Efficient Data-Sensitive Techniques for Parallel Retrieval of Keyword-indexed Information[1]

Rakesh M. Verma, S.L. Narayanan, H.Liu and A.Gupta,

Computer Science Department
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

UH-CS-10-10
October 20, 2010

Keywords: Information Storage and Retrieval, Efficient Query Evaluation, Parallel I/O, Digital Libraries

Abstract

Keyword based search of data, such as documents, maps, images, audio and video data, is an everyday activity for many millions of people with myriad uses, e.g., scientific computing, digital libraries, the web, catalogs, geographical information systems, music servers, etc. In this paper, we present several declustering algorithms based on existing similarity measures as well as their generalizations. Experiments show that the new declustering methods are indeed more efficient in declustering times and close in terms of parallel query times than quadratic declustering methods based on existing similarity measures. Our declustering algorithms are *sublinear* in the number of comparisons and scalable with increasing data and disks. The new methods are also capable of handling streaming data quite efficiently. We present some negative results on random and profile-based sampling. Although the new sampling strategies based on profiles of the documents outperform the old sampling strategies, which do not use a profile, they are still worse than random and round-robin. Further testing is required to confirm the sampling results.

[1] Research supported in part by NSF grant CCF 0306475.

Efficient Data-Sensitive Techniques for Parallel Retrieval of Keyword-indexed Information

Abstract

Keyword based search of data, such as documents, maps, images, audio and video data, is an everyday activity for many millions of people with myriad uses, e.g., scientific computing, digital libraries, the web, catalogs, geographical information systems, music servers, etc. In this paper, we present several declustering algorithms based on existing similarity measures as well as their generalizations. Experiments show that the new declustering methods are indeed more efficient in declustering times and close in terms of parallel query times than quadratic declustering methods based on existing similarity measures. Our declustering algorithms are *sublinear* in the number of comparisons and scalable with increasing data and disks. The new methods are also capable of handling streaming data quite efficiently. We present some negative results on random and profile-based sampling. Although the new sampling strategies based on profiles of the documents outperform the old sampling strategies, which do not use a profile, they are still worse than random and round-robin. Further testing is required to confirm the sampling results.

Categories and subject descriptors: H.3.2, H.3.3 [Information Storage and Retrieval]
General Terms: Algorithms, Performance.
Keywords: Efficient Query Evaluation, Parallel I/O, Digital Libraries

1. INTRODUCTION

Keyword based search of data, such as documents, maps, images, audio and video data, is an everyday activity for many millions of people with myriad uses, e.g., scientific computing, digital libraries, the web, catalogs, geographical information systems, music servers, etc. The growing imbalance between the speeds of processors and I/O devices has resulted in the I/O subsystem becoming a bottleneck in many applications (see [16] and references cited therein) including especially the above mentioned areas. The ever-rising volumes of data (e.g., over three billion web pages, NASA's EOSDIS project [16], etc.) and its dynamic nature compel the development of high-performance parallel I/O servers for these data intensive applications to allow: (i) more complex queries, (ii) increased utilization of system resources, (iii) reduced latencies, and (iv) to avoid the imminent severe performance degradation.

The I/O bottleneck can be alleviated by designing fast algorithms that decluster the data, i.e., distribute it across multiple disks so that data on the same disk are as dissimilar as possible, and parallelizing the I/O operations. There are two key optimization criteria: maximum parallelism for queries, which are typically boolean combinations of phrases of keywords (e.g., cryptography and not ``private key''), and uniform load of data across disks.

Automatic allocation of keyword-indexed information on parallel disks is a critical technology for dealing with the explosive growth of data in scientific computing, digital libraries and the web. In this paper, we present: (i) novel similarity measures and declustering algorithms for keyword-indexed information, and (ii) compare the performance in parallel query time, declustering time, and the scalability of the new measures against declustering algorithms derived from existing measures such as Dice's coefficient and Jaccard's coefficient. Similarity measures are needed to distribute the data across multiple disks so that data on the same disk are as dissimilar as possible, which reduces the probability of one disk becoming a bottleneck.

Note that the problem of search engines such as Google [12]-- that even though keep copies of every web page -- is a different and *easier* one, since they typically do not retrieve the actual data itself but hyperlinks, which are orders-of-magnitude smaller, to the data the user is interested in. Of course, they provide links to ``cached copies'' but they need retrieve this only when the user explicitly requests it, which is a sequential process in that the user first visits one site and then another and so on. Despite having a relatively easier problem to contend with, search engines still use ad-hoc parallelism, lazy computation trick of postponing the merging of inverted lists hoping that the user will be satisfied by the first page of links, and limit the number of keywords for relief. All these tricks for an easier problem serve to bolster our arguments about the difficulty of fast access to the data itself.

In this paper we present declustering algorithms or the data by designing and validating *new* similarity measures for keyword-indexed data including comparisons with methods based on exisiting measures. Our work addresses several limitations that are common to existing work on parallel I/O servers for keyword-indexed data as well as investigates important new research directions. Our declustering algorithms are *sublinear* in the number of comparisons and scalable with increasing data and disks. Because of the large volumes of data even linear-time algorithms can be slow, *whereas most existing data-sensitive methods are quadratic in the number of comparisons*. Furthermore, our algorithms are designed for *streaming* data. Such algorithms do not assume that all data is available beforehand, but instead make decisions as the data arrives. We present analytical study and experimental validation of the scalability of our algorithms with increasing data and disks, on collections drawn from Literature Online. Applications of proposed work include also automatic text categorization, Internet resource discovery services, and could include clustering as well.

Experimental results show that standard techniques such as round-robin or random allocation and their variants can result in more than 20% higher disk response time compared to similarity-based data allocation schemes using Dice's or Jaccard's coefficient, even for small collections and simple queries with at most three keywords [18, 2]. Excite has provided us with a large database of over one million actual user queries that guides one of the query model in our experiments.

Related Work. Multiple disks were proposed to achieve fault-tolerance from disk failures. Subsequently, they have been used for declustering also to increase parallelism in update and query operations. Earlier declustering methods such as round-robin, hashing, or range-partition based methods were developed for traditional file structures. Recently, several methods have been proposed for declustering spatial databases [14] for similarity or range queries. These approaches are either static (i.e. the data is known in advance) or based on uniformly partitioning the space itself (in this case the performance degrades with clustered data distributions). There is also some work on declustering for multimedia applications [19]. In contrast to the static case where all the data is available before the declustering algorithm is invoked and therefore optimal algorithms can be derived, in the streaming data case decisions to assign data to a disk must be made as the data arrives and are typically never/rarely revisited. These decisions, therefore, must necessarily be suboptimal except in rarely occurring simple situations. For the case of documents and boolean keyword or phrase queries, there is hardly any previous work on declustering the documents to the best of our knowledge. In [2], the authors adapt a spatial declustering method for keyword-indexed data and compare it against the simple matching model, round-robin and random allocation.

There are two efforts on partitioning inverted files [1], an index used in information retrieval systems. In [8] the performance of inverted indices is considered for a shared-everything multiprocessor machine, and in [6] the performance of inverted indices is considered for a shared-nothing system consisting of multiple processors and multiple disks. In both these papers, the authors focus on retrieving the postings files for the documents in parallel, and do *not* consider the problem of retrieving the documents themselves in parallel. Thus, the distribution of the documents themselves is not an explicit goal except that it occurs as a side-effect in [6] in one strategy called the *disk* strategy, wherein the documents are divided between the disks in a round-robin manner (see also [15] for a discussion). For some work on distributed information retrieval see [13]. For some general considerations involved in heterogeneous disk systems we refer to [20].

The main problem with existing work on similarity measures (this including proximity-based schemes such as [14]) is that the similarity function is between *two* data items or between a query and a *single* data item. This means that in a declustering algorithm when a new data item is to be assigned to a disk, it must be compared with every item previously assigned to the disks and an aggregation function must be applied such as sum of the similarities or maximum similarity with items on each disk. This leads to a declustering algorithm that requires *quadratic* comparisons in the number of data items assigned.

2 FRAMEWORK AND DEFINITIONS:

This section introduces the definitions of the symbols used in subsequent discussions (Table 2-1) and the basic framework for declustering documents under multi-disks architecture.

Table 2-1 Notations and Definitions

Symbol	Definition
N	Total number of documents in the collection
D	Total number of disks for N documents
D_d	The d^{th} disk, where $d \in \{1,2,3,...,D\}$
t_j	The j^{th} document in the collection where $j \in \{1,2,3,...,N\}$
N_d	The total number of documents on the d^{th} disk
sim (t_a, t_j)	The similarity of the a^{th} document and the j^{th} document
sim_{taDd}	The sum of the similarity of Document t_a with each document on Disk D_d $a \in \{1,2,3,...,N\}$; $d \in \{1,2,3,...,D\}$
t_a	Document to be assigned to a disk
t_{iDd}	The i^{th} document already assigned to Disk D_d where $i \in \{1,2,3,..., N_d\}$ $d \in \{1,2,3,...,D\}$

The assignment of a document to a disk is based on the calculation of its similarity, respectively, with each of the documents already on the disks. Following are the steps to assign a document to a disk based on the models of Section 3. (The first two steps are modified into a single step for the models of Section 4):

a) Calculate the similarity of Document t_a and Document t_{iDd}: sim (t_a, t_{iDd}) by similarity models. Details in Section 3.

b) Calculate the sum of the similarity on each disk for t_a: $sim_{t_a D_d} = \sum_{i=0}^{N_d} sim(t_a, t_{iD_d})$

c) Find disk D_{min} which has the minimum similarity, sim_{taDmin}:

$$sim_{t_a D_{min}} = \min \{sim_{t_a D_1}, sim_{t_a D_2}, ..., sim_{t_a D_d}\}$$

d) Assign Document t_a to Disk D_{min}

3 EXISTING SIMILARITY MODELS

This section describes the existing similarity models tested for the study. In keyword-indexed information retrieval system, each document can be viewed as a set of tokens. Tokens can be letters, words or strings of any length. A text document can be broken up into a set of

tokens. The set of the tokens, the occurrences of each token and the position of it are the key elements to distinguish one document from another. Most of the similarity models use these key elements to calculate similarity of two documents.
In the following definitions, Q is defined as the set of tokens for the first document t_q and P as the set of tokens of the second document t_p. The intersection of Q and P is denoted as $|Q \cap P|$. The union of Q and P is denoted as $|Q \cup P|$. $|Q|$ and $|P|$ represent the number of tokens in set Q and set P, respectively.

3.1 Simple Matching (coordination level match) [1]:
The Simple Matching Model is to calculate the intersection of token sets of two documents. $sim\ (t_q, t_p) = |Q \cap P|$

3.2 Dice's Coefficient [1]:
The Dice's Coefficient is the intersection of the two sets normalized by the total number of tokens of the two documents.

$$sim\ (t_q, t_p) = \frac{2|Q \cap P|}{|Q| + |P|}$$

3.3 Jaccard's Coefficient Model [1]:
The Jaccard's Coefficient Model is to calculate the intersection of the sets of two documents normalized by the union of the two sets.

$$sim\ (t_q, t_p) = \frac{|Q \cap P|}{|Q \cup P|}$$

3.4 **Cosine Coefficient [1]:**

$$sim\ (t_q, t_p) = \frac{|Q \cap P|}{|Q|^{1/2} * |P|^{1/2}}$$

3.5 Overlap Coefficient[11]:

$$sim\ (t_q, t_p) = \frac{|Q \cap P|}{\min(|Q|, |P|)}$$

3.6 Broder Scheme [4] [5]:
The Broder Scheme is an intersection-based model with chunking strategy. It views a document as a series of tokens. Each token contains more than one keyword and tokens can be overlapped. The size of the token is determined by the number of keywords it contains. The definition of the model is as follows:

$$sim\ (t_q, t_p)_w = \frac{|Q_w \cap P_w|}{|Q_w \cup P_w|}$$

where w is the size of the token. For example, if we define a token as containing two keywords, the token size w is 2. The tokens may overlap by one keyword or they may not overlap. If w is 1, this scheme is the same as Section 3.3, Jaccard's Coefficient Model.

3.7 SCAM (Relative Frequency Model) [10] [11]:
Relative Frequency Model is a comparison scheme based on the word occurrences of a document. It combines relative frequencies of words as indicators of similar words with cosine similarity measurement as stated by Shivakumar and Garcia-Molina [10]. The Relative Frequency Model is defined as follows:
First, define the closeness set c(Q,P) to contain words w_i. The closeness set is an indicator which indicates the set of words that occur with similar frequency in the two documents. A word w_i is in c(Q,P) if it satisfies the following condition: $\varepsilon - \left(\frac{F_i(Q)}{F_i(P)} + \frac{F_i(P)}{F_i(Q)}\right) > 0$

Where $\varepsilon = (2+,\infty)$, is a tolerance factor and it is a user-tunable parameter. $F_i(Q)$ is the number of occurrences of word w_i in t_q. $F_i(P)$ is the number of occurrences of word w_i in t_p. According to Shivakumar and Garcia-Molina [10] this model works well when $\varepsilon = 2.5$. Second, define the subset measurement of document t_q and document t_p to be:

$$subset\ (t_q, t_p) = \frac{\sum_{w_i \in c(Q,P)} (\alpha_i^2 * F_i(Q) * F_i(P))}{\sum_{i=1}^{N} (\alpha_i^2 * F_i^2(Q))}$$

where α is a weighing factor of each word w_i. In Shivakumar and Garcia-Molina [10] it was uniformly defined as 1. Then, the similarity of the two documents is determined as follows: $sim(t_q, t_p) = \max\{subset(t_q, t_p), subset(t_p, t_q)\}$. From the subset measurement, we can see that this model only takes in the words which fall into a closeness set.

3.8 SCAM model with Chunk Strategy [3][10][11]:
A document can be broken up into more primitive units called chunks. There are different ways to choose chunks. For a text document, a chunk can be a word, a sentence or a string of any length. Chunks can be overlapping or non-overlapping. Several chunking strategies have been discussed in [11].

The SCAM Model [10] and the COPS Model [3] are two copy detection mechanisms for a digital library. The SCAM model is based on words, i.e. a chunk contains one word. It believes that word access patterns have more locality then sentence access patterns. In COPS, documents are broken up into sentence, i.e. a chunk contains one sentence. While SCAM has been shown to work well over COPS [3], it loses position information which resulted in more false positives than COPS. In the present study the SCAM Model's algorithm is combined with a chunking strategy that takes two consecutive words as a chunk and the chunks are overlapped by one word.

3.9 Round Robin Model:

This model assigns the documents to the disks in a Round Robin way regardless of the similarity between the documents. The following formula determines the disk to be chosen for a document assignment: $D_d = j \% D$, where D_d is the disk selected, j is a counter incremented by 1 for each document assignment. D is the total number of disks.

3.10 Random Selection Model: The Random Selection model assigns the documents to a disk D_d randomly with each disk being equally selected.

3.11 **tf-idf Model [1]:** This is standard (please see [1]).

4 NEW SIMILARITY MEASURES

We have generalized the first five models above by denoting Q to be the *set of all tokens* for the documents $t_1, t_2, \ldots, t_q$ on the disk being considered and P as the set of tokens of the document t_p under consideration. The new models are denoted by version 2.0 below and the old models by version 1.0. For simple matching model this generalization gives the same similarity value. Note that a document is not compared individually with all the other documents, which means that the overall time complexity for declustering the entire collection is O(K(d)D), where K(d) denotes the sum of the number of keywords over all the documents and D is the number of disks. This complexity is obtained by constructing one structure for each disk indexed by keywords in the collection containing the frequency of each keyword among the documents on the disk for both set and Multiset options. The number of comparisons to decluster the collection complexity is a sublinear function in the size of the entire collection.

5 EVALUATION PROCESS:

Section 5.1 below describes the basic evaluation process and Section 5.2 presents the simulation program architecture based on the requirements of the evaluation process.

Our simulation program is designed to evaluate all eleven similarity models through query operation. The evaluation process is as follows:

- Assign all documents in a collection to a set of disks based on a similarity quantification measurement. To assign document k to a disk, we adopt a document assignment method from [2]. In this method, a sum of similarity of each document on each of the disks with document k is computed. The disk with the minimum sum is the one document k will be placed on. If there are more than one disk giving the same minimum, document k will be assigned to the disk which has the least number of blocks (in our experience least number of documents is inferior).
- Perform several sets of queries to retrieve the documents from the disks.
- Since documents reside on a disk in blocks, a document can be viewed as a set of blocks. The document retrieval time is measured by three parameters: a) access time, b) transfer rate and c) number of blocks to be retrieved from the disk.
- Documents located on different disks can be retrieved in parallel. The performance of each similarity model is evaluated by the maximum document retrieval time of the query results.

6 EXPERIMENTS:

The experiment set up includes document collections, query models, set options, similarity models and related evaluation parameters.

6.1 Document Collections:

Three collections of text documents are selected for similarity model evaluation. Table 6-1 shows their characteristics.

Table 6-1: Characteristics of Document Collections

Collections	Number of Documents	Minimum Size	Maximum Size	Total Size	Average Size	Standard Deviation
C1	50	6KB	64KB	1.3MB	26 KB	16.3
C2	150	2KB	341KB	9.5MB	65 KB	24.4
C3	300	2KB	51KB	9.5MB	33 KB	4.5

Collection 1 has the least documents. Collection 3 contains the most documents but has the same total size as Collection 2. The sizes of the documents in Collection 2 vary in a bigger range than those in Collection 3 and Collection 1.

6.2 Query Models:

Since there is no agreement on the distribution of keywords in user queries, we adopt the Uniform query model to test our similarity model. The Uniform query model assumes the uniform term distribution. Each term in the user query is equally and randomly selected for the global vocabulary list. Before generating the query, we construct a vocabulary distribution list for the entire document collection. The list is sorted in descending order by the occurrence of each word in a text file. Different query sets are formed by the randomly selected keywords from a fraction of the distribution list. Table 6-2 lists the parameters for the generation of query sets and Table 6-3 lists the set of queries for similarity evaluation.

Table 6-2: Parameters for Query Generation

Parameters	**Value**	**Description**
Stop Words	0	number of stop words eliminated
Keywords Distribution	40	fraction of keyword list in percentage of total size of vocabulary for query keyword generation

Document Threshold	80	number of document in which the word appears in percentage of total number of documents

Table 6-3: Sets of Queries Tested

Query set	Description	Number of queries
1	Single keyword queries	1000
2	2 keywords, "AND" only	1000
3	3 keywords, "AND" only	1000
4	6 keywords, "AND" only	1000
5	2 keywords, "OR" only	1000
6	3 keywords, "OR" only	1000
7	6 keywords, "OR" only	1000
8	Excite query set*	4463
Total number of queries		**11463**

*Amanda Spink and Judy Bateman [7] analyzed transaction logs of a set of 51,473 queries posed by 18,113 users of the Excite search engine service. They presented a set of statistical data resulting from their analysis. A set of queries called "Excite query set" is generated based on their data. The query set contains keywords from one through ten and there are both "AND" queries and "OR" queries in the set.

6.3 Set model vs. Multiset model:

When quantifying the intersection or union of two sets of keywords for two documents, there are two options used to define the keyword set: Set or Multiset. The following tables (Tables 6-4 and 6-5) give examples of the two options for a keyword set and its boolean calculation result:

Table 6-4 Example of Set option and Multiset option:

Document	Keywords in the document	Set option of keywords	Multiset option of keywords
t_q	d,e,a,d,b,a,a	{a, b, d, e}	{a, a, a, b, d, d, e}
t_p	c, a, d, b, a	{a, b, c, d}	{a, a, b, c, d}

Table 6-5 Examples of intersection and union calculation for both options:

Options	Intersection	Union
Set	{a, b, d}	{a, b, c, d, e}
Multiset	{a, a, b, d}	{a, a, a, b, c, d, d, e}

The Set option only takes distinguished tokens of two documents to form a set of keywords while Multiset option takes consideration of the frequency of the token inside each document. In this study, we use both Set option and Multiset options when evaluating Models 1 through 6 in Section 3 and Models 1 through in Section 4.

6.4 Chunking Strategy:

A full text document can be broken up into a series of smaller tokens called chunks. A chunk can further be broken into several units. A unit can be a word, a sentence or a string of any length. In this paper we use word as a unit for the chunk and one chunk contains one unit, i.e. one chunk contains one word. When chunk size is one, there is no overlap.Broder Scheme is the same as the Jaccard's Coefficient Model in terms of its Boolean-based similarity method. We will not list the results of Broder Scheme since we do not test Broder's sampling strategy in this study. The chunking strategy is tested only on Model 8, "Relative Frequency Model with Chunking Strategy".

6.5 Similarity Models: Table 6-6 lists the similarity models evaluated in this study:

Table 6-6: Similarity Models to be Evaluated

Model ID	Description	Set Mode	Chunk Size	Overlap
1s$/1m$	Simple Matching	Set/Multiset	1	0
2s$/2m$	Dice's Coefficient	Set/Multiset	1	0
3s$/3m$	Jaccard's Coefficient	Set/Multiset	1	0
4s$/4m$	Cosine Coefficient	Set/Multiset	1	0
5s$/5m$	Overlap Coefficient	Set/Multiset	1	0
6sv1/6mv1	Broder Scheme	Set/Multiset	1	0
7	SCAM (Relative Frequency Model)*			
8	SCAM model with Chunk *		2	1
9	Round Robin method			
10	Random selection method			
11	tf-idf method			

* For Relative Frequency Model, we chose $\varepsilon = 2.5$ and uniformly weighing parameter $w_i = 1$ [10]. $ can be version 1.0 or version 2.0.

6.6 Disk Selection: The document declustering test is performed on 5, 10, 15, 20 and 25 set of disks**.**

6.7 Testing Parameters:

The following table lists parameters and base values for document retrieval time evaluation:

Table 6-7: Parameters & Base Values for Retrieval Time Evaluation [6]

Parameters	Value	Description
Block size	2000	number of bytes per disk block
Access Time	6	disk access time in ms
Transfer Rate	1300	bytes per ms

6.8 Sampling:

Before assigning a document to a disk, its similarity with every document already on the disks has to be computed. In practice it is a very costly process, both on time and resources. If a sampling strategy can be applied to the similarity computation, the cost can be reduced. We study two different sampling strategies. In the *first* strategy, we randomly select a number of documents from each disk to compute the similarity with the new document. The number we chose is based on two parameters: the total number of documents N in the collection and the total number of disks D. In the *second* strategy, we construct a profile of all the documents on each disk based on the number of keywords in each document. We partition the interval (0, maximum number of keywords over documents in collection) into ranges of size 500 keywords each. On each disk, we take a sample of documents from each range in proportion to the number of documents in that range. For both strategies, two formulas are used to determine the number of sample documents to be used for computing the similarity with the new document: Number of sampled documents on a disk = $\log_2(N/D)$ or Number of sampled documents on a disk = $\sqrt{N/D}$.

7 RESULTS

The query sets in Section 6.2 are used to test the similarity models in Section 6.5. Figure 7-1 shows that the generalizations of the first five models of Section 3 have significantly smaller declustering times than the older versions.

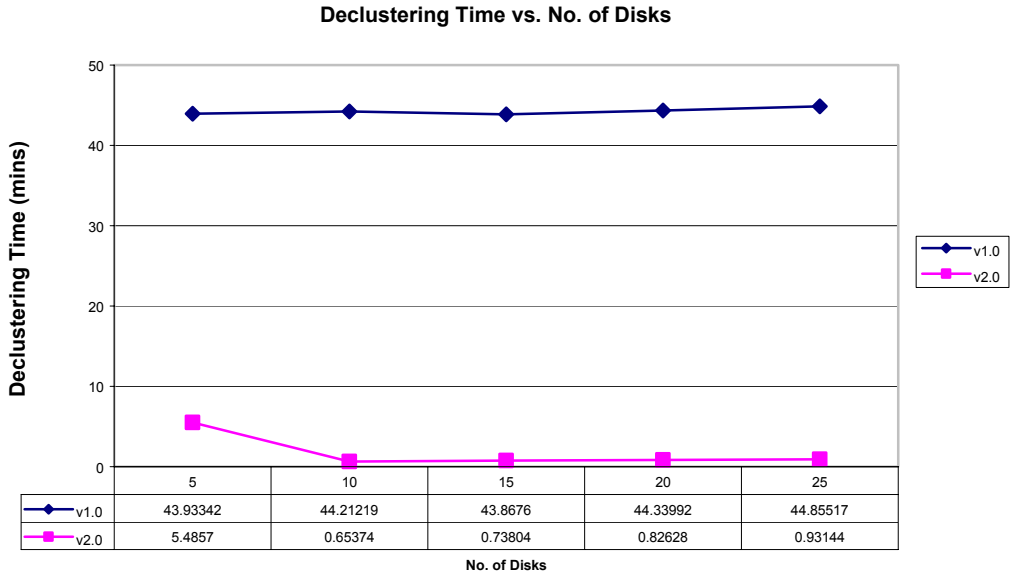


Figure 7-1 Declustering times for the first five models comparing version 1.0 and version 2.0

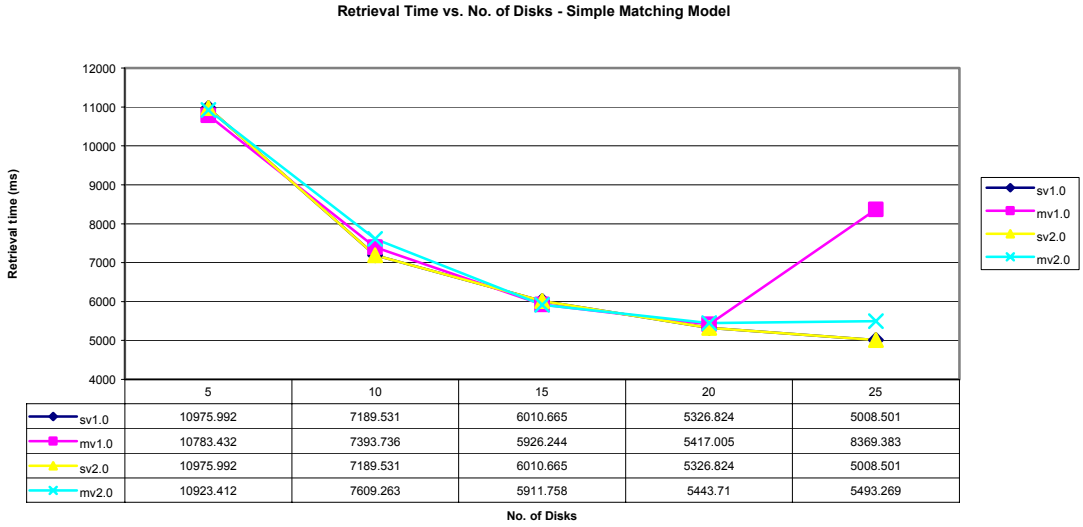
Next we present the query times in ms for the two data-insensitive methods and 5 values of D. The last column is the sum.

Table 7-1: Retrieval time (in ms) for 5, 10, 15, 20 and 25 disks and data insensitive methods.

Round Robin	12347.47	9523.073	6606.869	7468.242	5406.327	41351.98
Random Disk	11884.85	9084.693	7590.065	6140.635	5873.84	40574.08

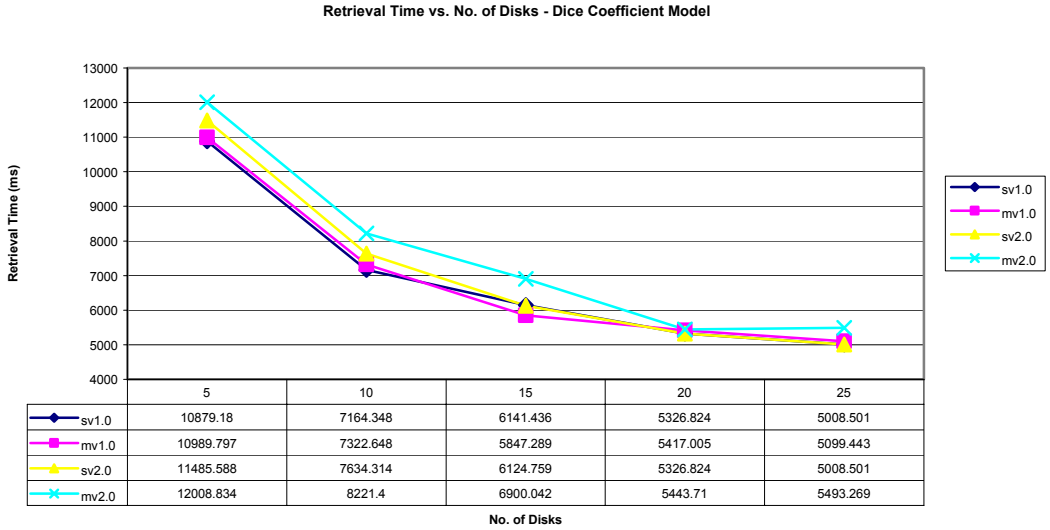
7.1 Models with Set Option vs. Models with Multiset Option:

Figures 7.1-1 through 7.1-3 compare models with Set option and Multiset option for Document Collection 2 on the first seven query sets (more on the Excite query set later). The figures shows that the Set option outperforms Multiset option in most cases, except that Multiset is better for version 1.0 Jaccard's and Cosine Coefficient Models. For lack of space, results are only shown for the best 3 models. Interestingly, we found that for collections C1 and C2 the unnormalized SCAM model also performs very well but we omit the results here, again for lack of space. Both versions of all three data sensitive models shown significantly outperform the data insensitive models.



	5	10	15	20	25
sv1.0	10975.992	7189.531	6010.665	5326.824	5008.501
mv1.0	10783.432	7393.736	5926.244	5417.005	8369.383
sv2.0	10975.992	7189.531	6010.665	5326.824	5008.501
mv2.0	10923.412	7609.263	5911.758	5443.71	5493.269

Figure 7.1-1 Multiset vs Set option for Simple Matching Model, 1st 7 query types and collection C2



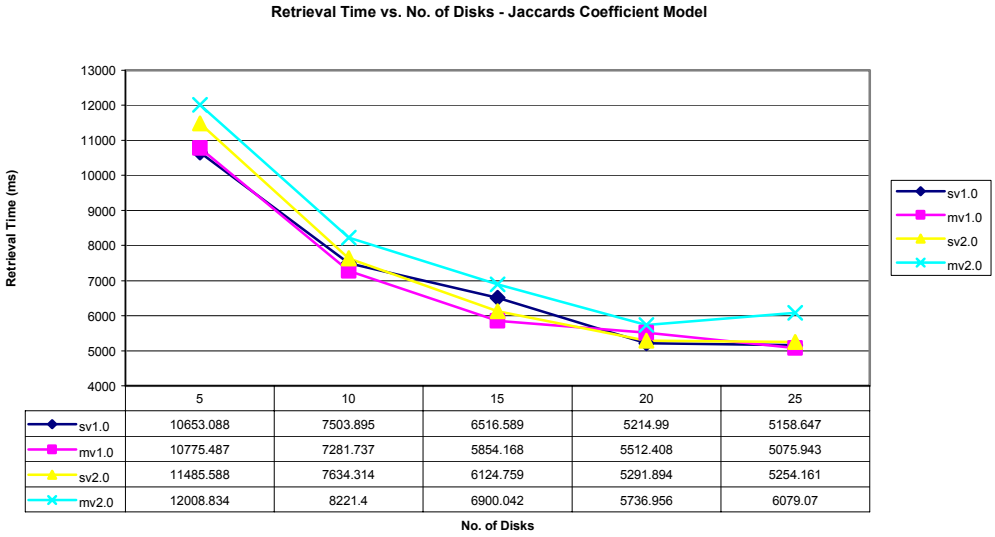
	5	10	15	20	25
sv1.0	10879.18	7164.348	6141.436	5326.824	5008.501
mv1.0	10989.797	7322.648	5847.289	5417.005	5099.443
sv2.0	11485.588	7634.314	6124.759	5326.824	5008.501
mv2.0	12008.834	8221.4	6900.042	5443.71	5493.269

Figure 7.1-2 Multiset vs Set option for Dice's Coefficient Model,1st 7 query types and collection C2

7.2 Chunking Strategies:

Different chunking strategies are used to test the SCAM Relative Frequency Model (Model 7). Model 7 is the SCAM model with chunk size being one. Model 8 is the SCAM Model with chunk size being two and overlap being one. Results (omitted) show that SCAM model with chunk size being one is superior over the same model with chunk size being two for "OR" query sets. The latter is worse than Round Robin and Random Selection methods for "OR" query sets, but it is better than Model 7 on "AND" query set.

When a user query is submitted, the query is broken down into a series of unique keywords. The query result for each unique keyword is generated one by one. Then, the results of all keywords are combined based on the Boolean logic of the keywords of the query. The combined result is the one to present to the user. In this query process, the chunk size being two does not show any advantage over the chunk size being one because the query model does not need the position information that chunk size being two offers. We predict that the chunking strategy will show more promising results if the phrase query is used instead of the individual keyword query. More experiments are needed on the chunking strategy for the phrase query



	5	10	15	20	25
sv1.0	10653.088	7503.895	6516.589	5214.99	5158.647
mv1.0	10775.487	7281.737	5854.168	5512.408	5075.943
sv2.0	11485.588	7634.314	6124.759	5291.894	5254.161
mv2.0	12008.834	8221.4	6900.042	5736.956	6079.07

Figure 7.1-3 Multiset vs Set option for Jaccard's Coefficient Model, 1st 7 query types and collection C2

7.3 Sampling Strategies: This section gives the results using sampling strategy for 5 generalized models (results are similar for the older models). First we give declustering timings for both sampling strategies and sample sizes.

Table 7-3 Declustering Timings (mins) for Log Sampling and for Sqrt Sampling

Number of Disks	Old Strategy	New Strategy	Old Strategy	New strategy
5	99.3474	38.1751	102.2735	41.47309
10	105.48054	50.19788	107.3559	53.63273
15	119.26482	59.81114	119.868	61.34722
20	114.07207	63.49129	115.553	62.2195
25	122.94642	67.00039	122.983	72.2027

Table 7.3-1 (7.3-2) shows the overall results for similarity models using log sampling and old (new) strategy. Table 7.3-3 (7.3-4) shows the overall results for similarity models using square root sampling and old (new) strategy.

Table 7.3-1 Sum of retrieval times (ms) of all query types and Old Log Sampling

Model	5	10	15	20	25	Sum	Rank
1sv2	24621.48	20378.3	16714.4	17535.49	15339.3	94589	1
2sv2	27092.73	21109.18	18114.08	18276.08	17997.72	102590	4
3sv2	24378.05	19688.79	18238.23	18232.15	16023.12	96560.3	2
4sv2	25236.07	19861.52	18192.34	18939.37	18074.02	100303	3
5sv2	51606.27	41496.74	56649.92	40315.92	31401.51	221470	5

Table 7.3-2 Sum of retrieval times (ms) of all query types and New Log Sampling

Model	5	10	15	20	25	Sum	Rank
1sv2	25806.75	16333.71	12068.33	11027.26	10140.04	75376.09	1
2sv2	30934.7	17214.14	16866.77	13878.25	12572.47	91466.32	3
3sv2	30934.7	17214.14	16866.77	13878.25	12572.47	91466.32	4
4sv2	25215.42	16897.63	12954.61	11655.94	10986.36	77709.96	2
5sv2	43320.3	43707.29	42137.34	43129.96	29517.52	201812.4	5

Table 7.3-3 Sum of retrieval times (ms) of all query types and Old Sqrt Sampling

Model	5	10	15	20	25	Sum	Rank
1sv2	24638.76	19469.03	15528.27	19343.88	15660.21	94640.14	1
2sv2	24005.66	22227.58	17675.14	18017.1	18549.06	100474.5	2
3sv2	28334.64	18945.68	19658.19	18058.48	19113.13	104110.1	3
4sv2	26444.09	21761.27	21812.56	17244.6	17455.93	104718.5	4
5sv2	60090.33	44959.91	48034.31	36913.93	34813.97	224812.5	5

Table 7.3-4 Sum of retrieval times (ms) of all query types and New Sqrt Sampling

Model	5	10	15	20	25	Sum	Rank
1sv2	24415.35	16644.35	12431.06	11027.26	10326.81	74844.82	1
2sv2	30839.24	19121.61	12343.11	13878.25	14874.18	91056.39	3
3sv2	30839.24	19121.61	16361.24	13878.25	14874.18	95074.52	4
4sv2	25125.07	15608.44	12839.8	11655.94	11374.86	76604.1	2
5sv2	54456.79	46017.95	50378.05	43129.96	42168.35	236151.1	5

The similarity models with sampling strategies surprisingly show worse results than Round Robin and Random Selection methods (Table 7-1) in all the cases. One of the reasons is that when the number of documents on each disk is low, Round Robin is a good strategy. Further experiments are needed with larger and more diversified document collections to confirm these negative results. Because of a lack of space, we have omitted the results for the Excite query set in Sections 7.1 through 7.3. These results agree with the results for the other query types.

8 CONCLUSIONS

In this paper, we have presented several declustering algorithms based on existing similarity measures as well as their generalizations. Experiments show that the new declustering methods are indeed more efficient than the quadratic declustering methods. The new models are also capable of handling streaming data quite efficiently. The set option is usually the winner over the multiset option. The new sampling strategies based on profiles of the documents also outperform the old sampling strategies, which do not use a profile. The results for sampling are negative, i.e., data-sensitive models with sampling perform worse than data insensitive methods. This intriguing result needs further testing.

9 REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. ACM Press/Addison Wesley 1999.

[2] S. Behl and R. M. Verma. Efficient Declustering Techniques for keyword-based Information Retrieval. Parallel and Distributed Computing and Systems 2002

[3] S. Brin, J. Davis and H. Garcia-Molina. Copy Detection mechanisms for Digital Documents. Proceedings of ACM SIGMOD Annual Conference, May 1995.

[4] A. Z. Broder. On the resemblance and Containment of Documents, On the web.

[5] A. Z. Broder, S. C. Glassman, M. S. Manasse and G. Zweig. Syntactic Clustering of the Web. SRC Technical Note 1997-015

[6] A. Tomasic and H. Garcia-Molina. Query Processing and Inverted Indices in Shared-Nothing Text Document Information Retrieval Systems. VLDB Journal, 2, 243-275(1993)

[7] B. J. Jansen, Amanda Spink, J. Bateman and T. Saracevic. Real Life Information Retrieval: A Study of User Queries on the Web. SIGIR Forum, Vol. 32. No. 1., pp.5-17

[8] B-S Jeong and E. Omiecinski. Inverted File Partitioning Schemes in Multiple Disk Systems. IEEE Transactions on Parallel and Distributed Systems. Vol.6, No. 2, February 1995

[9] B. Moon and J. H. Saltz. Scalability Analysis of Declustering Methods for Cartesian Product Files. IEEE Transactions on Software Engineering, April 1996

[10] N. Shivakumar and H. Garcia-Molina. SCAM: A Copy Detection Mechanism for Digital Documents. Proceedings of the 2nd International Conference on Theory and Practice of Digital Libraries, 1995.

[11] N. Shivakumar and H. Garcia-Molina. Building a Scalable and Accurate Copy Detection Mechanism. Proceedings of the 3rd International Conference on Theory and Practice of Digital Libraries, 1996.

[12] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. WWW7/Computer Networks and ISDN Systems, 30(1-7):107-117, 1998.

[13] Owen de Kretser, A. Moffat, T. Shimmin, and J. Zobel. Methodologies for distributed information retrieval. In International Conference on Distributed Computing Systems, pages 66-73, 1998.

[14] C. Faloutsos and P. Bhagwat. Declustering using fractals. In Int'l Conf. on Parallel and Distributed Systems, pages 18-25, 1993.

[15] David A. Grossman and Ophir Frieder. Information Retrieval – Algorithms and Applications. Kluwer Academic, 1998.

[16] David Kotz. Applications of parallel I/O. Technical Report PCS-TR98-337, Computer Science Dept., Dartmouth College, 1998.

[17] S. Kuo, M. Winslett, Y. Cho, et al. Efficient input and output for scientific simulations. In Proceedings of the 6th Workshop on Input/Output in Parallel and Distributed Systems, pages 33-44, 1999.

[18] H. Liu. Performance evaluation of declustering techniques for keyword-indexed information. MS thesis, U. of Houston, 2003.

[19] R. Muntz, J .R. Santos, and S. Berson. A parallel disk storage system for real-time multimedia applications. Int'l Journal of Intelligent Systems, 13(12):1137-74, 1998.

[20] D. Rotem, S. Seshadri, and L.M. Bernardo. Data Replication and Delay Balancing in Heterogeneous Disk Systems. Distributed Data & Structures Conference, Orlando, USA. Proceedings in Informatics 2, Carleton Scientific, 1999