Collection Statistics for Fast Duplicate Document Detection

ABDUR CHOWDHURY, OPHIR FRIEDER, DAVID GROSSMAN,
and MARY CATHERINE McCABE
Illinois Institute of Technology

We present a new algorithm for duplicate document detection that uses collection statistics. We compare our approach with the state-of-the-art approach using multiple collections. These collections include a 30 MB 18,577 web document collection developed by Excite@Home and three NIST collections. The first NIST collection consists of 100 MB 18,232 LA-Times documents, which is roughly similar in the number of documents to the Excite@Home collection. The other two collections are both 2 GB and are the 247,491-web document collection and the TREC disks 4 and 5—528,023 document collection. We show that our approach called I-Match, scales in terms of the number of documents and works well for documents of all sizes. We compared our solution to the state of the art and found that in addition to improved accuracy of detection, our approach executed in roughly one-fifth the time.

1. INTRODUCTION

Data portals are everywhere. The tremendous growth of the Internet has spurred the existence of data portals for nearly every topic. Some of these portals are of general interest; some are highly domain specific. Independent of the focus, the vast majority of the portals obtain data, loosely called documents, from multiple sources. Obtaining data from multiple input sources typically results in duplication. The detection of duplicate documents within a collection has recently become an area of great interest [Shivakumar and Garcia-Molina 1998; Broder et al. 1997] and is the focus of our described effort.

Typically, inverted indexes are used to support efficient query processing in information search and retrieval engines. Storing duplicate documents affects both the accuracy and efficiency of the search engine. Retrieving duplicate documents in response to a user's query clearly lowers the number of valid responses provided to the user, hence lowering the accuracy of the user's response set. Furthermore, processing duplicates necessitates additional computation

This work was partially supported by the National Science Foundation under the National Young Investigator program.
Author's address: Information Retrieval Laboratory, Illinois Institute of Technology, 10 West 31st Street, Chicago, IL 60616; email: abdur@ir.iit.edu; ophir@cs.iit.edu; grossman@iit.edu; mcatherm@comcast.net.

without introducing any additional benefit. Hence, the processing efficiency of the user's query is lowered.

A problem introduced by the indexing of duplicate documents is potentially skewed collection statistics. Collection statistics are often used as part of the similarity computation of a query to a document. Hence, the biasing of collection statistics may affect the overall precision of the entire system. Simply put, not only is a given user's performance compromised by the existence of duplicates, but also the overall retrieval accuracy of the engine is jeopardized.

The definition of what constitutes a *duplicate* is unclear. For instance, a duplicate can be defined as the exact syntactic terms, without formatting differences. Throughout our efforts however, we adhere to the definition previously referred to as a measure of *resemblance* [Broder et al. 1997; Heintze 1996]. The general notion is that if a document contains roughly the same semantic content it is a duplicate whether or not it is a precise syntactic match. When searching web documents, one might think that, at least, matching URL's would identify exact matches. However, many web sites use dynamic presentation wherein the content changes depending on the region or other variables. In addition, data providers often create several names for one site in an attempt to attract users with different interests or perspectives. For instance, www.fox4.com, onsale.channel9.com, and www.realtv.com all point to an advertisement for real TV.

While the previous examples are for web documents, the same holds true for other collections where multiple document sources populate a single document collection. The National Center for Complimentary and Alternative Medicine (NCCAM), part of the National Institutes of Health [NIH 2000] supports a search engine for medical data whose inputs come from multiple medical data sources. Given the nature of the data, duplicates are common. Since unique document identifiers are not possible across the different sources, the detection of duplicate information is essential in producing non-redundant results.

A previously proposed solution is the digital syntactic clustering (DSC) algorithm and its super shingle (DSC-SS) variant [Broder et al. 1997]. While these algorithms are commonly used, they have efficiency problems. One reported run took ten CPU days to process a thirty million-document collection [Broder et al. 1997]. Additionally, DSC-SS and DSC are known to perform poorly on small documents. Given that the average size of a document on the web is around 4 KB [Giles and Lawrence 1999; Lawrence and Giles 1998], working with small documents is imperative.

Our algorithm, called IIT-Match or I-Match for short, filters documents based on term collection statistics. Our results show that I-Match is five to six times faster than the DSC-SS algorithm. Furthermore, we show that I-Match does not ignore small documents and places each document into at most one duplicate set. Hence, I-Match increases accuracy and usability. Other approaches place potentially duplicate documents in multiple clusters. Hence, it is harder for a user to detect the actual duplicates. Finally, the sets of duplicates we detect are usually 'tighter' than DSC because we require an 'exact match' for the terms

remaining after our filtration process. However, like other approaches, we still identify non-exact duplicates.

2. PRIOR WORK

We partition prior work into three categories: shingling techniques, similarity measure calculations, and document images. Shingling techniques, such as COPS [Brin et al. 1995], KOALA [Heintze 1996], and DSC [Broder et al. 1997], take a set of contiguous terms or *shingles* of documents and compare the number of matching shingles. The comparison of document subsets allows the algorithms to calculate a percentage of overlap between two documents. This type of approach relies on hash values for each document subsection and filters those hash values to reduce the number of comparisons the algorithm must perform. The filtration, therefore, improves the runtime performance. Note that the simplest filter is strictly a syntactic filter based on simple syntactic rules, and the trivial subset is the entire collection. We illustrate later why such a naive approach is not generally acceptable. In the shingling approaches, subdocuments rather than whole documents, are compared, thus each document may produce many potential duplicates. Returning many potential matches requires vast user involvement to sort out potential duplicates, diluting the usefulness of the approach.

To combat the inherent efficiency issues, several optimization techniques were proposed to reduce the number of comparisons made. One approach was to retain only a portion of the shingles. That is, the approach either removed frequently occurring shingles [Heintze 1996] or retained only every 25th shingle [Broder et al. 1997]. This reduction, however, hinders the accuracy. Since no semantic premise is used to reduce the volume of data, a random degree of 'fuzziness' is introduced to the matching process resulting in relatively non-similar documents being identified as potential duplicates. Even with the performance-improving technique of removing shingles occurring in over 1000 documents and keeping only every 25th shingle, the implementation of the DSC algorithm took 10 CPU days to process 30 million documents [Broder et al. 1997].

The DSC algorithm has a more efficient alternative, DSC-SS, which uses super shingles. This algorithm takes several shingles and combines them into a super shingle. This results in a document with a few super shingles rather than many shingles. Instead of measuring resemblance as a ratio of matching shingles, resemblance is defined as matching *a single super shingle* in two documents. This is much more efficient because it no longer requires a full counting of all overlaps. The authors, however, noted that DSC-SS does "not work well for short documents" so no runtime results are reported [Broder et al. 1997].

Approaches that compute document-to-document similarity measures [Buckley et al. 1999; Sanderson 1997] are similar to document clustering work [Salton et al. 1975] in that they use similarity computations to group potentially duplicate documents. All pairs of documents are compared, that is, each document is compared to every other document and a similarity weight is

calculated. A document to document similarity comparison approach is thus computationally prohibitive given the theoretical $O(d^2)$ runtime, where d is the number of documents. In reality, these approaches only evaluate documents with an overlap of terms. Thus, the actual runtime is data dependent and difficult to predict accurately.

To further examine this class of duplicate or clustering approaches, we examine the mechanics of their term selection and weighting to cluster or compare documents for similarity. Typically, most of these document processing systems use an inverted index to efficiently retrieve documents containing a given index term. Various techniques exist that select the terms that are retrieved from the inverted index, how these terms are analyzed and manipulated, and how a similarity measure is computed [Salton et al. 1975; Singhal et al. 1996; Robertson et al. 1999; Kwok 1996]. Independent of the particular techniques chosen, the final computed weight is then used to sort the retrieved documents.

The basic hypothesis of similarity measure similar document detection approaches is that two documents are similar if the similarity measure of one document to the other is high. Therefore, for each document, a similarity score is obtained between the document and all other documents in the collection. This means that the entire posting list for each term in each document must be retrieved. This approach of using the document as a query, thus clustering on those results sets, is computationally infeasible for large collections or dynamic collections since each document must be queried against the entire collection. Sanderson used a variation on this where the terms are selected via a relevance feedback algorithm [Sanderson 1997; Rocchio 1971], which used IDF (inverse document frequency) to weight which terms would be used for the original query/document. Each term queried against the inverted index must retrieve all the documents for the posting list to be analyzed. For large collections, where a common term may occur in millions of records, this is computationally expensive. For term selection approaches the cost is significantly less, but still requires at least the same number of I/O operations as the number of terms selected via the relevance feedback algorithm. Our approach is based on eliminating these I/O costs and still finding duplicate documents as efficiently as possible.

Finally, research for image duplicate detection is addressed in [Kjell et al. 1994; Scotti and Lilly 1999]. While these are approaches to find duplicate data, the techniques and issues are image processing, rather than document processing and thus are not examined or addressed in this paper.

3. ALGORITHM

Our motivation is to provide a duplicate detection algorithm that can scale to the size of the web and handle the short documents typically seen there. Furthermore, we seek to place each document in only one set of potential duplicates. The degree of similarity supported should be sufficiently loose to identify non-exact matches but tight enough to assure that true duplicates are detected. Last, the approaches and algorithms discussed here are addressed to finding

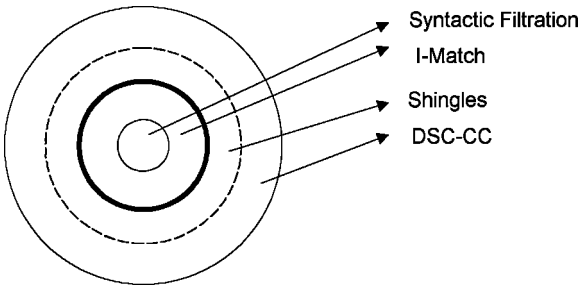

Fig. 1. Restrictiveness of techniques.

duplicate documents not plagiarism or similar problems; thus, the similarity threshold is considerably higher.

In Figure 1, we illustrate the relative restrictiveness of different algorithms. DSC-SS is the loosest approach because it only requires one super shingle to match. Shingling is tighter because a percentage overlap in the remaining shingles is required. However, shingles and DSC-SS are very sensitive to adjustments in shingle size and thresholds. We drew a dotted line to indicate that these may be adjusted in such a way that shingling would be the less restrictive. Syntactic filters are the most restrictive because they leave most of the terms in the document representation. Thus, documents must be very close to an exact match to resemble. The I-Match approach strikes a balance between parsing and the previously described existing techniques.

I-Match does not rely on strict parsing, but instead, uses collection statistics to identify which terms should be used as the basis for comparison. An inverse document frequency weight is determined for each term in the collection. The *idf* for each term is defined by $t_x = log\ (N/n)$, where N is the number of documents in the collection and n is the number of documents containing the given term. The use of *idf* collection statistics allows us to determine the usefulness of terms for duplicate document detection. It was previously shown that terms with high collection frequencies often do not add to the semantic content of the document [Grossman et al. 1993; Smeaton et al. 1997]. Our approach hinges on the premise that removal of very infrequent terms or very common terms results in good document representations for identifying duplicates.

We input a document, filter the terms based on collection statistics (and other simple parsing techniques) and compute a single hash value for the document. All documents resulting in the same hash value are duplicates. We use the SHA1 algorithm [NIST 1995] for our hash, using the ordered terms in the document as input and getting <docid, hashvalue> tuples as output. The ordering of terms is critical to detect similar documents that have reordered the paragraphs. The SHA1 hash algorithm is used because it is designed to be very fast and is good for messages of any length. It is designed for text processing and is known for its even distribution of hash values.

SHA1 produces a 20-byte or 160-bit hash value. By using a secure digest algorithm, we reduce the probability of two different token streams creating the

same hash value to $P(2^{-160})$. We insert each <docid, hashvalue> tuple into a tree requiring processing time on the order of $(O(d \log d))$. Other efficient storage and retrieval data structures such as a hash table could be used as alternatives, which would give a complexity of $(O(d))$. The identification of duplicates is handled through the inserts into the tree or hash table. Any collisions of hash values represent duplicates and the document identifiers are stored in that node of the tree or hash bucket. A scan through the tree or hash table produces a list of all clusters of duplicates, where a node contains more than one document. Pseudocode for the algorithm is as follows.

1. Get document.
2. Parse document into a token stream, removing format tags.
3. Using term thresholds (idf), retain only significant tokens.
4. Insert relevant tokens into unicode ascending ordered tree of unique tokens.
5. Loop through token tree and add each unique token to the SHA1 [NIST 1995] digest. Upon completion of token tree loop, a (doc_id, SHA1 Digest) tuple is defined.
6. The tuple (doc_id, SHA1 Digest) is inserted into the storage data structure based on SHA1 Digest key.
7. If there is a collision of digest values then the documents are similar.

The overall runtime of our approach is $(O(d \log d))$ in the worst case where all documents are duplicates of each other and $(O(d))$ otherwise, where d is the number of documents in the collection. All similar documents must be grouped together. That is, the corresponding document identifiers must be stored as a group. In the most extreme case, all of the hash values are the same (all the documents are similar to each other). In such a case, to store all the document identifiers together in a data structure (tree) requires $(O(d \log d))$ time. Typically, however, the processing time of our approach is $O(d)$ time.

The calculation of *idf* values can be approached with either of two methods. The first is with the use of a training collection to produce a set of terms *idf* tuples before the deduplication work occurs. Since term *idf* weights change slightly as collection sizes grow, this is an acceptable solution [Frieder et al. 2000]. A second approach is to run two passes over the documents, where the first pass calculates the *idf* weights of the terms, and the second pass finds duplicates with the I-Match algorithm. This approach would increase the actual run time of the algorithm, but the theoretical complexity would remain unchanged.

Our time complexity is comparable to the DSC-SS algorithm, which generates a single super shingle if the super shingle size is large enough to encompass the whole document. Otherwise, it generates k super shingles while we generate only one. Since k is a constant in the DSC-SS timing complexity, the two algorithms are theoretically equivalent. I-Match, however, is more efficient in practice.

The real benefit of I-Match over DSC-SS, however, is not the timing improvement but the fact that small sized documents are not ignored. With DSC-SS, it is quite likely that for sufficiently small documents, no shingles are

Fig. 2. I-Match-Doc. Document thresholds based on NIDF values.

identified for duplicate detection. Hence, those short documents are not considered even though they may be duplicated. Given the wide variety of domains for which duplicate document detection may be used, for example, document declassification, email traffic processing, and so on, neglecting short documents is a potentially serious issue.

4. I-MATCH RESULTS

We implemented the DSC, DSC-SS and I-Match algorithms in the IIT Advanced Information Retrieval Engine (AIRE) system [Chowdhury et al. 2000]. To test our algorithm, we implemented a variety of filtering techniques based on various thresholds. Figure 2 graphically illustrates several I-Match thresholding techniques. In Figure 2, the shaded regions are discarded term regions. The next section describes, in detail, the different thresholding techniques. We partition the description of our experimentation into the following sections: experimental layout, syntactic one-pass approaches, quality of duplicate sets found, handling of short documents, runtime performance, and effects on the average precision and recall.

4.1 Experimental Layout

We experimented with two filtration techniques based on collection statistics: I-Match-Doc and I-Match-IDF. I-Match-Doc filters the unique terms of a given document by idf value to reach a specified percentage of the original unique terms of the document. The non-filtered terms are used to create the hash value. For instance, 75% of the document might be reached by removing the 25% of the terms with the lowest idf values (most frequent terms). Another example retains 50% of the original unique tokens of the document by removing 25% of the terms with the lowest idf and 25% of the terms with the highest idf (least frequent). Thus, a percentage in terms of the number of unique terms of the original document will always remain, except for extremely small documents. That is, a document containing less than four unique tokens would be filtered if we wanted to keep less than 25% of the original document.

The I-Match-IDF filtration technique filters terms based on normalized IDF values. The term IDF values are normalized across the collection so that they fall within a 0 to 1 interval. For each document, an IDF cut-off is used, thus any term above or below a certain idf value is removed from the terms used to create the hash.

For each approach, we calculated the number of documents that were completely filtered, that is, were not evaluated due to the removal of all tokens. We calculated the average distinct terms before and after filtration and the

Table I. Experimental Collections

Collection Name	Collection Size	Number of Documents
Excite@Home Web	30 MB	18,577
NIST LA Times	100 MB	18,232
NIST Web	2 GB	247,491
NIST TREC disks 4 & 5	2 GB	528,023

average number of terms in each document pre and post filtration. We counted the number of duplicate clusters found with each approach. We evaluated each duplicate set found and counted how many of documents within the cluster matched on the evaluation technique and how many of those did the title or URL match. Therefore, if a document was found to have a duplicate and both documents had either an identical title or URL then it was counted as a duplicate-title, otherwise it was counted just as a duplicate. We evaluated the number of unique documents in our collection, so a document cluster was counted only once. Lastly, we noted the time to evaluate the collection. We tracked the following for each approach and each collection.

—*Number of documents filtered by the approach*
—*Pre/Post average number of unique terms per document*
—*Pre/Post average number of terms per document (document size)*
—*Number of document clusters found*
—*Number of duplicates found with the same URL/Title*
—*Number of duplicate documents found with just the duplicate approach*
—*Processing time*

We now describe the various thresholding values used. We ran experiments of the I-Match-Doc approach with thresholds of 10, 20, 30, 40, 50, 60, 70, 80, and 90% of the most common terms and the inverse of the least common terms, totaling 18 experiments. We ran the LOW and HIGH filters first, filtering the lowest *X percentage*, and the highest *X percentage*, based on *idf* value. Then we filtered the edges of the document—the most frequent and least frequent terms, keeping the middle ones, 20%, 40%, 60% and 80%. Finally, we filtered the middle of the document, keeping only the most and least frequent terms, inner 20%, 40%, 60%, and 80%, 8 more experiments.

The I-Match-IDF filters use cut-off thresholds to filter any word above and below certain normalized idf values. For the DSC-SS variant algorithm experiments, we collected document sizes both pre and post filtration, and the timing results. Document size information was used to see how sensitive these types of algorithms are to smaller documents. The DSC-SS runs used super shingle sizes of 2, 4, 5, 10, 15, 20, 25, 50, 75 and 100 shingles where each shingle was 10 terms. The DSC experiments used thresholds of 0.5, 0.6, 0.7, 0.8 and 0.9. Table X contains the notation description of the I-Match experiments.

We used four document collections, as shown in Table I. Each collection was chosen to test particular issues involved with duplicate detection. The first is an 18,577-web document collection flagged as duplicates by Excite@Home. The

Excite@Home document collection was produced from ten million web documents gathered through crawling the World Wide Web. These documents were then filtered by the Excite@Home engineers to include only those documents thought to be 'duplicate.' The collection contains 18,577 documents, each of which is suspected of having a duplicate web document within the collection. Many URLs are in the collection repeatedly because of multiple spider inputs. This collection is approximately 30 megabytes in size. The Excite@Home collection is highly duplicated. Thus, as better approaches are used, the greater is the percentage of the collection found as duplicate.

The second is an 18,232-document Los Angles Times collection. A subset of the entire LA Times collection provided by NIST, this subset was selected to roughly mirror the Excite@Home collection in terms of the number of documents, but to be comprised of significantly longer documents. We synthesized the collection and used the LA Times subset as a ground truth data collection. That is, we inserted known duplicate documents into the collection and analyzed the performance of the various approaches in finding the inserted duplicates.

The third and fourth collections are likewise from NIST and are the TREC web and ad-hoc collections. The NIST web collection is a subset of a backup of the web from 1997 that was used in the TREC web track for TREC 7 and 8. This collection was chosen as a representation of a larger standard web collection to show the scalability of the I-Match algorithm. The NIST Web collection is used to test the run time performance of DSC, DSC-SS and I-Match approaches.

The TREC disks 4–5 are chosen as a second document collection of 2 gigabytes to see what effects duplication has on the average precision and recall. Since this collection has standard query and judgment results, it is a good collection to see if duplication has an effect on the end result sets. The NIST TREC collection is used to test the effects of duplication on known relevance judgments.

Unfortunately, there is no available absolute body of truth or a benchmark to evaluate the success of these techniques. Thus, it is difficult to get any type of quantitative comparison of the different algorithms and thresholding techniques. This is not likely to change in the near future. As document collections grow, the likelihood of judgments of duplicates being made is small; therefore, the best that can be hoped for is to provide fast efficient techniques for duplication detection that can be passed on to analysis for further evaluation.

4.2 Syntactic Filtration

The most obvious way to identify duplicate documents is to directly hash the entire contents of a document to a unique value. This type of approach finds exact matches by comparing the calculated hash value with the other document hash values. A simple hash of the entire document is not resilient to small document changes, like an additional space added to a document, the addition or deletion of the word "the," a stem change to a term, or the replication of a sentence or paragraph. Because of these reasons, hash values are not commonly used

for duplicate document detection. However, they are used to see if a particular document has changed.

We experimented with various filtration techniques to improve the resilience of the direct hash approach to small document changes. If a simple filtration technique based on strictly syntactic information is successful then fast duplicate and similar document detection could be achieved. We had to evaluate this basic approach prior to considering the use of more sophisticated, collection dependent, hence computationally expensive, filtration techniques.

We experimented with five filtering techniques that removed all white spaces from a document, and created a list of unique tokens to hash.

—*sw – Stop word filtration*
—*tg5 – Terms less than 5 characters in length*
—*tl25 – Terms greater than 25 characters in length*
—*nosc – Terms with special characters*
—*stem – Stemming*

All permutations of the filtration techniques were investigated. We used the 571-stop-word list used by many participants of the Text Retrieval Conference and available on the SMART information retrieval site [SMART FTP 2000]. For word length filters, we removed all the words less than the average word length [Baeza-Yates and Ribeiro-Neto 1999], five, in the *length* > 5 (*tg5*) filter. To filter very long words, we arbitrarily selected 25 as the cutoff for the *length* > 25 (*tl25*) filter. For stemming, we used the Porter stemming algorithm [Porter 1980].

The effect of filtering tokens on the degree of duplicate document detection is shown in Table II. We used the Excite@Home collection because the collection is fully duplicated. Therefore, the percentage of duplicates found is an evaluation metric of the effectiveness of the filter. Also shown in the table, is the percentage of terms retained after each filtering technique. Generally speaking, as we show in Table II, the higher the filtration, the greater the degree of detection. While several of the filtration techniques do find 88% of the collection, the duplicates they find are near or exact matches and a maximum number of unique documents of 2038. In contrast, I-Match for this same collection detects 96.2% duplication and a maximum number of unique documents of 568. Clearly the lower the maximum number of unique documents, the better is the detection capability.

Our simple filtering techniques reduced the list of tokens used to create the hash. By eliminating white spaces and only keeping unique tokens, many small document changes are eliminated. Keeping only unique tokens eliminates movement of paragraph errors, stemming removes errors caused by small token changes, and stop word removal removes errors caused by adding or removing common irrelevant tokens, in terms of semantics. We found that removing tokens containing 'special characters' (i.e./, -, =, etc.) performed the best in terms of removing tokens from documents.

These syntactic filtration techniques are very fast, however, the degree of duplication they detect is limited. Such approaches detect only near or exact duplicates and do not find documents with small differences, like an updated

Table II. Syntactic Experiments

	Percentage of Original Lexicon	Percent Found as Duplicates	Unique Documents Found in Collection
nothing	100.0%	62.2%	7017
sw	99.9%	62.2%	7017
tg5	93.2%	62.4%	6966
sw,tg5	93.2%	62.4%	6966
tl25	60.1%	82.4%	3253
sw,tl25	60.1%	82.4%	3253
tg5,tl25	53.4%	82.7%	3199
sw,tg5,tl25	53.4%	82.7%	3199
nosc	9.5%	87.4%	2214
nosc,sw	9.4%	87.4%	2197
nosc,tg5	7.0%	88.0%	2048
nosc,tg5,sw	6.9%	88.0%	2043
nosc,tl25	9.5%	87.4%	2214
nosc,tl25,sw	9.4%	87.4%	2197
nosc,tl25,tg5	7.0%	88.0%	2048
nosc,tl25,tg5,sw	6.9%	88.0%	2043
stem	80.4%	62.2%	7014
stem,sw	80.4%	62.2%	7014
stem,tg5	78.2%	62.4%	6963
stem,sw,tg5	78.2%	62.4%	6963
stem,tl25	41.2%	82.4%	3248
stem,sw,tl25	41.2%	82.4%	3248
stem,tg5,tl25	39.0%	82.7%	3192
stem,sw,tg5,tl25	39.0%	82.7%	3192
stem,nosc	6.9%	87.4%	2211
stem,nosc,sw	6.9%	87.4%	2194
stem,nosc,tg5	5.2%	88.0%	2045
stem,nosc,tg5,sw	5.2%	88.0%	2039
stem,nosc,tl25	6.9%	87.4%	2211
stem,nosc,tl25,sw	6.9%	87.4%	2194
stem,nosc,tl25,tg5	5.2%	88.0%	2045
stem,nosc,tl25,tg5,sw	5.2%	88.0%	2038

date string or different URL. Therefore, simple filtration techniques such as these do not suffice, and efforts such as DSC, DSC-SS, and I-Match merit further investigation.

4.3 Duplicate Sets

For the task of "remove all duplicates from this collection," it is helpful to get a list of duplicate document sets so that one from each set can be retained and the rest removed. Imagine getting, instead, many different lists of duplicates, where one document may be in many lists. This is essentially what DSC and DSC-SS return. The DSC-SS algorithm creates a duplicate document set for each super shingle that exists in at least two documents. Thus, each document (if it matches more than one super shingle) may appear in multiple document lists. For example, given documents D1, D2, D3 and D4 with super shingles as follows: D1 contains super shingle A. D2 contains super shingle A and B.

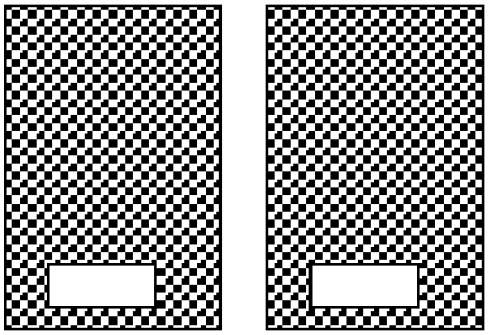

Fig. 3. Differing documents.

D3 and D4 contain super shingle B. The resulting sets of duplicates include {D1, D2} (from super shingle A), {D2, D3, D4} (from super shingle B). Now all of the clusters must be scanned to get a list of duplicates for D2. In contrast, I-Match places each document in one and only one duplicate document set.

Consider two documents that match all text except in one small portion as shown in Figure 3. Perhaps a name and an address for a regional contact are changed. It is likely that DSC-SS would identify these two documents as duplicates because the small section that differs may not be represented at all in the selected shingles, or a super shingle exists without a shingle from this section. I-Match will group these together as duplicates only if all terms in the differing section were filtered. This is quite likely with the name and address example because names are generally very infrequent across the collection, the numbers are removed in parsing, and state names are generally very common across the collection. On the other hand, if any word in the differing section is kept, the two documents are not matched.

To find the best performing I-Match approach, we contrived a set of duplicates to test the various approaches with a known test set of duplicate documents inserted into an existing collection. We computed the average document length for the test collection. We then chose ten documents from the collection, that were the average document length. These documents were used to create a test duplicate document collection. Each document is used to create 10 test duplicate documents. This is achieved by randomly removing every ith word from the document. In other words, for every ith word, pick a random number from one to ten. If the number is higher than the random threshold (call it alpha) then pick a number from 1 to 3. If the random number chosen is a 1 then remove the word. If the number is a 2 then flip it with a word at position $i + 1$. If it is a 3, add a word (randomly pick one from the term list). Last, these duplicate documents are inserted into the collection.

We then ran the I-Match thresholding techniques, DSC, and the DSC-SS with the creation of a super shingle for every 2 and 4 shingles on the LA Times subcollection, looking for the new test duplicate documents. We found two I-Match filtration techniques to be very effective, I-Match (Doc-L-90 and IDF-L-10). Doc-L-90 takes only terms with the highest IDF values, that is, very infrequent

Table III. Documents Found Ratio

Found Ratio					
Document	*DSC*	*DSC-SS-2*	*DSC-SS-4*	*DOC-L-90*	*IDF-L-10*
LA123190-0013	27.3%	0.0%	18.2%	36.4%	63.6%
LA123190-0022	54.5%	63.6%	18.2%	100.0%	100.0%
LA123190-0025	27.3%	0.0%	0.0%	90.9%	100.0%
LA123190-0037	18.2%	18.2%	0.0%	90.9%	100.0%
LA123190-0043	36.4%	0.0%	0.0%	90.9%	90.9%
LA123190-0053	18.2%	45.5%	45.5%	90.9%	100.0%
LA123190-0058	45.5%	18.2%	0.0%	90.9%	81.8%
LA123190-0073	54.5%	0.0%	0.0%	100.0%	100.0%
LA123190-0074	0.0%	0.0%	0.0%	90.9%	100.0%
LA123190-0080	27.3%	18.2%	0.0%	54.5%	63.6%
Average	30.9%	16.4%	8.2%	83.6%	90.0%

Table IV. Document Clusters Formed

Num Clusters					
Document	*DSC*	*DSC-SS-2*	*DSC-SS-4*	*DOC-L-90*	*IDF-L-10*
LA123190-0013	9	11	9	9	7
LA123190-0022	6	7	9	3	2
LA123190-0025	9	11	11	4	3
LA123190-0037	10	10	11	4	1
LA123190-0043	8	11	11	2	2
LA123190-0053	10	9	9	3	2
LA123190-0058	7	10	11	3	3
LA123190-0073	6	11	11	3	3
LA123190-0074	11	11	11	2	1
LA123190-0080	9	10	11	8	9
Average	8.5	10.1	10.4	4.1	3.3

terms, and only the 10% most infrequent terms are used for each document. The second approach (IDF-L-10) uses only the terms with normalized idf values of 0.1 or greater, thus very frequent terms in the collection are removed. In the following tables, we present the data obtained in our evaluation of the different approaches.

As shown, both I-Match approaches yield a significantly higher percentage of detection than either DSC or either of the DSC super shingle approaches. Furthermore, as expected, the super shingle approaches declined in the percentage detected, as the super shingle size increased. The DSC performance was better than both super shingle approaches.

The most effective I-Match techniques retain the highest idf valued terms from a document either as a percentage or as a normalized value. We produced 10 duplicate documents for 10 test documents, thus creating 11 known duplicate documents for each cluster. In Table III, we show the percentage of the total document duplication found for each approach. Both I-Match approaches find a greater duplication percentage for all test cases.

In Table IV, we illustrate that the I-Match techniques yield a smaller number of document clusters than any of the shingling techniques. We know, by design,

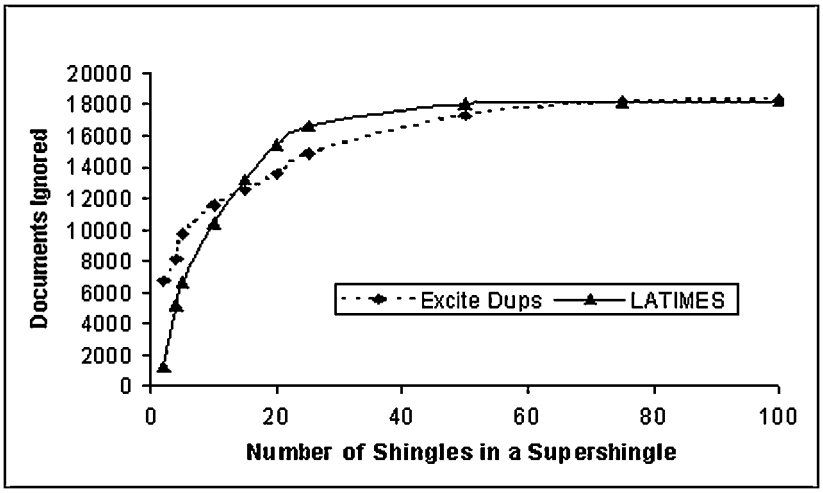

Fig. 4. Super shingle size vs. documents dropped.

that for each document the actual number of clusters to be formed should ideally be 1 since besides the original document, the other ten copies are simply slight modifications of the original. Therefore, a perfect similar document detection algorithm would generate one cluster per document. As shown, the I-Match configurations result in an average number of clusters per document of approximately 3 to 4. DSC and the super singling variants are significantly worse, ranging from 8 to 10 clusters.

Last, false positives were examined. The result sets from all runs were analyzed for false positives; a false positive was flagged if the duplicate detection algorithm clustered a different document other than the known duplicates. No false positives were detected with the I-Match approaches while the DSC reported two false positives and DSC-SS runs, one false positive. While this is not a high percentage, the reduced effectiveness of the DSC approach for clustering duplicates, and a higher rate of false positives may be issues for specific applications.

4.4 Short Documents

While DSC-SS is more efficient than DSC, it has known deficiencies with short documents. To evaluate how often DSC-SS completely ignores a document, we ran the algorithm against the Excite@Home duplicate document collection and the NIST LA Times collection. As presented in Figure 4, for the Excite@Home document collection, DSC-SS ignored over 6,500 documents for a super shingle size of two. As for the LA Times collection, DSC-SS ignored over 1,200 documents. In comparison, DSC ignored 5052 and 636 documents, respectively. I-Match, in the worst case, ignored only four documents.

In Figure 4, we illustrate the increase in the number of documents ignored as the number of shingles used to create a single super shingle increases.

Table V. DSC-SS Short Document Filtration

Super Shingle Size	Documents Ignored	Percentage Filtered
100	220073	88.92%
75	209928	84.82%
50	189071	76.40%
25	133614	53.99%
20	112703	45.54%
15	86288	34.87%
10	54212	21.90%
5	22257	8.99%
4	16805	6.79%
2	6528	2.64%

Table VI. Post Average Document Size

Super Shingle Size	Post Avg Doc Size
100	9860
75	8123
50	6109
25	3833
20	3389
15	2963
10	2575
5	2272
4	2225
2	2140

The more shingles used to make a super shingle, the more documents are ignored.

We then ran DSC-SS algorithm against the 2 GB NIST collection with super shingle sizes of 100, 75, 50, 25, 20, 15, 10, 5, 4 and 2 shingles. In Table V, we once again show that the greater the super shingle size, the more documents ignored, thus validating our prior results using the LA Times and Excite@Home collections. In Table V, we also illustrate the percentage of the collection filtered.

The I-Match algorithm uses various term filtration techniques based on collection statistics to filter terms. We conducted 52 different filtration experiments. For most I-Match runs, only about 150 documents were filtered (less than .06% of the collection). Since our best filtration techniques take only a percentage of the document, only documents with a couple of unique terms are ignored. The only I-Match thresholding technique to filter a substantial percentage of documents, filters based on IDF values, retaining only a normalized IDF value of 0.9 or greater. This technique keeps less than 50% of the collection, similar to a DSC-SS of 50. In spite the degree of filtering, no I-Match thresholding technique dropped any significant number of documents. That is, the greatest number of documents dropped was 143 out of 247,491.

As the super shingle size increases, the average size of a document that survives the filtration process increases. In Table VI, we present the average

Table VII. Duplicate Processing Time

Algorithm	Mean Time	Std Deviation	Median Time
DSC	595.4	4.3	593.5
DSC-SS	587.6	18.5	587.1
I-Match	96.9	33.4	82.6
Syntactic	5	N/A	N/A

number of tokens per document retained after super shingling. The average token size is about six characters in length. The sizes of the documents are presented as the number of terms. Thus, multiplying by six [Baeza-Yates and Ribeiro-Neto 1999] estimates the average size of a document. This sub-collection of the web has slightly higher document sizes than the 4 K sizes reported in [Giles and Lawrence 1999]. This shows us that the DSC-SS algorithm performs poorly on web sized documents.

4.5 Runtime Performance

We divide the runtime performance experiments and results into two sections. The first set of experiments compares the I-Match and the DSC-SS algorithms on the Excite@Home test document collection, shown in Table VII. The second set of experiments compares the I-Match, DSC-SS and DSC algorithms using the 2-gigabyte NIST web collection. All experiments were run on a SUN ES-450; each process ran with about 200 MB for all algorithms.

I-Match was approximately five times faster than DSC-SS for the Excite@Home collection. The pure syntactic filtration technique ran in less than 5 seconds, but as discussed previously, only exact matches are found with this technique. Varying the threshold for super shingle sizes does not significantly influence the runtime since the same amount of work must occur. Our best performing I-Match techniques, in terms of accuracy, ran in 142 (Doc-L-90) and 134 (IDF-L-10) seconds.

The DSC and DSC-SS timings for the Excite@Home collection are comparable since the third and fourth steps of the DSC algorithm are I/O bound in nature but are relatively negligible for a small collection. The third and fourth steps in the DSC approach become a greater percentage of the cost as the collection grows, as seen in Table VIII for the 2 GB NIST collection.

We compared the run time of I-Match to the DSC and DSC-SS algorithms running against the NIST 2 gigabyte Web collection. As with the Excite@Home experiments, the parsing/indexing system builds shingle data and relevance feedback data structures when indexing a collection. Thus, preprocessing the text and creating shingle and token data times are not contained in our timing results, just the specific clustering or duplication algorithm.

As shown in Table VIII, I-Match was approximately six times faster than DSC-SS and almost 9 times faster than the DSC algorithm. The faster speed of the I-Match algorithm suite is due to the processing of fewer tokens. The

Table VIII. Processing Time for 2 GB NIST Web Collection

Algorithm	Mean Time	Std Deviation	Median Time
DSC	31838.22	807.9	30862.5
DSC-SS	24514.7	1042.1	24475.5
I-Match	3815.8	975.8	3598.8
Syntactic	65	N/A	N/A

average distinct tokens per document for the NIST 2 gigabyte collection is approximately 475, while the average document size is over 2000 terms long. Since a sliding window creating shingles produces about the same number of shingles as the size of the document, the added amount of processing is proportional. This is true for all small window sizes proportional to the total document size. If a large window size for super shingles is used, the DSC-SS approach is just a hash approach and will not match on similar documents. This ratio of distinct terms to document size is consistent with our TREC collection statistics.

The DSC algorithm has several additional steps, which are I/O bound in nature, and contributes to its additional run time. Table VIII contains an average of timing results for each of the given techniques. DSC had five experiments, using a threshold of 50%, 60%, 70%, 80% and 90%. DSC-SS had ten experiments (enumerated in Table V). I-Match results are from 52 experiments described above. Last, a syntactic filtration comparison is given. Detailed experimental results are presented as an appendix and are listed in Tables IX and XI with the legend describing the experimentation given in Table X.

4.6 Duplication in TREC Results

We examined the effects of duplication on result sets. We used the I-Match algorithm on the TREC disks 4–5, which were used for TREC 6–8. We used the NIST relevance judgments to flag documents judged by NIST. If a duplicate was found and a positive judgment was made for a given query, we checked to make sure that no false judgments were made on its duplicates. A dozen inconsistencies were found for TREC 6. Eight inconsistencies were found for TREC 7. Seventeen inconsistencies were detected in TREC 8 and 65 inconsistencies were noted for the web track of TREC 8. Examining these inconsistencies, we found documents that were identical and judged differently. TREC topic 301, judged document FBIS3-58055 relevant and FBIS3-58025 not relevant; they are the same document except for the document number. Another example for topic 301 is that document FBIS3-33287 was judged relevant and document FBIS3-41305 was not judged as relevant although these documents are identical except for the title and the document number. Similar examples were found for TREC 7 and 8 and the web track of TREC 8. While this does not diminish the usefulness of the TREC judgments, it does show that accurate duplicate document detection would eliminate these inconsistencies in test collections.

5. CONCLUSIONS AND FUTURE WORK

Algorithms for detecting similar documents are critical in applications where data is obtained from multiple sources. The removal of similar documents is necessary, not only to reduce runtime, but also to improve search accuracy. Today, search engine crawlers are retrieving billions of unique URL's, of which hundreds of millions are duplicates of some form. Thus, quickly identifying duplicate detection expedites indexing and searching. One vendor's analysis of 1.2 billion URL's resulted in 400 million exact duplicates found with a MD5 hash (S. Cicciarelly, personal communication). Reducing the collection sizes by tens of percentage points results in great savings in indexing time and a reduction in the amount of hardware required to support the system. Last and probably more significant, users benefit by eliminating duplicate results. By efficiently presenting only unique documents, user satisfaction is likely to increase.

We proposed a new similar document detection algorithm called I-Match and evaluated its performance using multiple data collections. The document collections used varied in size, degree of expected document duplication, and document lengths. The data was obtained from NIST and from Excite@Home.

I-Match relies on collection statistics to select the best terms to represent the document. I-Match was developed to support web document collections. Thus, unlike many of its predecessors, I-Match efficiently processes large collections and does not neglect small documents. In comparison to the prior state-of-the-art, I-Match ran five times faster than DSC-SS against the Excite@Home test collection and six times faster against the NIST 2 GB collection. Furthermore, unlike the efficient version of the prior art, I-Match did not skip the processing of small documents.

In terms of human usability, no similar document detection approach is perfect however; our experimentation shows the I-Match IDF-L-10 to be the most effective approach for finding duplicate documents. The ultimate determination of how similar a document must be to be considered a duplicate, relies on human judgment. Therefore, any solution must be easy to use. To support ease of use, all potential duplicates should be uniquely grouped together. Shingling approaches, including the DSC and DSC-SS approaches, however, group potential duplicate documents according to shingle matches. Therefore, any match in even a single shingle results in a potential duplicate match indication. This results in the scattering of potential duplicates across many groupings, and many false positive potential matches. I-Match, in contrast, treats a document in its entirety and maps all potential duplicates into a single grouping. This reduces the processing demands on the user.

Our future efforts will focus on the processing of corrupted text and multilingual document collections. Since our statistics are collection-based, the incorporation of foreign languages is unlikely to cause great difficulty. However, cross language processing, namely translated document processing, is likely to be very difficult.

APPENDIX

Table IX. I-Match WT2G Experiments

TREC 2 GB WEB		247491										
Experiment	Post Doc Num	Filtered	Pre Avg Dist Terms	Post Avg Dist Terms	Pre Doc Size	Post Doc Size	Dup Hash	Dup URL	Dup Total	Dup Clusters	Max Cluster	Time
baseline	247491	0	471	471	2085	2085	60	0	60	46	5	62
Doc-h-f1	247348	143	471	471	2085	2085	53535	0	53535	22019	782	6869
Doc-h-f2	247348	143	471	471	2085	2085	54398	0	54398	22243	782	6086
Doc-h-f3	247348	143	471	471	2085	2085	55578	0	55578	22484	782	5707
Doc-h-f4	247348	143	471	471	2085	2085	56756	0	56756	22710	782	5376
Doc-h-f5	247348	143	471	471	2085	2085	58235	0	58235	22975	782	5287
Doc-h-f6	247348	143	471	471	2085	2085	60905	0	60905	23520	782	3701
Doc-h-f7	247348	143	471	471	2085	2085	65183	0	65183	23926	782	3415
Doc-h-f8	247348	143	471	471	2085	2085	75169	0	75169	24286	1805	3259
Doc-h-f9	247348	143	471	471	2085	2085	108157	0	108157	22957	4288	3104
Doc-l-f1	247348	143	471	471	2085	2085	52280	0	52280	21747	782	4311
Doc-l-f2	247348	143	471	471	2085	2085	52331	0	52331	21753	782	4100
Doc-l-f3	247348	143	471	471	2085	2085	52389	0	52389	21776	782	3810
Doc-l-f4	247348	143	471	471	2085	2085	52498	0	52498	21831	782	3710
Doc-l-f5	247348	143	471	471	2085	2085	52712	0	52712	21928	782	3568
Doc-l-f6	247348	143	471	471	2085	2085	53196	0	53196	22067	782	5439
Doc-l-f7	247348	143	471	471	2085	2085	54079	0	54079	22306	782	4668
Doc-l-f8	247348	143	471	471	2085	2085	55788	0	55788	22801	782	3317
Doc-l-f9	247348	143	471	471	2085	2085	60053	0	60053	24268	1200	3174
IDF-h-f1	244348	3143	471	476	2085	2107	126710	0	126710	31065	782	2473
IDF-h-f2	247265	226	471	472	2085	2086	60858	0	60858	23636	782	3012
IDF-h-f3	247317	174	471	471	2085	2085	56616	0	56616	22914	782	3551
IDF-h-f4	247339	152	471	471	2085	2085	54864	0	54864	22576	782	3924
IDF-h-f5	247341	150	471	471	2085	2085	54130	0	54130	22358	782	4081
IDF-h-f6	247343	148	471	471	2085	2085	53466	0	53466	22012	782	4115
IDF-h-f7	247344	147	471	471	2085	2085	53107	0	53107	21925	782	4266
IDF-h-f8	247345	146	471	471	2085	2085	52816	0	52816	21900	782	4383
IDF-h-f9	247345	146	471	471	2085	2085	52574	0	52574	21829	782	4472
IDF-l-f1	247318	173	471	471	2085	2086	52391	0	52391	21813	782	4399
IDF-l-f2	246870	621	471	472	2085	2088	52732	0	52732	22022	782	3773
IDF-l-f3	245724	1767	471	474	2085	2097	53372	0	53372	22437	423	3133
IDF-l-f4	244098	3393	471	475	2085	2101	55271	0	55271	22994	238	2897
IDF-l-f5	240678	6813	471	479	2085	2122	58963	0	58963	24275	238	2645
IDF-l-f6	227196	20295	471	492	2085	2171	57403	0	57403	24995	130	2696
IDF-l-f7	203845	43646	471	519	2085	2285	50840	0	50840	24132	30	2535
IDF-l-f8	165898	81593	471	568	2085	2497	35423	0	35423	20139	10	2430
IDF-l-f9	114434	133057	471	616	2085	2724	13242	0	13242	11696	2	2608
DocR-O-f1f9	247348	143	471	471	2085	2085	57316	0	57316	23226	782	4050
DocR-O-f2f8	247348	143	471	471	2085	2085	54248	0	54248	22293	782	3576
DocR-O-f3f7	247348	143	471	471	2085	2085	53201	0	53201	22046	782	3869
DocR-O-f4f6	247348	143	471	471	2085	2085	52556	0	52556	21849	782	4211
DocR-I-f1f9	247200	291	471	472	2085	2086	53726	0	53726	22087	782	5267
DocR-I-f2f8	247200	291	471	472	2085	2086	54743	0	54743	22294	782	4043
DocR-I-f3f7	246907	584	471	472	2085	2087	55840	0	55840	22525	782	3517
DocR-I-f4f6	245740	1751	471	474	2085	2090	57865	0	57865	22963	798	3250
IDFR-O-f1f9	244743	2748	471	476	2085	2106	88641	0	88641	25354	676	2622
IDFR-O-f2f8	247291	200	471	472	2085	2086	57349	0	57349	22981	782	2991
IDFR-O-f3f7	247334	157	471	471	2085	2085	54415	0	54415	22472	782	3577
IDFR-O-f4f6	247346	145	471	471	2085	2085	53030	0	53030	22173	782	4088
IDFR-I-f1f9	247293	198	471	472	2085	2086	52703	0	52703	21905	782	4112
IDFR-I-f2f8	246759	732	471	473	2085	2088	53395	0	53395	22245	782	3419
IDFR-I-f3f7	245457	2034	471	475	2085	2099	55295	0	55295	22997	423	2921
IDFR-I-f4f6	241515	5976	471	479	2085	2117	62341	0	62341	24360	238	2615

Table X. I-Match Experiment Legend

I-Match Experiment	*Description*
baseline	Syntactic one-pass hash approach, stemming and removable of special character terms.
Doc (% Doc approach)	Takes the X percent of the document based on idf values of the terms. **l** = Highest on the left side of tree. So the terms with the X highest idf values are used. **h** = Lowest on the left side of tree. So the terms with the X lowest idf values are used.
IDF (IDF approach)	Filters terms that don't meet the normalized idf value threshold are removed. **l** = Terms with idf value is greater than the filter value, the term are kept. **h** = Terms with idf value is lower than the filter value, the term are kept.
DocR (%Doc Range approach)	Takes the X percent of the document based on idf values of the terms. The range takes either the middle X percent or the outer X percent based on the l or h value. **I** = The inner X percent of terms based on idf values are kept. **O** = The outer X percent of terms based on idf values are kept.
IDFR (IDF range approach)	Filters terms based on normalized idf values. Thus if the term is in the range of idf values it is kept for the final hash. **I** = Keeps terms with idf values between the two values **O** = Keeps terms with idf values greater and less than the 2 filter values.

Table XI. WT2G DSC-SS Experiments

WT2G								
	Post Num Docs	Filtered	Pre Dist Shingles	Post Dist Shingles	Pre Doc Size	Post Doc Size	Dup Clusters	Time
DSC-SS-2	239886	7605	470	484	2079	2140	41001	23465
DSC-SS-4	229609	17882	470	503	2079	2225	35638	23441
DSC-SS-5	224157	23334	470	513	2079	2272	33771	26064
DSC-SS-10	192202	55289	470	578	2079	2575	27224	24984
DSC-SS-15	160126	87365	470	659	2079	2963	22667	23609
DSC-SS-20	133711	113780	470	746	2079	3389	19183	24318
DSC-SS-25	112800	134691	470	833	2079	3833	16249	23669
DSC-SS-50	57343	190148	470	1255	2079	6109	7546	25366
DSC-SS-75	36486	211005	470	1626	2079	8123	4948	24147
DSC-SS-100	26341	221150	470	1926	2079	9860	3673	26084

ACKNOWLEDGMENT

The authors thank the comments of the anonymous referees.

REFERENCES

BAEZA-YATES, R. AND RIBEIRO-NETO, B. 1999. *Modern Information Retrieval*. Addison Wesley.

BRIN, S., DAVIS, J., AND GARCIA-MOLINA, H. 1995. Copy Detection Mechanisms for Digital Documents. In *Proceeding of the Special Interest Group on Management of Data* (SIGMOD'95) (San Francisco, CA., May). 298–409.

BRODER, A., GLASSMAN, S., MANASSE, S., AND ZWEIG, G. 1997. Syntactic clustering of the web. In *Proceedings of the Sixth International World Wide Web Conference* (WWW6'97) (Santa Clara, CA., April). 391–404.

BUCKLEY, C., CARDIE, C., MARDIS, S., MITRA, M., PIERCE, D., WAGSTAFF, K., AND WALZ, J. 1999. The Smart/Empire TIPSTER IR System. In *Proceedings of TIPSTER Phase III* (San Francisco, CA.). 107–121.

CHOWDHURY, A., HOLMES, D., MCCABE, M. C., GROSSMAN, D., AND FRIEDER, O. 2000. The use of fusion with AIRE at TREC-9. In *Proceedings of the Ninth Text Retrieval Conference* (TREC-9, Gathersburg, MD, November).

FRIEDER, O., GROSSMAN, D., CHOWDHURY, A., AND FRIEDER, G. 2000. Efficiency considerations in very large information retrieval servers. *J. Dig. Inf. 1*, 5 (Apr).

GILES, L. AND LAWRENCE, S. 1999. Accessibility and distribution of information on the web. *Nature 400*, 107–109.

GROSSMAN, D., HOLMES, D., AND FRIEDER, O. 1993. A DBMS Approach to IR in TREC-4. In *Proceedings of the Fourth Text Retrieval Conference* (TREC-4) (Gaithersburg, Maryland, November).

HEINTZE, N. 1996. Scalable document fingerprinting. In *Proceedings of the Second USENIX Electronic Commerce Workshop* (Oakland, CA., November). 191–200.

KJELL, B., WOODS, W., AND FRIEDER, O. 1994. Discrimination of authorship using visualization. *Information Processing and Management*. Pergamon Press *30*, 1 (Jan), 141–150.

KWOK, K. L. 1996. A new method of weighting query terms for Ad-hoc retrieval. In *Proceedings of the 19th Annual International*. *ACM Special Interest Group on Information Retrieval* (SIGIR '96) (Zurich, Switzerland, August). pp.187–195.

LAWRENCE, S. AND GILES, C. L. 1998. Searching the World Wide Web. *Science*. *280*, 5360, 98–100.

NIH. 2000. http://nccam.nih.gov/, The National Institutes of Health (NIH), National Center for Complementary and Alternative Medicine (NCCAM), April 12, 2000.

NIST. 1995. Secure Hash Standard, U.S. Department of Commerce/National Institute of Standards and Technology, FIPS PUB 180-1 (April 17).

PORTER, M. 1980. An algorithm for suffix stripping. *Program 14*, 3, 130–137.

ROBERTSON, S. WALKER, S., AND BEAULIEU, M. 1999. Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC and interactive, *Proceedings of the 7th Text Retrieval Conference* (TREC-7'99) (July). 253–264.

ROCCHIO, J. 1971. Relevance Feedback in Information Retrieval. In *The Smart System—experiments in automatic document processing*, pages 313–323. Prentice Hall, Englewood Cliffs, NJ.

SALTON, G., YANG, C. S., AND WONG, A. 1975. A vector-space model for information retrieval. *C. ACM*. ASIS. *18*, 11, 613–620.

SANDERSON, M. 1997. Duplicate detection in the Reuters collection. *Technical Report* (TR-1997-5) *of the Department of Computing Science at the University of Glasgow*, Glasgow G12 8QQ, UK.

SCOTTI, R. AND LILLY, C. 1999. George Washington University Declassification Productivity Research Center. http://dprc.seas.gwu.edu. July 31.

SHIVAKUMAR, N. AND GARICA-MOLINA, H. 1998. Finding near-replicas of documents on the web. In *Proceedings of Workshop on Web Databases* (WebDB'98) (Valencia, Spain, March). 204–212.

SINGHAL, A., BUCKLEY, C., AND MITRA, M. 1996. Pivoted Document Length Normalization, *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR'96) (Zurich, Switzerland, August). 21–29.

SMART FTP site: 2000. ftp://ftp.cs.cornell.edu/pub/smart/. January 19.

SMEATON, A., KELLEDY, F., AND QUINN, G. 1997. Ad-hoc retrieval using thresholds, WSTs for French monolingual retrieval, Document-at-a-Glance for high precision and triphone windows for spoken documents. In *Proceedings of the Sixth Text Retrieval Conference* (TREC-6, Gathersburg, Maryland).

Received July 2000; revised April 2001; accepted November 2001