Searching by Corpus with Fingerprints

Charu C. Aggarwal
IBM T. J. Watson Research Center
Hawthorne, NY 10532, USA
charu@us.ibm.com

Wangqun Lin
National University of Defense Technology
Changsha, Hunan, China
linwangqun@nudt.edu.cn

Philip S. Yu
University of Illinois at Chicago
Chicago, IL, USA
psyu@cs.uic.edu

ABSTRACT

The growing sizes of text repositories on the world wide web has created a need for efficient indexing and retrieval methods for text collections. Almost all of the text retrieval and indexing methods have been designed for the case of simple keyword search, in which a few keywords are specified, and the text is retrieved on the basis of matches to these keywords. However, in many applications there is a need for a greater *specificity* during the search, such as the use of phrases, sentences, text fragments, or even documents for the retrieval process. An even more general case is one in which *a collection of* documents is available as a query to the search process. In such cases, it is desirable to return sets of all pairwise similar documents. Such queries are referred to as *corpus to corpus* queries, and are computationally intensive because of the very large number of document pairs which need to be compared. Such cases cannot be efficiently processed by the available indexing and searching methods. Most of the currently available techniques can index the text based on only a small number of keywords or representative phrases. In this paper, we design a compressed finger print index which can support the following more general queries: **(a)** The method can process very efficient document-to-corpus search because of their efficient bit-wise operations for the search process. **(b)** We further extend the method to work for corpus-to-corpus queries, in which it is desirable to determine the most similar *pairs of* documents in two collections. We design an efficient search technique which is able to reduce the search time for large collections. The key technique used to enable this is an efficient fingerprint representation, which can be used effectively for the search process. To the best of our knowledge, this is the first work on *corpus-based search* in massive document collections.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering

General Terms

Algorithms

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*EDBT 201*2, March 26–30, 2012, Berlin, Germany.
Copyright 2012 ACM 978-1-4503-0790-1/12/03 ...$10.00

1. INTRODUCTION

The rapid growth of text information has lead to a tremendous need for efficient methods for retrieving such text documents. Most of the known methods for text indexing and retrieval are designed for keyword-based search. Typically, such queries are designed for a small number of words and phrases for retrieval purposes. Most of these techniques are based on variations of the inverted index representation [12]. While such index structures are quite effective for the case of single keyword-based queries, they are not designed for cases in which it is desired to determine responses to queries in which the target is a text document or even a corpus of documents. Some examples of relevant applications are as follows:

- In many scientific applications, the target for the search may be a publication or patent in a particular subject area. For such fine grained searches, it is often possible to have similar portions of the text at several places. In some cases, a particular author may have the tendency to use similar sentence structure across different documents, and such fine grained behavior may have an impact on the search process. This is not possible with the use of the traditional index structures.
- News articles on the same story often share large segments of the text in common. This is because the quoted portions of the text, or the text which is obtained from a professional newswire service may be almost identical. This results in some largely identical segments of text, though other portions may be different. By using a collection of documents, it is also possible to perform *duplicate or partial duplicate detection* across the text corpus.
- Professional product descriptions at online sites may often share large segments of the text which are identical. This is because descriptive content is often standardized across similar products which are produced by different sources. At the same time, there may be enough distinctive vocabulary in other portions, so as to throw off a pure bag-of-words based search process.

The problem of corpus-to-corpus search is defined as one in which we have a target corpus of documents $\mathcal{T}$ and a document corpus $\mathcal{C}$. We would like to determine the k document

pairs from $\mathcal{T}$ and $\mathcal{C}$ with the greatest similarity. This is useful in applications where it is desirable to determine similar documents across the two collections. The problem of corpus-to-corpus search is particularly challenging because of the potentially large number of computations which may be required in such a process. For example, consider a large corpus containing 10^7 documents, and the search set contains 10^4 documents. Let us also assume that each document contains an average of 10^3 words. In such a case, even a simple comparison between the two collections on the basis of the words are likely to require at least $10^7 \times 10^4 \times 10^3 = 10^{14}$ word-pair comparisons. Assuming that each comparison (including the overheads of parsing, and string comparisons) conservatively required 100 CPU cycles, it would require a 1GHz computer 10^7 seconds, which is more than 10 days of computation. For larger document collections and query sets, this can quickly become impractical. Our goal is significantly reduce this computational time without compromising on the quality of the retrieved documents. We propose the following:

- For document-to-corpus queries, we design an efficient hash-based fingerprint representation, which is friendly to the use of efficient cluster-based indexing techniques. We show that such a technique can achieve very fast online response times.
- For corpus-to-corpus queries, our indexing technique provides an efficient search process, which provides several orders of magnitude scale up in query processing times. While this is not necessarily designed to provide online query processing, a huge response time of a few minutes is much more acceptable than a response time of a few days.
- In addition, our approach also has some advantages in terms of the *quality* of the retrieved results because of the approach used for search and processing. We will discuss more on this issue later.

One important aspect of document-to-corpus search is that the *structure* of the document is very important in the retrieval process. This is not the case for keyword-based search, in which similarity is based on the membership of a few words. A long target document may have a significant amount of embedded structure in it, and it can be challenging to use this structure in the retrieval process. Some recent work [1] has shown that the use of distance information for document-to-corpus similarity search provides superior quality results to the use of well known similarity measures such as the cosine metric. *In any case, document-to-corpus similarity search is cannot be performed efficiently with traditional structures such as the inverted index because of the large number of inverted lists which need to be accessed.* The larger the document, the more inefficient the use of an inverted structure is likely to be.

In this paper, we will use a compression based approach in order to create a *hash-based compressed representation* of the distance-graphs for the document. These compressed representations essentially function as fingerprints which can be clustered and utilized for efficient query processing. The distance-graph is a novel representation for text which was proposed in [1], and has the advantage of retaining partial information about the underlying structure of the document. We further construct a compressed representation of the distance graph, which can be leveraged for effective retrieval and processing. Specifically, each document is compressed into a stream of bits which retain partial information about the underlying structure. This allows for efficient bitwise matching operations during the query processing approach. A clustering approach is used in order to further improve the quality of the index representation. We provide experimental results illustrating the effectiveness of the method.

This paper is organized as follows. In the next section, we will discuss the distance-graph representation and its use for creating an index. In section 3, we will discuss the algorithm for query processing and provide an analysis of the effectiveness of the approach. The experimental results are presented in section 4. Section 5 contains the conclusions and summary.

1.1 Related Work

The problem of text indexing has been widely studied in the literature [5, 14] because of the tremendous application to a wide variety of information retrieval applications. Text can either be indexed as strings [9] or as a bag-of-words [12, 11]. The string-based indexing method [7, 9] works well with strings in the biological domain, but is generally not very widely used for text applications, because of the semantic interpretability of words, and the variation in the word order. The most widely used method for text indexing is the bag-of-words methods in terms of the inverted index [2, 10, 12]. However, this approach is generally effective for smaller search queries, because a larger target text document requires the access of a large number of inverted lists for query processing. While match queries can be efficiently implemented over a small number of terms, a large number of lists can make the retrieval process much more challenging. Furthermore, the computation of more involved distance functions such as the cosine function becomes computationally challenging with the use of an inverted file. Furthermore, it has been shown [1] that the bag-of-words approach is also *qualitatively* not very effective for document-to-document retrieval, without the use of information about word ordering in the target documents. Some methods [4, 13, 15, 16] have proposed the mining of phrases on order to improve the quality of similarity search, though these methods are not designed as indexing schemes. Therefore, efficiency continues to be a problem.

Another well known method for text indexing are those of signature files [3, 5, 8, 6, 17]. The signature file also uses a hash-based approach for signature construction, but it uses a hash function of a *fixed range* over the *bag-of-words* approach. It has been shown in [17], that such an approach is not very effective for the case where there is tremendous variability in source and target document length, as would naturally be the case in any corpus-to-corpus retrieval system. In our fingerprint scheme, we use a 2-dimensional hashing scheme, in which the range of the hash function is document-dependent, the second dimension of the hashing is used to increase the robustness required for effective similarity computation. The conversion is applied on the distance graph in order to retain information about word order.

2. STRUCTURE-BASED BIT STRING REPRESENTATION

When the target query is a document, as opposed to a set of keywords, the relative ordering of the words are much more important for the similarity search process. Larger documents require a much more fine-grained search than a bag-of-words representation, because of the greater importance of the word-ordering. Distance graphs are a natural representation [1] which preserve a high level of information about the ordering and distance between the words in the document. In this paper, we will show how to leverage the distance-graph representation in combination with a hash-based index.

Before proceeding further, we will review the concept of a distance graph, as defined in [1]. Distance graphs can be defined to be of a variety of *orders* depending upon the level of distance information which is retained. Specifically, distance graphs of order m retain information about word pairs which are at a distance of at most m in the underlying document. We define a distance graph as follows:

DEFINITION 1 (DIST. GRAPH [1]). *A distance graph of order m for a document D drawn from a corpus $\mathcal{C}$ is defined as graph $G(\mathcal{C}, D, m) = (N(\mathcal{C}), A(D, m))$, where $N(\mathcal{C})$ is the set of nodes defined specific to the corpus $\mathcal{C}$, and $A(D, m)$ is the set of edges in the document. The sets $N(\mathcal{C})$ and $A(D, m)$ are defined as follows:*

- *The set $N(\mathcal{C})$ contains one node for each distinct word in the entire document corpus $\mathcal{C}$. Therefore, we will use the term "node i" and "word i" interchangeably to represent the index of the corresponding word in the corpus. Note that the corpus $\mathcal{C}$ may contain a large number of documents, and the index of the corresponding word (node) remains unchanged over the representation of the different documents in $\mathcal{C}$. Therefore, the set of nodes is denoted by $N(\mathcal{C})$, and is a function of the corpus $\mathcal{C}$.*
- *The set $A(D, m)$ contains a directed edge from node i to node j if the word i precedes word j by* **at most** *m positions at least once in the document. For example, for successive words, the value of m is 1.*

We note that the set $A(D, m)$ always contains an edge from each node to itself. This is because of the fact that any word precedes itself at distance 0 by default.

For the purposes of representation, it is assumed that the stop-words are removed from the text *before* distance graph construction. In other words, stop-words are not counted while computing the distances for the graph, and are also not included in the node set $N(\mathcal{C})$. This greatly reduces the number of edges in the distance graph representation. This also translates to better efficiency during processing.

We note that the order-0 representation contains only self loops with corresponding word frequencies. Therefore, this representation is quite similar to the vector-space representation. Representations of higher orders provide structural insights of different levels of complexity. An example of the distance graph representation for a well-known nursery rhyme *"Mary had a little lamb"* is illustrated in Figure 1. In this figure, we have illustrated the distance graphs of orders 0, 1 and 2 for the text fragment. The distance graph is constructed only with respect to the remaining words in the document, after the stop-words have already been pruned. The distances are then computed with respect to the pruned representation. Note that the distance graphs or order 0 contain only self loops. The number of edges in the representation will increase for distance graphs of successively higher orders. Another observation is that the frequency of the self loops in distance graphs of order 2 increases over the order-0 and order-1 representations. This is because of repetitive words such as "little" and "lamb" which occur within alternate positions of one another. Such repetitions do not change the frequencies of order-0 and order-1 distance graphs, but do affect the order-2 distance graphs. We note that distance graphs of higher orders may sometimes be richer, though this is not necessarily true for orders higher than 5 or 10. It has been shown in [1] that it is better to use distance graphs of lower orders such as 1 or 2.

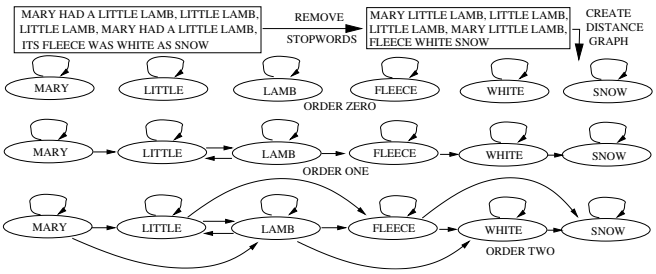


Figure 1: Illustration of Distance Graph Representation

In this paper, we use such a compression-based index on top of this distance graph representation which (a) can be efficiently used for document-to-corpus similarity search, (b) can be used efficiently for corpus-to-corpus similarity search, and (c) qualitatively more effective than a pure vector-space based similarity.

2.1 A Hash-Based Fingerprint Index

In this section, we will present a hash-based compression index for indexing the documents. We note that the number of possible distinct edges is very large is potentially the size of the square of the underlying lexicon. For a lexicon of size 10^5, the number of possible distinct edges can be 10^{10} or greater. The large domain size of distinct edges can create a challenge for designing efficient indexing methods for the distance graph representation. Hash-based methods are a natural method for compressing the edge-inclusion information, so as to ensure the ability of performing effective similarity search. Furthermore, we use a *binary representation* of the hashed values, so as to ensure the use of efficient bitwise operations during the similarity search process.

The fingerprint data structure consists of a two-dimensional array with $w \cdot h$ *binary* cells with a length of h and width of w. Since each cell is binary, it can be stored efficiently, with

only one bit. Each hash function corresponds to one of w 1-dimensional arrays with h cells each. Each such set of $w \cdot h$ cells is used to store the representation of a single distance-graph. As we will see later, the value of h is dependent on the number of edges in the distance-graph. The binary values in the cells are useful in keeping track of the presence or absence of the different edges in the distance graph with the use of a hash-based mapping.

The $w \cdot h$ cells in this fingerprint are defined with the use of w independent hash functions, each of which take the string representation of an edge as an argument, and map onto uniformly random integers in the range $[0, h-1]$. Each of the w arrays of h cells is used to map the presence or absence of edges in a particular distance graph with the use of one of the hash functions. These different ways of mapping provide more robustness to the mapping process. Thus, the (i, j)th cell corresponds to the ith hash function mapping to the value $j-1$. For each edge in the distance graph, we apply each of the w hash functions (to a string representation of the edge) in order to obtain a mapping to a number in $[0 \ldots h-1]$. When the ith hash function maps to value j, we set the bit in cell $(i, j+1)$ from the fingerprint to 1. Membership can be checked for a particular edge in the distance graph by evaluating whether all the w hash functions for that edge map onto a cell with a value of 1. In some cases, different distinct edges may be mapped onto the same cell by the hash function. This is referred to as a *collision*, and it reduces the accuracy of the representation. This is the reason that w independent hash functions are used in order to improve the robustness of the mapping process. The idea is to create an approximate *fingerprint* which provides an accurate idea of the inclusion information most of the time. As we will see later, this representation can be used to create an efficient and effective index for query processing.

In order to accommodate documents of different sizes in the collection, we use hash functions of different ranges. The documents are mapped to one of these sketch tables, depending upon the number of edges in the corresponding distance graph. For a given document, for which the distance graph contains L edges, we use a sketch table whose hash function ranges in the interval $\{0 \ldots (2^{\lceil \log(L) \rceil} - 1) \cdot (1+\alpha)\}$, where $\alpha > 0$ is a user-defined parameter larger than 0. The parameter α is referred to as the *hash-range factor*, as it affects the range of the hash function, and the accuracy which the document is represented. As we will see later, the value of α regulates the level of accuracy of the hashing process. Since the value of L can vary widely depending upon the size of the document, it is clear that we need hash functions of exponentially increasing ranges in order to accommodate documents of different sizes. In order to facilitate further discussion, we need to define the concept of the *level* of a fingerprint. A fingerprint of the qth level uses hash functions which lie in the interval $[0 \ldots (2^q - 1) \cdot (1+\alpha)]$. Each fingerprint uses w different hash functions, which are independent of one another. While these hash functions remain the same *for each document in a given level* in the entire corpus, they will naturally be different across different levels. This is because the range of the hash function is different across different levels. The jth hash function of the qth level sketch table is denoted by $g^{qj}(\cdot)$. We note that the total number of levels of fingerprints varies logarithmically in the maximum number of edges in any document. Therefore, the number of levels is quite modest in practice.

Next, we discuss how a particular document can be converted into the bit-wise format of the sketch table. Let us consider a document D, which contains L edges. We use a sketch table whose level q and corresponding hash-function range $[0, (2^q - 1) \cdot (1+\alpha)]$ is defined by setting $q = \lceil \log(L) \rceil$. For each edge between word nodes i and j, we create a new string by concatenating i, '#', and j, and then apply the hash functions directly to the newly concatenated string $i \oplus' \#' \oplus j$, where $\oplus$ denotes the concatenation operator. In addition, we explicitly store the number of distinct edges in a particular document D. The number of distinct edges in a particular document D is denoted by $\eta(D)$.

The bit string representing the distance graph is of size $2^q \cdot (1+\alpha) = 2^{\lceil \log(L) \rceil} \cdot (1+\alpha)$, and therefore contains a number of bits which is at most a $2 \cdot (1+\alpha)$ factor of the edges in the distance graph. As we will see later, the use of the bit representation facilitates efficient document-to-corpus similarity search. The bit vector for the document D and the ith hash function at the qth level is denoted by $B^q(D, i)$. We define the fingerprint representation of a document as follows:

Definition 2 (Fingerprint Representation). *The fingerprint representation of a document D with $\eta(D)$ distinct edges (in the corresponding distance graph) and level $O(D) = \lceil log(\eta(D)) \rceil$ is defined as the set of the following values:*

- *The representation contains a set of $w \cdot h$ binary cells, in the form of a 2-dimensional binary array. Each of the w 1-dimensional arrays corresponds to a different hash function, with corresponding bit vector $B^{O(D)}(D, i)$ with h bits in it. The value of i (index of the hash function) may range from 1 through w. The value of w is decided by the user. The range h of the hash function is $2^{O(D)} \cdot (1+\alpha)$, where $\alpha > 0$ is a user-defined parameter.*
- *The fingerprint representation contains the number of edges $\eta(D)$ in the underlying distance graph.*

2.2 Fingerprint-based Similarity

The most naive method for retrieving the document is to directly use a sequential-scan based similarity search. Our first step is to define a *fingerprint-based similarity function*, which derives its motivation from the cosine function for similarity. The bit-string representations lend themselves to similarity computations because of the efficiency of bitwise operations on many platforms. We define the similarity between two documents as the normalized dot product between the corresponding representations. For a document D with level $O(D)$, let the vector $B^{O(D)}(D, i)$ be the ith bit vector which represents the presence of absence of edges in $A(D, i)$. We note that the value of i may range from 1 through w. Before defining the normalized dot product, we first define the unnormalized dot product between the two fingerprint vectors. The fingerprint dot product $FDot(D_1, D_2)$ is defined *only* between documents of the same order.

DEFINITION 3. *The fingerprint dot product between the documents D_1 and D_2 of the same order q is defined as the* **average** *of all dot products between $B^q(D_1, i)$ and $B^q(D_2, i)$ over different values of the index i of the hash function.*

$$FDot(D_1, D_2) = \frac{\sum_{i=1}^{w} B^q(D_1, i) \cdot B^q(D_2, i)}{w} \quad (1)$$

The dot product between each pair $B^q(D_1, i)$ and $B^q(D_2, i)$ can be computed quite simply as a dot product between two bit string vectors of length h each.

We note that since documents are maintained in compressed form, the similarity computation cannot be performed exactly. The fingerprint representation however provides a way to efficiently *approximate* the dot-product between the binary representation of the underlying distance graphs. We further use normalization factors in order to adjust for the fact that different documents may contain a different number of edges in the distance graph representations. These factors are essentially the norms of the underlying vectors. The normalized fingerprint similarity between documents D_1 and D_2 is defined as follows:

DEFINITION 4 (NORMALIZED FIN. SIM.). *The normalized fingerprint similarity between D_1 and D_2 is obtained by dividing un-normalized value by $\sqrt{\eta(D_1) \cdot \eta(D_2)}$. In other words, if $H(D_1, D_2)$ represent the normalized fingerprint dot product, we have:*

$$H(D_1, D_2) = \frac{FDot(D_1, D_2)}{\sqrt{\eta(D_1) \cdot \eta(D_2)}} \quad (2)$$

We note that the naive query processing technique requires only bit vector computations, which are typically very efficient on most platforms. For example, the dot product between the bit vectors $B^(D_1, i)$ and $B^q(D_2, i)$ can be computed using the "AND" operations. The main factor to keep in mind is that we have bit vector representations of *multiple levels*, which need to be compared against a given target document. Since the dot product is defined only between documents of the same level, the target document needs to be converted into different levels in order to enable similarity computation across the whole collection. We note that the number of levels is not very large, and is logarithmically related to the size of the largest document. Therefore, if L_{max} be the number of edges in the largest document, then the number of levels is given by $\log_2(L_{max})$. The target document is transformed into representations of these different levels. The representation of a given level of the target document is used in order to perform retrieval with documents of each level. In other words, when we transform the target document to bit strings of level q, then we use only the documents of level q for retrieval. The similarity computation can be performed with the documents of the different levels, and then the closest set of documents over different levels can be combined together.

2.3 Faster Document-to-Corpus Queries

The last section provided a naive approach for performing faster document-to-corpus queries. In this section, we will design a more efficient index which can use the hash-based fingerprints for a more efficient solution. The overall approach for performing faster document-to-corpus queries uses a branch-and-bound pruning methodology, which segments the document collections into different clusters, and then prunes some of these clusters in an ordered search process. Specifically, we use two parameters γ and μ in order to regulate the clustering process. The parameter γ represents the *normalized radius* of the cluster, whereas the parameter μ represents the *critical mass* of a cluster. Specifically, the fingerprint representation $\mathcal{C}^f$ of corpus $\mathcal{C}$ is partitioned into groups $\mathcal{G}_1 \ldots \mathcal{G}_r$, along with a special outlier group $\mathcal{O}$ which satisfies the following properties:

- There exists a fingerprint representative $F_i \in \mathcal{G}_i$, such that the fingerprint representation of each graph in $\mathcal{G}_i$ is at a maximum hamming distance of at most $\gamma \cdot h \cdot w$ to the representative point of its cluster. We call the graph F_i the *medoid representative* of group $\mathcal{G}_i$. The parameter γ is referred to as the *hamming radius* of the group.
- Each group $\mathcal{G}_i$ contains only fingerprints of a particular level. Since F_i contains $h \cdot w$ bits, it follows that the parameter γ is a normalized radius which can be compared effectively over fingerprints of different levels and different number of hash functions.
- Each group $\mathcal{G}_i$ contains at least μ fingerprints. This is the *critical mass requirement* for each group, and is essential in order to ensure an efficient pruning process.

Once these groups have been created, they can be used in conjunction with a simple branch-and-bound technique in level to perform the query processing.

Before discussing the process of construction of these groups in more detail, we will discuss how they can be leveraged for the purpose of query processing. Let us consider a corpus $\mathcal{C}$ which has already been partitioned into groups $\mathcal{G}_1 \ldots \mathcal{G}_r$,

Algorithm *FindClosestDocument*(Target: T,
  Partitioned Groups: $\mathcal{G}_1 \ldots \mathcal{G}_r, \mathcal{O}$, NumOfNeighbors: k);
begin
  $PessimisticBound = 0$;
  $BestSoFar = \{\}$;
  Scan entire outlier set $\mathcal{O}$ and update
    $PessimisticBound$ and $BestSoFar$;
  Compute $Optimistic(i)$ for each $\mathcal{G}_i$;
  for each $i \in \{1 \ldots r\}$ in order of
   reducing value of dot product of fingerprint
   representation of T and F_i **do**
  begin
    if $Optimistic(i) \geq PessimisticBound$
    begin
      Compute similarity of each fingerprint in
       $\mathcal{G}_i$ to the conversion of target document T
       to a fingerprint of the same level as $\mathcal{G}_i$
       and update the values of $BestSoFar$ and
       $PessimisticBound$, if applicable
    end
  end
  return($BestSoFar$);
end end

Figure 2: Query Processing with Fingerprints

along with the corresponding medoid representative F_i. It is assumed that the value of r is much less than the total number of documents in the collection. This is also ensured by the fact that we impose a minimum critical mass μ on the collection. Let us also assume that the fingerprint representation of the target document T is denoted by T^f. The first step is to estimate the cosine distance between the target fingerprint T^f, and each of the different medoid representatives $F_1 \ldots F_r$. As discussed earlier, this cosine distance is computed by first estimating the dot product by using the minimum of the dot product values over the w different bit string representations. This is averaged over the w different representations. This dot product is then normalized using the norms of the two documents. This is possible since the norms are stored explicitly with the fingerprints.

The overall approach is to compute the dot product of the target document T to each of the fingerprint representatives $F_1 \ldots F_r$ and sort them in order of decreasing value of this dot product. The clusters associated with each of these groups are then processed in order of reducing similarity. In order to apply the branch and bound method, we maintain a *global pessimistic bound on the similarity value*, and a *local optimistic bound which is specific to a particular group*. The cluster associated with a particular group may be ignored for the purposes of further exploration if the local optimistic bound is no better than the global pessimistic bound. It remains to explain how the local optimistic bound and global pessimistic bound are maintained for each group.

The global pessimistic bound for each group is simply the closest fingerprint found to the target fingerprint so far. In the event that we are trying the find the k best neighbors (rather than the nearest neighbor only), this pessimistic bound is set to the kth best value found so far. This global pessimistic bound continues to improve over time, as more fingerprints are compared to the target. Furthermore, the different groups are processed in order of decreasing optimistic bound. Therefore, as time passes, the chances of the pruning condition being satisfied will increase monotonically.

The overall algorithm is illustrated by the algorithm *FindClosestDocument* in Figure 2. The best set of documents found so far is simply the set of k documents with the highest fingerprint-based similarity. This is stored in the set $BestSoFar$ in Figure 2. The variable $PessimisticBound$ denotes the similarity of the kth best document in this set. We first scan the entire outlier set $\mathcal{O}$ and compute the k best neighbors from this set. We use these neighbors to update $BestSoFar$ and $PessimisticBound$. Next, we compute the optimistic bound $Optimistic(i)$ for each group, and process them in reducing value of $Optimistic(i)$. A group is scanned only if $Optimistic(i)$ is at least equal to $PessimisticBound$. We note that the value of $Optimistic(i)$ continually reduces throughout the process (because of the sort order of processing), and the value of $PessimisticBound$ continues to increase as more and more records are encountered. Therefore, as soon as we reach the first group for which $Optimistic(i)$ is less than $PessimisticBound$, we are guaranteed that this will continue to be the case for all remaining groups, and we can terminate scanning any more groups for the nearest neighbors. At this point, we can report the best set of documents found so far as the k nearest neighbors.

2.3.1 Computation of the Optimistic Bound

A key subroutine required for the use of the above method is the computation of the optimistic bound of target document T to the group $\mathcal{G}_i$. It remains to explain, how this optimistic bound is computed. Let us consider the group $\mathcal{G}_i$, which has the representative fingerprint denoted by F_i. Let us assume that the norm of the target document is $\eta(T)$. Let us assume that the *level* of all documents in $\mathcal{G}_i$ is q. Then, the *smallest* number of edges in the distance graph representation of any document in this group is 2^{q-1}. Then, it can be shown that an optimistic bound of the fingerprint distance of target document T to any document in $\mathcal{G}_i$ is given by $\frac{\sqrt{\eta(F_i)} \cdot H(T,F_i)}{2^{(q-1)/2}} + \frac{\gamma}{2^{(q-1)/2} \cdot \sqrt{\eta(T)}}$.

LEMMA 1. *Let $\mathcal{G}_i$ be any group of documents with level q and representative medoid fingerprint F_i and hamming radius γ. Then, any documents in $R_i \in \mathcal{G}_i$ has similarity $H(T, R_i)$ to T, which satisfies the following upper bound:*

$$H(T, R_i) \leq \frac{\sqrt{\eta(F_i)} \cdot H(T, F_i)}{2^{(q-1)/2}} + \frac{\gamma \cdot h}{2^{(q-1)/2} \cdot \sqrt{\eta(T)}} \quad (3)$$

Proof Sketch: Let R_i be any document in the group $\mathcal{G}_i$. Then, the value of normalized dot product $H(T, R_i)$ is defined as follows:

$$H(T, R_i) = \frac{FDot(T, R_i)}{\sqrt{\eta(T)} \cdot \sqrt{\eta(R_i)}} \quad (4)$$

In order to compute an optimistic (upper) bound on the expression above, we need to compute an upper bound on the numerator and a lower bound on the denominator. Since the hamming distance between R_i and F_i is no larger than $\gamma \cdot h \cdot w$, it follows that:

$$FDot(T, R_i) \leq FDot(T, F_i) + \gamma \cdot h \quad (5)$$

Furthermore, since the smallest number of edges in the distance graph for an document in this group (of level q) is given by 2^{q-1}, it follows that:

$$\eta(R_i) \geq 2^{(q-1)} \quad (6)$$

By substituting the upper bound of Equation 5, and the lower bound of Equation 6 in the respective numerator and denominator of Equation 4, it is possible to convert the expression of Equation 4 into an inequality which provides an upper bound on $H(T, R_i)$. After performing the substitution and subsequent algebraic simplification, it is easy to show the result. □

The query processing method discussed above assumes that the documents have already been partitioned into groups $\mathcal{G}_1 \ldots \mathcal{G}_r$ with the appropriate hamming thresholds. It remains to show how this is done.

2.4 Constructing the Index Partition

Each partition $\mathcal{G}_i$ of the index needs to contain fingerprints which are of the same level. This step is executed as a preprocessing step in order to create an index which is subsequently used for repeated query processing. The first step

is to convert the corpus $\mathcal{C}$ into the fingerprint representation $\mathcal{C}^f$. Let L_{max} be the largest number of edges in any distance graph representation of the corpus $\mathcal{C}$. We note that this fingerprint representation contains documents of different levels $\mathcal{E}_1 \ldots \mathcal{E}_s$, where $\mathcal{E}_i$ is the set of documents in corpus $\mathcal{C}$ of level i. The value of s is $\log_2(L_{max})$.

For the documents $\mathcal{E}_i$ of the ith level, we use a two phase processing approach for partitioning into groups with minimum mass μ and (normalized) hamming radius γ. In the first phase, we retain a set of representatives medoids $\mathcal{Q}$, which are initialized to a single fingerprint from $\mathcal{E}_i$. Associated with the jth medoid in $\mathcal{Q}$, we have a set of fingerprints $\mathcal{Z}_j$. The documents are scanned one by one, and it is checked whether the closest medoid in $\mathcal{Q}$ to the currently being scanned fingerprint D is at a normalized hamming distance of at most γ across the w different hash functions. If this is indeed the case, and the closest medoid is the jth medoid, then we add the current fingerprint D to the set $\mathcal{Z}_j$. Otherwise, we add the fingerprint D to the current medoid set $\mathcal{Q}$, which increases by 1. At the end of the first phase, we have a representative medoid set $\mathcal{Q}$ for $\mathcal{E}_i$, though many of the corresponding sets $\mathcal{Z}_j$ to each medoid in $\mathcal{Q}$ may not satisfy the critical mass requirement μ. Therefore, each set $\mathcal{Z}_j$ which has less than μ fingerprints needs to either be redistributed to a different *representative medoid*, or it needs to be added to the outlier set. Therefore, at the beginning of the second phase, we prune all medoids from $\mathcal{Q}$ which have less than μ fingerprints assigned to them. We create a new subset of documents which were assigned to these medoids. This subset of documents are scanned again one by one, and it is checked whether the corresponding hamming distance to the closest (remaining) medoid in pruned set $\mathcal{Q}$ is less than γ. If this is the case, then this document is re-assigned to the corresponding medoid. Otherwise, it is added to the outlier set $\mathcal{O}$. The set of medoids and their assigned documents forms a valid partition of fingerprints of level $\mathcal{E}_i$, which satisfies the hamming radius and critical mass requirements. This process is repeated for documents of each level $\mathcal{E}_i$, and the corresponding set of groups and the outlier set is correspondingly augmented during this process.

2.5 Extension to Corpus-to-Corpus Similarity Queries

The method can also be extended to the case of corpus-to-corpus similarity queries. In this case, we have a target corpus $\mathcal{T}$, and we wish to determine all documents pairs from $(\mathcal{T}, \mathcal{C})$, which have the largest similarity. The main difference in this case is the way in which the most similar objects are tracked, and the optimistic and pessimistic bounds are computed. The value of $Optimistic(i)$ is now computed by performing the same optimistic bound computation between F_i and each $T \in \mathcal{T}$, and then picking the particular value of $T \in \mathcal{T}$ for which the optimistic bound is as large as possible. As in the previous case, we maintain the top k most similar candidates, except that in this case, we maintain the target-document pairs. When a particular partition $\mathcal{G}_i$ is not pruned, we compute the similarity between each $T \in \mathcal{T}$ and each document of the partition, and dynamically maintain the k best candidates. The similarity between the kth best pair is maintained as the pessimistic bound.

2.6 Speeding up the Pruning with Approximation

It is possible to speed up the pruning process further, by allowing some approximation in the determination of the k nearest neighbors. We define an ϵ-approximate set of k nearest neighbors as follows:

Definition 5 (ϵ-approximate nearest neighbors). *Any set $\mathcal{S}$ of k-nearest neighbors to the target T is said to be ϵ-approximate, if the similarity of each element in $\mathcal{S}$ to T is at least within ϵ of the similarity of the kth nearest neighbor to T.*

We note that this relaxation in the definition of the k nearest neighbors can greatly speed up the pruning process. In order to determine the ϵ-approximate k-nearest neighbors, the only difference in the algorithm is to use a relaxation on the pruning condition. In this case, we prune a group $\mathcal{G}_i$, only if $Optimistic(i)$ is no larger than $PessimisticBound + \epsilon$. This relaxation of the pruning condition greatly speeds up the search process. Furthermore, we will see that this relaxation does not affect the practical quality of the solution obtained.

3. EXPERIMENTAL RESULTS

In this section, we will present the effectiveness and efficiency results for our method. We used a variety of real data sets in order to test the effectiveness of our approach. Our goal is to show that the approach is able to achieve significantly greater efficiency than the baseline which uses the inverted index. In addition, we will show that the similarity function which is used by the fingerprint method also continues to retain its effectiveness. All experiments were conducted on a Lenovo X61 running Windows 7 Ultimate with an Intel Core2 Duo T8300 Pentium 4 processor with a speed of 2.4 GHz, 2 GB of RAM, and a 160 GB hard drive. We implemented our algorithms in Java using jdk 1.6 and a virtual memory of 1GB.

3.1 Baseline Method

The baseline method for our scheme was the inverted representation for indexing. In this context, we used the standard cosine measure for similarity based on the tf-idf representation of the text documents. The inverted representation is described in [12], and was implemented as follows. For each word in the lexicon, we created an inverted list with the document identifiers containing the word. Each of these document identifiers was attached to a normalized frequency (of the word in the document), and the corresponding L_2-norm of the (normalized) frequencies. The L_2-norm of the frequencies is required in order to compute the normalized value of the cosine similarity. For each target document, we determined the corresponding inverted lists, and maintained a set of floating point counters for each document identifier occurring in any of these inverted lists. These counters were used in conjunction with the method discussed in [12] for computing the closest documents to the target.

3.2 Evaluation Measures

Since the nature of the similarity function changes with the use of the fingerprint method, it is critical that the corresponding quality of the distances continue to be maintained.

Therefore, we need to test not only for efficiency, but also for effectiveness. One way of testing the quality is to use an *external criterion* in terms of class labels. For a given target corpus query set, we ranked all the source-target document pairs in order of reducing similarity, and picked the top 10% of the pairs based on different similarity measures. We computed the fraction of times that these pairs matched in class label value. We refer to this as the *label match purity*, and express this as a fraction in the range $[0, 1]$.

We also tested the efficiency of the query processing time for both schemes. For the case of the standard cosine similarity, the inverted index was used for query processing. We also tested the index construction time for both the fingerprint and inverted representations.

3.3 Data Sets

The data sets used were commonly used for evaluation of automatic text categorization techniques. All words in these data sets were stemmed and the stop-words were removed. Since these data sets were derived from classification applications, each of them had a clearly demarcated training and test data set. In each case, we sampled the query set from the test collection and used the training data as the corpus to be queried:

Reuters-21578: The *Reuters-21578* data set contains documents from *Reuters* newswire service. We downloaded the "R8" version[1] of the Reuters-21578 data set which contains the top 8 most frequent classes and includes 2189 test documents and 5485 training documents.

20 Newsgroups Data Set: The *20 Newsgroups data set* is a collection of approximately 20,000 newsgroup documents, which are partitioned (nearly) evenly across 20 different cybergroups. We chose the "bydate" version[2] of 20 Newsgroups which contains 7528 test documents and 11293 training documents.

CADE Data Set: The web pages for this data set[3] were extracted from the CAD Web Directory, which are Brazilian web pages classified by human experts. These data set contains 13661 test documents and 27322 training documents.

3.4 Effectiveness Results

First, we will present the effectiveness results of the fingerprint representation in terms of the quality of the similarity search. This is necessary, because the use of the fingerprint representation results in a change in the similarity function. Unless otherwise mentioned, we set the normalized hash range α to 3, the number of hash functions w to 1, the order of the distance graph m to 1, μ to 10, and the normalized radius for building the fingerprints to 1. In each case, we used a query-corpus size of 1000. The purity results for different data sets and variations of the parameters are illustrated in Figures 3(a) to (g). We will now describe the detailed results in each of the figures below.

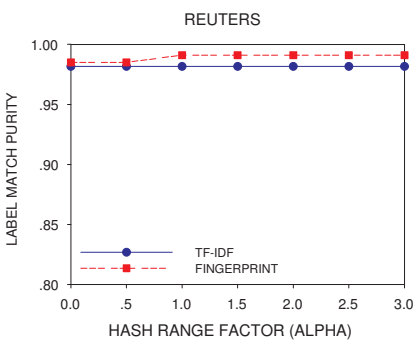
In Figures 3(a) to (c), we have illustrated the variation in label purity with increasing value of the hash range factor α for the *Reuters*, *20 Newsgroup*, and *Cade* data sets respectively. The hash range factor is illustrated on the X-axis, and the label purity is illustrated on the Y-axis for each figure (and corresponding data set). The results for the standard tf-idf based similarity are also illustrated in the same figure. Since the cosine similarity does not use the parameter α, the value for the traditional similarity measure is a horizontal line in all cases. We note that the fingerprint scheme performed better for increasing values of α, because this results in a larger hash range, and therefore fewer collisions. We note that for values of α larger than 1, the fingerprint scheme always has a higher effectiveness as compared to the traditional similarity scheme.

In Figure 3(d), we have illustrated the variation in the effectiveness with different normalized radius rates. All schemes are illustrated on the same figure. We note that the radius rate controls the granularity of the clustering process. We have also illustrated the effectiveness of the tf-idf scheme in the same figure, but this effectiveness is always a horizontal line, since the radius parameter is specific to our scheme. The radius parameter is illustrated on the X-axis, whereas the cluster purity for all schemes and all data sets are illustrated on the Y-axis. All other parameters are set to their default values mentioned above. It is evident that the fingerprint scheme is insensitive to the value of the radius. This is quite encouraging, because it implies that the scheme can be used effectively with a wide range of parameters. In each case, the fingerprint scheme performs at least slightly better than the tf-idf method, though the difference is small at this choice of parameters for two of the data sets (*Reuters* and *20 Newsgroup*). In the case of the *Cade* data set, however, the difference is quite significant. These results also show that the scheme is quite robust to different choices of the radius.

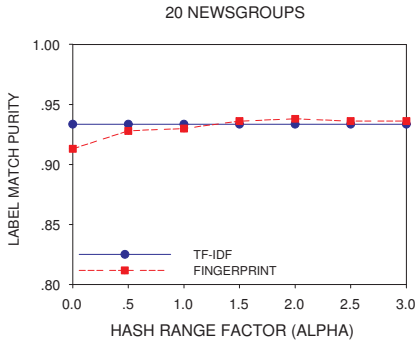
We also note that the choice of ϵ decides the level of approximation for the scheme. An increase value of ϵ facilitates a greater level of pruning (and therefore better efficiency) at the expense of quality. Therefore, it was interesting to test the level of quality degradation over different choice of the parameter ϵ. The results are illustrated in Figure 3(e). All parameters were set to the default values mentioned above. The value of ϵ is illustrated on the X-axis, whereas the label match purity is illustrated on the Y-axis. It is evident that for values of $\epsilon < 0.25$, the fingerprint scheme maintained its entire quality. However, if the value of ϵ was chosen to be too large, the quality of the solutions degraded. We will also show in the efficiency section, that the use of larger values of ϵ also improved the underlying query processing times drastically. Savings were also achieved for values of ϵ which were less than 0.25.

We note that multiple hash functions were used in order to improve the robustness of the scheme. In Figure 3(f), we have tested the effectiveness of the fingerprint scheme for different number of hash functions. This is denoted by the parameter w. The results are illustrated for all three data sets as a bar chart for different number of hash functions. We have also included a bar for the effectiveness of the baseline tf-idf scheme. One of our interesting observations was that the use of more than one hash function did not improve the accuracy much in most cases (at least in the average case). Therefore, it was possible to use an efficient scheme using only one hash function, without losing too

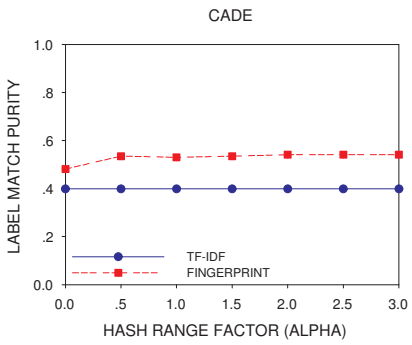
[1]http://web.ist.utl.pt/ acardoso/datasets/
[2]http://web.ist.utl.pt/ acardoso/datasets/
[3]http://web.ist.utl.pt/ acardoso/datasets/



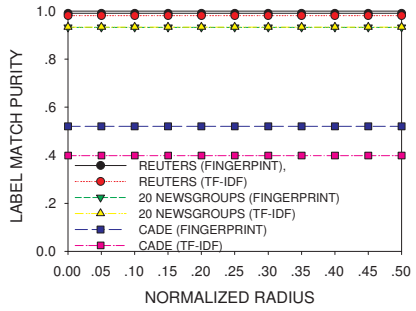
(a) Reuters (α-variation)



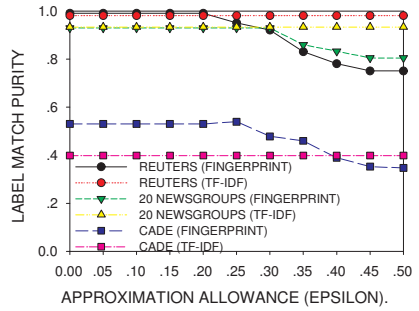
(b) 20 Newsgroup (α-variation)



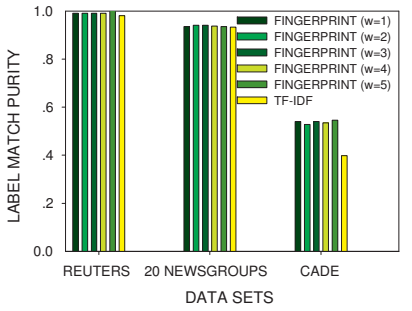
(c) Cade (α-variation)



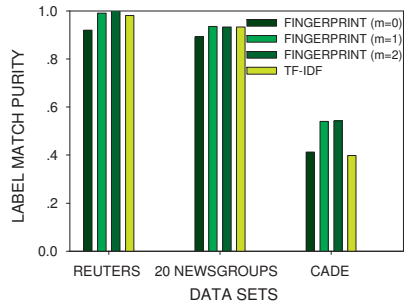
(d) All (Radius Variation)



(e) All (ϵ-variation)

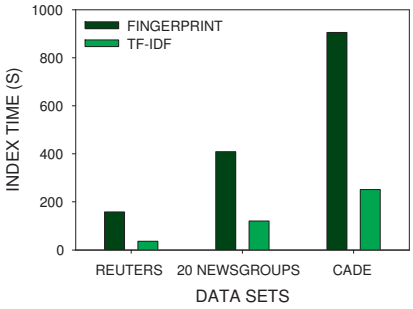


(f) All (Number of Hash Functions w-variation)

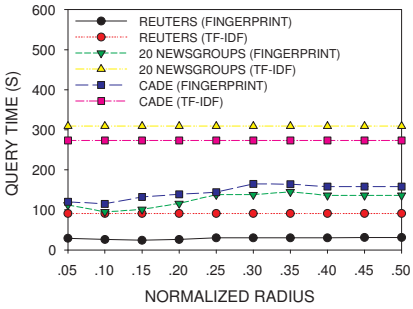


(g) All (Distance Graph Order m-variation)

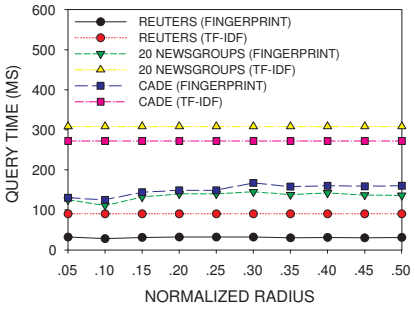
Figure 3: Effectiveness results



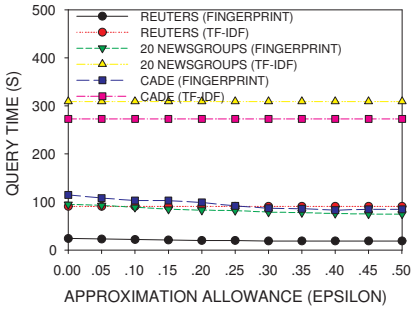
(a) Indexing Time



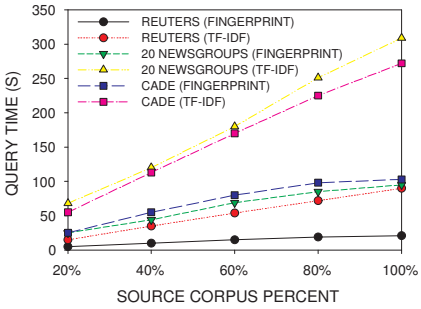
(b) Radius Scalability (CC Query)



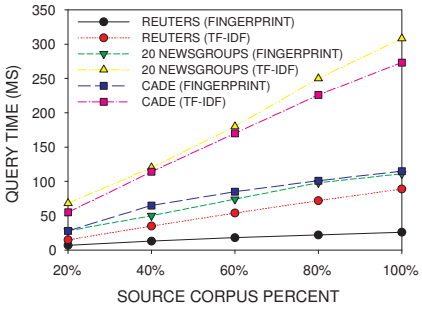
(c) Radius Scalability (DC Query)



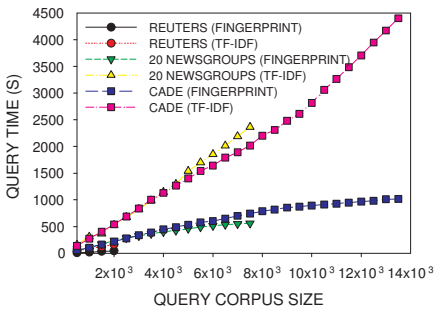
(d) ϵ-Scalability (CC Query)



(e) Corpus Size Scalability (CC Query)



(f) Corpus Size Scalability (DC Query)



(g) Query Size Scalability (CC Query)

Figure 4: Efficiency results

much effectiveness. We also tested the scheme for distance graphs of different orders, which is denoted by the parameter m. The results are illustrated in Figure 3(g) for all the three data sets. It is evident that in each case, the use of distance graphs of order 1, provided optimum results in all cases for the different data sets. This is a useful observation, since distance graphs are quite sparse, and it is possible to implement the algorithms quite efficiently for distance graphs of this order.

One summary observation from the different results presented is that the effectiveness of the fingerprint representation is either competitive to or superior to the tf-idf method for all data sets. Furthermore it is quite robust to the variation in different parameter values, and it works well for low values of the distance graph order and number of hash function. The latter observation bodes well for its efficiency. In fact, we will show that the fingerprint method also has tremendous efficiency advantages over the tf-idf scheme. On an overall basis, this would suggest that the fingerprint scheme has a clear advantage over the baseline both in terms of effectiveness and efficiency. In the next section, we present detailed efficiency results.

3.5 Efficiency Results

The fingerprint scheme has a pre-processing stage which is a one-time cost required for creation of the index. The tf-idf scheme also has a one-time cost required for index creating. The running times for index creation of both schemes are illustrated in Figure 4(a). It is evident that the fingerprint scheme requires more time for index construction because it is more complex than the simple inverted index, and it requires a clustering stage for partitioning of the index. However, we note that this is simply the pre-processing time, which is a *one-time cost*, and the query-processing times are much more critical in terms of usability of the scheme. Both the schemes require a few minutes for index construction, and this is quite acceptable for a one-time cost. As we will see below, the fingerprint scheme provides significant savings in terms of query processing time, the importance of which outweighs any overhead for one-time pre-processing.

For the case of query efficiency, we tested both the *document-to-corpus* and the *corpus-to-corpus* queries separately, because (unlike effectiveness), these two kinds of queries behave in a fundamentally different way. As in the previous case, we tested both the fingerprint and the tf-idf scheme. Unless otherwise mentioned, we set the normalized hash range α to 1, the number of hash functions w to 1, the order of the distance graph m to 1, μ to 10, and the normalized radius for building the fingerprints to 1. In each case, we used a default query corpus size of 1000.

In Figure 4(b), we have illustrated the efficiency of the corpus-to-corpus queries (CC Queries) with increasing value of the normalized radius for all the three data sets. The results for the tf-idf method are also presented as a horizontal line, as the results do not depend upon the normalized radius. The normalized radius is illustrated on the X-axis, and the query time is illustrated on the Y-axis. It is evident that the fingerprint method was always significantly more efficient than the tf-idf method, and the query times increased slightly with increasing radius. We have also presented the results for the document-to-corpus queries (DC queries) in Figure 4(c). The results are quite similar to the case of the corpus-to-corpus queries.

The parameter ϵ is used in order to control the level of approximation of the scheme. In Figure 4(d), we have illustrated the efficiency variation with the parameter ϵ for all three data sets. The parameter ϵ is illustrated on the X-axis, whereas the query time is illustrated on the Y-axis. It is evident that the efficiency increasing with increasing value of ϵ. This is because a larger fraction of the data is pruned with increasing value of the parameter ϵ. An important point to remember from our earlier effectiveness results in Figure 3(e) is that the quality of the results maintain their robustness over a wide range of values for the parameter ϵ. Therefore, the greater efficiency can be achieved at practically no loss in accuracy.

We also tested the scalability of the method with increasing text corpus size and query-corpus size. The scalability results with increasing text corpus size for the corpus-to-corpus and document-to-corpus queries are illustrated in Figures 3(e) and (f) respectively. While the tf-idf method scales linearly with increasing text collection size, the fingerprint method scales *sublinearly* with text collection size. This is because the efficiency of branch-and-bound increases with increasing database size. For the same fraction of database access, a larger collection allows for better pruning because of superior pessimistic bounds. In each case, the fingerprint method is much more efficient than the tf-idf method.

We also tested the scalability of the corpus-to-corpus queries with increasing query corpus size. The results for the corpus-to-corpus and document-to-corpus queries are illustrated in Figures 3(g))respectively. The tf-idf method needs to iteratively go through all the query documents, and its scalability is therefore linear with query size. On the other hand, the fingerprint method has the advantage of better pruning with increasing query corpus size. Therefore, it shows sublinear scalability with increasing query corpus size. Furthermore, it continues to be significantly more efficient than the baseline tf-idf method in all cases. This suggests that the fingerprint scheme is more efficient and scalable with increasing query- and database sizes in a wide variety of scenarios.

4. CONCLUSIONS AND SUMMARY

In this paper, we presented an efficient fingerprint-based index which provides the first available technique for resolving corpus-to-corpus queries. Such queries have become increasingly important in real scenarios because of search scenarios in which it may be desirable to determine documents which could be similar to any member from a particular *query corpus*. We designed an efficient fingerprint-based scheme in conjunction with a branch-and-bound method, which provides better accuracy than the tf-idf method in terms of quality, is significantly more efficient, and also shows *sublinear scalability* with increasing corpus and query sizes. The sublinear scalability also suggests that the approach is particularly useful for very large databases and query sets, which is the most difficult case in practical scenarios.

Acknowledgements

Research of the first author was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

The second author is supported by National Natural Science Foundation of China through grants 60933005, 60873204 and the National High-Tech Program through grant 2010AA012505.

The third author is supported by NSF through grants IIS-0905215, DBI-0960443, CNS-1115234, IIS-0914934, OISE-1129076, OIA-0963278, and Google Mobile 2014 Program.

5. REFERENCES

[1] C. Aggarwal and P. Zhao. Graphical models for text: a new paradigm for text representation and processing. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 899–900, New York, NY, USA, 2010. ACM.

[2] A. Blumer, J. Blumer, D. Haussler, R. McConnell, and A. Ehrenfeucht. Complete inverted files for efficient text retrieval and analysis. *J. ACM*, 34:578–595, July 1987.

[3] U. Deppisch. S-tree: a dynamic balanced signature index for office retrieval. In *Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '86, pages 77–87, New York, NY, USA, 1986. ACM.

[4] J. Fagan. Automatic phrase indexing for document retrieval. In *Proceedings of the 10th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '87, pages 91–101, New York, NY, USA, 1987. ACM.

[5] C. Faloutsos. Access methods for text. *ACM Comput. Surv.*, 17:49–74, March 1985.

[6] C. Faloutsos and S. Christodoulakis. Signature files: an access method for documents and its analytical performance evaluation. *ACM Trans. Inf. Syst.*, 2:267–288, October 1984.

[7] R. Grossi and J. S. Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching (extended abstract). In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 397–406, New York, NY, USA, 2000. ACM.

[8] C. J. Guarin. Access by content of documents in an office information system. In *Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '88, pages 629–644, New York, NY, USA, 1988. ACM.

[9] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge University Press, New York, NY, USA, 1997.

[10] A. Moffat and J. Zobel. Self-indexing inverted files for fast text retrieval. *ACM Trans. Inf. Syst.*, 14:349–379, October 1996.

[11] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24:513–523, August 1988.

[12] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[13] P. D. Turney. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2:303–336, May 2000.

[14] I. H. Witten, T. C. Bell, and A. Moffat. *Managing Gigabytes: Compressing and Indexing Documents and Images*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[15] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, C. G. Nevill-manning, and G. Inc. Kea: Practical automatic keyphrase extraction. In *In Proceedings of the 4th ACM conference on Digital Libraries*, pages 254–255, 1998.

[16] Y. Yang, N. Bansal, W. Dakka, P. Ipeirotis, N. Koudas, and D. Papadias. Query by document. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 34–43, New York, NY, USA, 2009. ACM.

[17] J. Zobel, A. Moffat, and K. Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Trans. Database Syst.*, 23:453–490, December 1998.