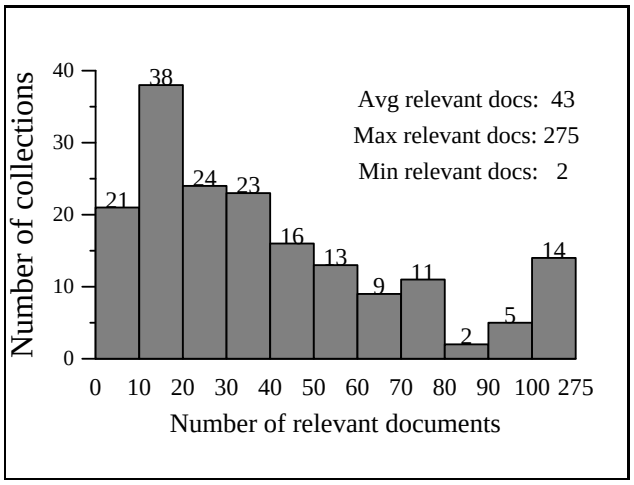

Figure 3: *Distribution of relevant documents in the Qprobed testbed.*

the so-called *representative testbed* from the trec123-100col-bysource documents. Every tenth collection in the uniform testbed starting from the first collection is collapsed into a single large collection. The same procedure is repeated starting with the second and third collections and another two large collections are created. The testbed thus contains the uniform testbed plus an additional three large collections.

Qprobed-176col-dups (Qprobed): 176 collections have been generated by passing 200 probe queries to an index of the TREC GOV documents. Queries are the most frequent single terms in a query log supplied by a major search engine, of queries with a highly ranked answer in the `.gov` domain. For each query, a random number of results between 5 000 and 30 000 are extracted and gathered as a collection. Queries that return less than 5 000 documents are discarded. The average size of collections in this testbed is 16 100 documents while the largest and smallest collections contain 29 956 and 5 016 documents. The distribution of collection sizes and the number of relevant documents among collections are depicted in Figures 2 and 3. Considering Figure 3, for example, there are 50 collections that have between 5 000 and 10 000 documents, and there are 38 collections that contain between 10 and 20 relevant documents each for TREC topics 551–600.

The degree of overlap among collections in this testbed is diverse. Figure 4 shows that there are

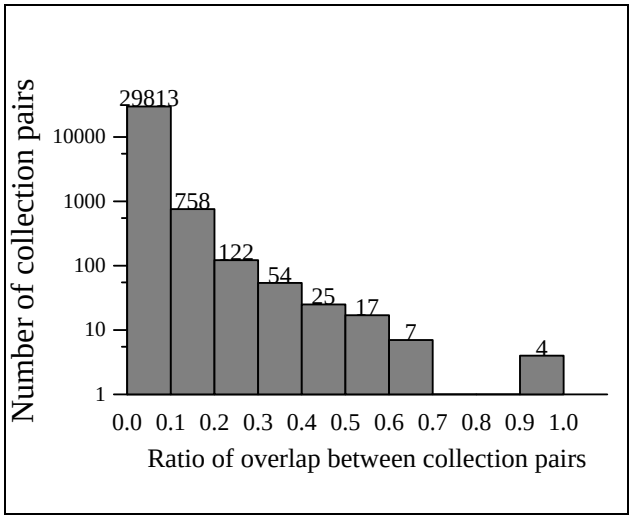

Figure 4: *Degree of overlap between collections in the Qprobed testbed.*

29 813 collection pairs that have less than 10% overlap while the rate of overlap for four collection pairs is close to 100%.

The ideal testbed for these experiments would be complete crawls of websites from the hidden web, but by construction such crawls are not easily available. Another good testbed would be crawls of sites known to contain high levels of overlap. There are such sites in the TREC GOV data (in which we believe more than half the pages are duplicates or near duplicates), but identifying them is far from straightforward. It is for these reasons that other ways of forming testbeds are of interest. The uniform and skewed collections are somewhat artificial, but they allow exploration of the effectiveness of DIR methods as a function of the extent of overlap. The Qprobed testbed consists of collections with some degree of internal consistency, and the collections are comprised of documents selected by a method that is fully independent of the techniques we are exploring.

In the next section we investigate the effectiveness of different DIR algorithms on the discussed testbeds.

7 Experimental results

On each testbed, we explore three combinations of collection selection and result merging algorithms. These are CORI-CORI, ReDDE-CORI, and ReDDE-SSL where the first part specifies the collection selection method and the last part represents the merging algorithm used for the experiments.

We use query-based sampling to gather 300 documents from each collection. The sampled documents from each collection are used as its representation set. In addition, aggregating all representation sets creates a central sample index that is used by SSL result merging (Si & Callan 2003*b*). All experiments reported in this paper use the Lemur toolkit.[4] Methods are compared according to their precision at the top n returned documents ($P@n$). For each query, at most 1 000 answers are returned.

We use a 64-bit GHV with $w = 2$ and a threshold of 8 mismatches out of possible 32, as recommended by Bernstein et al. (2006). Collections provide the broker with a GHV for each answer they return. They also publish the document scores of their returned answers that then will be used by the broker for result

[4] `http://www.lemurproject.org`

merging. That is, we assume a semi-cooperative environment.

Duplicate and near-duplicate documents in an answer list can be regarded as redundant and irrelevant, as they do not add to the user's knowledge. We use GHV to identify duplicate and near-duplicate documents in the final merged results. Then we compare the search effectiveness of two scenarios where in the first one, duplicates are considered as irrelevant and in the second set duplicates are removed. In the first scenario, the broker removes those documents that GHV detects as redundant from the list of results. The upper tables for each combination contain the precision values for this scenario.

In the second scenario, the returned duplicate and near-duplicate documents are considered as irrelevant. To avoid bias, we do not use GHV to judge whether documents are duplicates and near-duplicates; instead, the broker uses a list of near-duplicate documents in the TREC GOV that are identified by DECO (Bernstein & Zobel 2004) duplicate detection method.[5]

One might expect that the lists of duplicate documents detected by GHV and DECO would be significantly different. However, our investigations show that 93% of documents that are identified as near-duplicate by GHV are also detected by DECO, while, 72% of all documents that are identified by DECO are detected by GHV. The high rate of accuracy and coverage of GHVs for detecting near-duplicate documents is consistent with the reported values by Bernstein et al. (2006) on a centralized index. On average, over all experiments, use of GHVs detects 9 near-duplicates per query while this number is 12.5 for DECO. Considering that 1 000 documents are collected per query, the difference of 2.5 documents has negligible impact on the final effectiveness. Also, this is further evidence that GHVs avoid false misses.

Note that DECO removes exact-duplicate pairs according to their document identifiers. All exact-duplicates are detected and removed by GHV as there is no mismatch between their vectors.

We use the t-test to measure the statistical significance of difference between results in the presence and absence of duplicates. The differences at 0.95, 0.99 and 0.999 confidence intervals are respectively specified with $*$, †, and ‡. The baseline for all experiments against which significance is measured is the effectiveness of retrieval when the returned duplicates and near-duplicates are considered as irrelevant. The differences are measured against the scenario in which redundant documents are removed by GHV.

Table 1 shows the effectiveness of different combinations on the uniform testbed. Cutoff (CO) values indicate the number of collections selected for each query. Comparing the first two sets of experiments (CORI-CORI and ReDDE-CORI) suggests that CORI and ReDDE collection selection methods have similar performance on the uniform testbed. Considering that all collections in this testbed contain an identical number of documents, the results are consistent with the previous observations of Si & Callan (2003*a*), which suggest the performance of CORI and ReDDE are similar when the distribution of collection sizes is not skewed. As in experiments on collections that do not overlap (Si & Callan 2003*b*), SSL merges results more effectively than does CORI on the uniform testbed.

The impact of removing duplicates and near-duplicates becomes more apparent as the cutoff value grows. In addition, the effectiveness of ReDDE-SSL improve more significantly than the other combinations when duplicates are removed. This suggests that selecting more collections and using ReDDE-SSL combination increase the likelihood of finding duplicate documents in the final results. The probable explanation is that ReDDE-SSL is more effective than the other methods at giving the same document the same score when it is present in multiple collections. That is, if one copy of a duplicate document is fetched, then under ReDDE-SSL the other copy is also likely to be fetched. (The number of duplicates and near-duplicates returned by different combinations is shown in Table 4 and is discussed later.)

Table 1: *The impact of removing duplicate and near-duplicate documents. Results are obtained by running the TREC topics 551–600 (title) on the "uniform" testbed. CO represents the cutoff value.*

	P@5	**P@10**	**P@15**	**P@20**
Duplicates Removed (CORI-CORI)				
CO3	0.0840	0.0620	0.0507	0.0470
CO5	0.0840	0.0700	0.0587	0.0510
CO10	0.1760*	0.1240†	0.1093*	0.0970†
CO20	0.1959†	0.1449†	0.1252†	0.1153†
Duplicates Irrelevant (CORI-CORI)				
CO3	0.0800	0.0600	0.0493	0.0460
CO5	0.0800	0.0680	0.0573	0.0500
CO10	0.1560	0.1120	0.1027	0.0890
CO20	0.1592	0.1306	0.1129	0.1031
Duplicates Removed (ReDDE-CORI)				
CO3	0.0907	0.0542	0.0431	0.0354
CO5	0.1250	0.0729	0.0583	0.0521
CO10	0.1583	0.1083	0.0958	0.0857
CO20	0.1625	0.1333†	0.1139*	0.1083†
Duplicates Irrelevant (ReDDE-CORI)				
CO3	0.0907	0.0542	0.0431	0.0354
CO5	0.1208	0.0708	0.0583	0.0479
CO10	0.1500	0.0979	0.0917	0.0833
CO20	0.1542	0.1187	0.1056	0.1010
Duplicates Removed (ReDDE-SSL)				
CO3	0.1167	0.0667	0.0528	0.0490
CO5	0.1583	0.1062*	0.0861	0.0719
CO10	0.1875*	0.1479*	0.1347*	0.1177*
CO20	0.2042*	0.1979*	0.1667‡	0.1458†
Duplicates Irrelevant (ReDDE-SSL)				
CO3	0.1167	0.0667	0.0514	0.0479
CO5	0.1417	0.0979	0.0847	0.0708
CO10	0.1759	0.1375	0.1264	0.1135
CO20	0.1792	0.1792	0.1500	0.1323

Table 2 shows similar results on the skewed testbed. For small cutoff values, ReDDE significantly outperforms CORI, because the three largest collections contain many relevant documents that get high ranks by ReDDE for the majority of queries. The better performance of ReDDE is not surprising as it is designed for situations where the distribution of collection sizes is skewed (Si & Callan 2003*a*). SSL is again the dominant merging algorithm and produces better results than CORI.

As in previous experiments on the uniform testbed, removing duplicate documents changes the final precision more significantly for larger cutoff values. The gaps are also similarly larger when ReDDE is used for collection selection and SSL is used for result merging, for the reasons given above.

However, the results on the Qprobed testbed, shown in Table 3, are rather different. The CORI-CORI combination produces the greatest effectiveness when duplicate documents are removed. Comparing the results of ReDDE-CORI and ReDDE-SSL combinations suggests that the performance of SSL

[5]The list of near-duplicate documents detected by DECO is available at: `http://www.cs.rmit.edu.au/~ybernste`

Table 2: *The impact of removing duplicate and near-duplicate documents. Results are obtained by running the TREC topics 551–600 (title) on the "skewed" testbed. CO represents the cutoff value.*

	P@5	P@10	P@15	P@20
Duplicates Removed (CORI-CORI)				
CO3	0.0880	0.0660	0.0573	0.0520
CO5	0.0800	0.0680	0.0613	0.0530
CO10	0.1680†	0.1220*	0.1133*	0.0970†
CO20	0.1840†	0.1380‡	0.1253†	0.1140‡
Duplicates Irrelevant (CORI-CORI)				
CO3	0.0840	0.0640	0.0560	0.0510
CO5	0.0760	0.0660	0.0600	0.0520
CO10	0.1440	0.1120	0.1053	0.0900
CO20	0.1520	0.1200	0.1133	0.0990
Duplicates Removed (ReDDE-CORI)				
CO3	0.1625	0.1312	0.1139*	0.1052
CO5	0.1625	0.1417	0.1208*	0.1125
CO10	0.1667	0.1437	0.1250*	0.1177*
CO20	0.1583*	0.1417*	0.1278*	0.1208†
Duplicates Irrelevant (ReDDE-CORI)				
CO3	0.1583	0.1271	0.1097	0.1010
CO5	0.1583	0.1375	0.1167	0.1083
CO10	0.1583	0.1354	0.1194	0.1125
CO20	0.1542	0.1333	0.1208	0.1135
Duplicates Removed (ReDDE-SSL)				
CO3	0.1625	0.1417†	0.1097*	0.1000*
CO5	0.1708*	0.1458*	0.1278†	0.1115†
CO10	0.1917*	0.1583†	0.1444†	0.1260†
CO20	0.1833†	0.1646†	0.1556‡	0.1437‡
Duplicates Irrelevant (ReDDE-SSL)				
CO3	0.1458	0.1208	0.1014	0.0906
CO5	0.1542	0.1354	0.1111	0.1010
CO10	0.1750	0.1312	0.1250	0.1104
CO20	0.1583	0.1417	0.1306	0.1135

Table 3: *The impact of removing duplicate and near-duplicate documents. Results are obtained by running the TREC topics 551–600 (title) on the "Qprobed" testbed. CO represents the cutoff value.*

	P@5	P@10	P@15	P@20
Duplicates Removed (CORI-CORI)				
CO3	0.1959	0.1776†	0.1578†	0.1378*
CO5	0.2163	0.1857†	0.1592†	0.1439†
CO10	0.2204†	0.1898†	0.1660‡	0.1469†
CO20	0.2571†	0.2143‡	0.1782‡	0.1612‡
Duplicates Irrelevant (CORI-CORI)				
CO3	0.1878	0.1612	0.1442	0.1296
CO5	0.2000	0.1653	0.1401	0.1235
CO10	0.1918	0.1633	0.1333	0.1204
CO20	0.2204	0.1776	0.1429	0.1255
Duplicates Removed (ReDDE-CORI)				
CO3	0.1750	0.1500	0.1194	0.1000*
CO5	0.2000	0.1729	0.1403*	0.1167*
CO10	0.2125	0.1938*	0.1597†	0.1406†
CO20	0.2083	0.1958	0.1681†	0.1469†
Duplicates Irrelevant (ReDDE-CORI)				
CO3	0.1667	0.1396	0.1139	0.0938
CO5	0.1875	0.1583	0.1292	0.1094
CO10	0.2000	0.1771	0.1389	0.1271
CO20	0.1958	0.1812	0.1444	0.1302
Duplicates Removed (ReDDE-SSL)				
CO3	0.1625	0.1396	0.1083	0.0958
CO5	0.2125*	0.1542†	0.1319†	0.1146*
CO10	0.2458†	0.1958†	0.1611†	0.1427‡
CO20	0.2292†	0.2021‡	0.1736‡	0.1542‡
Duplicates Irrelevant (ReDDE-SSL)				
CO3	0.1625	0.1313	0.1028	0.0885
CO5	0.1917	0.1333	0.1153	0.1042
CO10	0.2042	0.1687	0.1403	0.1167
CO20	0.1750	0.1500	0.1319	0.1156

and CORI merging methods is similar. Therefore, the high effectiveness of CORI-CORI is largely due to its effective collection selection method. We observed similar trends for CORI and ReDDE on a similar testbed (trec4-kmeans (Xu & Callan 1998)) in our preliminary experiments. We believe that ReDDE is not as effective as CORI on testbeds where the distribution of collection sizes in not skewed and the documents within each collection have similar topicality.

These results illustrate the importance of using diverse testbeds in such experiments; the uniform and skewed results are not predictive of the results on a collection created with another method.

As in experiments on the other testbeds, removal of duplicates may drastically improve the final precision. The difference can be significant at the 0.999 confidence interval for larger cutoff values.

The number of duplicate and near-duplicate documents detected by GHVs for different combinations is presented in Table 4. In the uniform and skewed testbeds, the number of near-duplicates detected climbs significantly for CORI-CORI while it remains near constant for the other combinations. In the Qprobed testbed, the number of near-duplicates in the result declines by selecting more collections, which is possibly an artefact of the way the testbed was constructed.

Comparing the exact-duplicate (ED) numbers for ReDDE-CORI and ReDDE-SSL suggests that SSL is more likely to return exact-duplicates in the final results than CORI. Using CORI and ReDDE collection selection algorithms leads to similar number of exact-duplicates on the uniform and skewed testbeds. However, collections selected by CORI return more exact-duplicates on the Qprobed testbed.

The coverage values in Table 4 represent the fraction of unique documents in the testbeds that are being searched on different cutoff points. The coverage values grow linearly for all combinations in the uniform and Qprobed testbeds. This implies that collections selected by CORI and ReDDE contain similar number of documents.

In the skewed testbed, the coverage values for ReDDE collection selection increases very quickly for small cutoff values while it grows linearly for CORI. This is due to the fact that the three largest collections in the skewed testbeds get high ranks by ReDDE for the majority of queries. Therefore, they are very likely to be selected by the broker in the top three or five collections. However in CORI, the three largest collections do not have any advantage over the other collections to be selected.

8 Conclusions

We have investigated the problem of distributed information retrieval on collections that overlap, evaluating the effectiveness of several collection selection and result merging algorithms. Our experiments are broadly consistent with previous observations on the traditional, disjoint DIR testbeds: ReDDE is a better collection selection algorithm than CORI when the distribution of collection sizes is skewed, and SSL is a more effective result merging method than CORI. However, on a testbed where documents in each col-

Table 4: *Number of duplicates, near-duplicates and coverage values obtained by collection selection and result merging combinations on the three testbeds. For all experiments, the TREC topics 551–600 (title) have been used and the numbers are averaged over all queries. CO is the cutoff value. For each query, 1 000 answers are collected. ND and ED stand for near-duplicate and exact-duplicate respectively.*

	ND	**ED**	**coverage**	**ND**	**ED**	**coverage**	**ND**	**ED**	**coverage**
	CORI-CORI			**ReDDE-CORI**			**ReDDE-SSL**		
Uniform									
CO3	8	53	0.06	9	19	0.06	10	53	0.06
CO5	10	78	0.11	9	66	0.11	11	77	0.11
CO10	11	116	0.21	8	110	0.21	9	118	0.21
CO20	15	191	0.38	11	174	0.38	11	195	0.38
Skewed									
CO3	9	56	0.06	11	106	0.31	10	143	0.31
CO5	11	82	0.11	9	140	0.35	11	193	0.35
CO10	13	122	0.23	12	163	0.42	10	229	0.42
CO20	15	211	0.42	9	211	0.54	11	298	0.54
Qprobed									
CO3	15	136	0.06	8	76	0.04	8	83	0.04
CO5	9	218	0.11	6	145	0.07	6	158	0.07
CO10	5	340	0.21	5	241	0.14	6	264	0.14
CO20	5	451	0.38	6	335	0.27	4	371	0.27

lection may share topicality, CORI seems to be a better option than ReDDE for collection selection.

For this work we introduced three testbeds created from documents available in the TREC GOV corpus. The different testbeds yield results that are somewhat inconsistent, demonstrating that design of experiment is critical to achieving robust results in this area—conclusions based solely on one testbed might not generalise.

We have shown that removing duplicates and near-duplicates can significantly improve the final search effectiveness. We used grainy hash vectors to detect duplicate and near-duplicate documents in the final list of results on the broker. Grainy hash vectors successfully identified 72% of all near-duplicate documents in the results. The accuracy of GHVs is consistent with the values we previously reported on a centralized index (Bernstein et al. 2006). These results demonstrate that duplicate removal need not be expensive, and can greatly enhance the quality of results returned by a search engine.

References

Allan et al, J. (2003), 'Challenges in information retrieval and language modeling: report of aworkshop held at the center for intelligent information retrieval, University of Massachusetts Amherst, september 2002', *SIGIR Forum* **37**(1), 31–47.

Avrahami, T., Yau, L., Si, L. & Callan, J. (2006), 'The FedLemur: federated search in the real world', *Journal of the American Society for Information Science and Technology* **57**(3), 347–358.

Bergman, M. (2001), 'The deep Web: Surfacing hidden value', *Journal of Electronic Publishing* **7**(1).

Bernstein, Y., Shokouhi, M. & Zobel, J. (2006), Compact features for detection of near-duplicates in distributed retrieval, *in* 'Proceedings of String Processing and Information Retrieval Symposium (to appear)', Glasgow, Schotland.

Bernstein, Y. & Zobel, J. (2004), A scalable system for identifying co-derivative documents, *in* 'Proceedings of String Processing and Information Retrieval Symposium', Padova, Italy, pp. 55–67.

Bernstein, Y. & Zobel, J. (2005), Redundant documents and search effectiveness, *in* 'Proceedings of 14th ACM CIKM Conference on Information and Knowledge Management', Bremen, Germany, pp. 736–743.

Brin, S., Davis, J. & García-Molina, H. (1995), Copy detection mechanisms for digital documents, *in* 'Proceedings of ACM SIGMOD international conference on Management of Data', San Jose, California, pp. 398–409.

Broder, A. Z., Glassman, S. C., Manasse, M. S. & Zweig, G. (1997), 'Syntactic clustering of the web', *Computer Networks and ISDN Systems* **29**(8-13), 1157–1166.

Callan, J. & Connell, M. (2001), 'Query-based sampling of text databases', *ACM Transactions on Information Systems* **19**(2), 97–130.

Callan, J., Connell, M. & Du, A. (1999), Automatic discovery of language models for text databases, *in* 'Proceedings of ACM SIGMOD International Conference on Management of Data', Philadelphia, Pennsylvania, pp. 479–490.

Callan, J., Croft, W. B. & Harding, S. M. (1992), The INQUERY retrieval system, *in* 'Proceedings of third International Conference on Database and Expert Systems Applications', Valencia, Spain, pp. 78–83.

Callan, J., Lu, Z. & Croft, W. B. (1995), Searching distributed collections with inference networks, *in* 'Proceedings of 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Seattle, Washington, pp. 21–28.

Chowdhury, A., Frieder, O., Grossman, D. & McCabe, M. C. (2002), 'Collection statistics for fast duplicate document detection', *ACM Transactions on Information Systems* **20**(2), 171–191.

Conrad, J. G., Guo, X. S. & Schriber, C. P. (2003), Online duplicate document detection: Signature reliability in a dynamic retrieval environment, *in* 'Proceedings of 12th ACM CIKM Conference on Information and Knowledge Management', New Orleans, Louisiana, pp. 443–452.

Cooper, J. W., Coden, A. R. & Brown, E. W. (2002), Detecting similar documents using salient terms, *in* 'Proceedings of 11th ACM CIKM Conference on Information and Knowledge Management', McLean, Virginia, pp. 245–251.

Craswell, N., Bailey, P. & Hawking, D. (2000), Server selection on the World Wide Web, *in* 'Proceedings of Fifth ACM Conference on Digital Libraries', San Antonio, Texas, pp. 37–46.

Craswell, N. & Hawking, D. (2002), Overview of the TREC-2002 Web Track, *in* 'Proceedings of TREC-2002', Gaithersburg, Maryland.

Croft, B. (2000), 'Combining approaches to information retrieval', *Advances in information retrieval, chapter 1* pp. 1–36.

D'Souza, D., Zobel, J. & Thom, J. (2004), Is CORI effective for collection selection? an exploration of parameters, queries, and data, *in* 'Proceedings of Australian Document Computing Symposium', Melbourne, Australia, pp. 41–46.

Fetterly, D., Manasse, M. & Najork, M. (2003), On the evolution of clusters of near-duplicate web pages, *in* 'Proceedings of first Latin American Web Congress', IEEE, pp. 37–45.

Fox, E. & Shaw, J. (1994), Combination of multiple searches, *in* 'Proceedings of TREC-1994', NIST Special Publication, Gaithersburg, Maryland, pp. 105–108.

Gauch, S., Wang, G. & Gomez, M. (1996), 'ProFusion: Intelligent fusion from multiple, distributed search engines', *Journal Universal Computer Science* **2**(9), 637–649.

Gravano, L., Chang, C. K., Garcia-Molina, H. & Paepcke, A. (1997), STARTS: Stanford proposal for Internet metasearching, *in* 'Proceedings of ACM SIGMOD International Conference on Management of Data', Tucson, Arizona, pp. 207–218.

Gravano, L., Garcia-Molina, H. & Tomasic, A. (1999), 'GlOSS: text-source discovery over the Internet', *ACM Transactions on Database Systems* **24**(2), 229–264.

Hawking, D. & Thomas, P. (2005), Server selection methods in hybrid portal search, *in* 'Proceedings of 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Salvador, Brazil, pp. 75–82.

Hernandez, T. & Kambhampati, S. (2005), Improving text collection selection with coverage and overlap statistics, *in* 'Proceedings of 14th International Conference on World Wide Web', Chiba, Japan.

Hoad, T. C. & Zobel, J. (2003), 'Methods for identifying versioned and plagiarised documents', *Journal of the American Society for Information Science and Technology* **54**(3), 203–215.

Ilyinski, S., Kuzmin, M., Melkov, A. & Segalovich, I. (2002), An efficient method to detect duplicates of web documents with the use of inverted index, *in* 'Proceedings of 11th International Conference on World Wide Web', Honolulu, Hawaii.

Ipeirotis, P. G. & Gravano, L. (2002), Distributed search over the hidden Web: Hierarchical database sampling and selection., *in* 'Proceedings of 28th International Conference on Very Large Data Bases', Hong Kong, China, pp. 394–405.

Ipeirotis, P. G. & Gravano, L. (2004), When one sample is not enough: improving text database selection using shrinkage, *in* 'Proceedings of ACM SIGMOD International Conference on Management of Data', Paris, France, pp. 767–778.

Kirsch, T. (2003), 'Document retrieval over networks wherein ranking and relevance scores are computed at the client for multiple database documents', *U.S. Patent 5,659,732* .

Kolcz, A., Chowdhury, A. & Alspector, J. (2004), Improved robustness of signature-based near-replica detection via lexicon randomization, *in* 'Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', Seattle, WA, pp. 605–610.

Lee, J. (1997), Analyses of multiple evidence combination, *in* 'Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval', Philadelphia, Pennsylvania, United States, pp. 267–276.

Lyon, C., Malcolm, J. & Dickerson, B. (2001), Detecting short passages of similar text in large document collections, *in* 'Proceedings of Conference on Empirical Methods in Natural Language Processing', Philadelphia, Pennsylvania.

Manber, U. (1994), Finding similar files in a large file system, *in* 'Proceedings of USENIX Winter Technical Conference', San Fransisco, CA, pp. 1–10.

Meng, W., Yu, C. & Liu, K. (2002), 'Building efficient and effective metasearch engines', *ACM Computing Surveys* **34**(1), 48–89.

Nottelmann, H. & Fuhr, N. (2003), Evaluating different methods of estimating retrieval quality for resource selection, *in* 'Proceedings of 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Toronto, Canada, pp. 290–297.

Powell, A. L. & French, J. (2003), 'Comparing the performance of collection selection algorithms', *ACM Transactions on Information Systems* **21**(4), 412–456.

Pugh, W. & Henzinger, M. H. (2003), 'Detecting duplicate and near-duplicate files (United States Patent 6,658,423)'.

Rasolofo, Y., Abbaci, F. & Savoy, J. (2001), Approaches to collection selection and results merging for distributed information retrieval, *in* 'Proceedings of 10th ACM CIKM International Conference on Information and knowledge management', Atlanta, Georgia, pp. 191–198.

Selberg, E. & Etzioni, O. (1997), 'The MetaCrawler architecture for resource aggregation on the web', *IEEE Expert* **12**(1), 8–14.

Shokouhi, M., Scholer, F. & Zobel, J. (2006), Sample sizes for query probing in uncooperative distributed information retrieval, *in* 'Proceedings of of Eighth Asia Pacific Web Conference', Harbin, China, pp. 63–75.

Si, L. & Callan, J. (2003*a*), Relevant document distribution estimation method for resource selection, *in* 'Proceedings of 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Toronto, Canada, pp. 298–305.

Si, L. & Callan, J. (2003*b*), 'A semisupervised learning method to merge search engine results', *ACM Transactions on Information Systems* **21**(4), 457–491.

Si, L. & Callan, J. (2004), Unified utility maximization framework for resource selection, *in* 'Proceedings of 13th ACM CIKM Conference on Information and Knowledge Management', Washington, D.C., pp. 32–41.

Si, L. & Callan, J. (2005), Modeling search engine effectiveness for federated search, *in* 'Proceedings of 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', Salvador, Brazil.

Si, L., Jin, R., Callan, J. & Ogilvie, P. (2002), A language modeling framework for resource selection and results merging, *in* 'Proceedings of 11th ACM CIKM International Conference on Information and Knowledge Management', New York, NY, pp. 391–397.

Sugiura, A. & Etzioni, O. (2000), Query routing for web search engines: architectures and experiments, *in* 'Proceedings of the 9th international World Wide Web conference on Computer networks', North-Holland Publishing Co., Amsterdam, The Netherlands, pp. 417–429.

Wu, S. & McClean, S. (2006), 'Result merging methods in distributed information retrieval with overlapping databases', *Journal of Information Retrieval (In press)* .

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

Yuwono, B. & Lee, D. L. (1997), Server ranking for distributed text retrieval systems on the internet, *in* 'Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)', World Scientific Press, Melbourne, Australia, pp. 41–50.

Zamir, O. & Etzioni, O. (1999), Grouper: a dynamic clustering interface to web search results, *in* 'Proceedings of 8th International Conference on World Wide Web', Toronto, Canada, pp. 1361–1374.

Zobel, J. (1997), Collection selection via lexicon inspection, *in* P. Bruza, ed., 'Proceedings of the Australian Document Computing Symposium', pp. 74–80.

Zobel, J. & Bernstein, Y. (2006), The case of the duplicate documents: Measurement, search, and science, *in* 'Proceedings of of Eighth Asia Pacific Web Conference', Harbin, China, pp. 26–39.