2012 IEEE 28th International Conference on Data Engineering

Horizontal Reduction: Instance-Level Dimensionality Reduction for Similarity Search in Large Document Databases

Min Soo Kim #1, Kyu-Young Whang #2, Yang-Sae Moon *3

Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST)
373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Korea
{1 minskim, 2 kywhang}@mozart.kaist.ac.kr

* Department of Computer Science, Kangwon National University
192-1 Hyoja2-dong, Chuncheon, Kangwon 200-701, Korea
3 ysmoon@kangwon.ac.kr

***Abstract*—Dimensionality reduction is essential in text mining since the dimensionality of text documents could easily reach several tens of thousands. Most recent efforts on dimensionality reduction, however, are not adequate to large document databases due to lack of scalability. We hence propose a new type of simple but effective dimensionality reduction, called *horizontal (dimensionality) reduction*, for large document databases. Horizontal reduction converts each text document to a few bitmap vectors and provides tight lower bounds of inter-document distances using those bitmap vectors. Bitmap representation is very simple and extremely fast, and its instance-based nature makes it suitable for large and dynamic document databases. Using the proposed horizontal reduction, we develop an efficient k-nearest neighbor (k-NN) search algorithm for text mining such as classification and clustering, and we formally prove its correctness. The proposed algorithm decreases I/O and CPU overheads simultaneously since horizontal reduction (1) reduces the number of accesses to documents significantly by exploiting the bitmap-based lower bounds in filtering dissimilar documents at an early stage, and accordingly, (2) decreases the number of CPU-intensive computations for obtaining a real distance between high-dimensional document vectors. Extensive experimental results show that horizontal reduction improves the performance of the reduction (preprocessing) process by one to two orders of magnitude compared with existing reduction techniques, and our k-NN search algorithm significantly outperforms the existing ones by one to three orders of magnitude.**

I. Introduction

Rapid advances in computer and network technologies have lead to explosive growth of unstructured or semi-structured text documents such as Web pages, e-mails, news articles, and research papers. Such a vast amount of text documents has drawn much attention to text mining on large text databases [1]. Text mining refers to the process of discovering new and meaningful knowledge or information from large document sets [2], and its representative methods are text classification [3], [4] and text clustering [5], [6]. Text documents are usually represented as vectors, and their similarity measures are frequently computed in the text mining process. Computing the similarity measures, however, is very inefficient because document vectors* are of very high dimensionality that could range from several thousands to several tens of thousands [7].

To overcome the high dimensionality problem of text documents, there have been many efforts on dimensionality reduction that projects high-dimensional vectors onto lower-dimensional subspaces [5]. For example, Latent Semantic Indexing (LSI) [8], a representative reduction technique, considers the semantics among terms in the document set and provides high quality similarity search results. Document Frequency (DF) [9], a simpler reduction technique, simply selects only a few most frequent terms from the document set. These existing techniques, however, have various disadvantages: for example, LSI is too complex to be used for large document sets [10]; and DF has a difficulty in controlling the quality of dimensionality reduction.

In this paper we propose a new dimensionality reduction technique that efficiently works for large document sets. Using the reduction technique, we also present an efficient algorithm for similar document search. To this end, we first present a novel concept of *vertical* and *horizontal (dimensionality) reductions* that is a new point of view in classifying dimensionality reduction techniques. The former is the schema-level reduction; the latter is the instance-level reduction. We then classify existing reduction techniques into the horizontal and vertical reductions. We here note that most existing techniques are classified as vertical reductions, and none of them can be seen as a complete horizontal reduction. As an implementation technique of horizontal reduction, we first propose *Upper/Lower-Reduction* (*UL-Reduction* in short), which represents each document as two bitmap vectors. UL-Reduction determines two thresholds, *lower* and *upper*, and constructs two bitmap vectors for each document by comparing the weight of each term with *upper* and *lower*. Using the bitmap vectors we can obtain the *UL-distance*, an approximate

* Each component of a document vector corresponds to a term in the text document, and its value corresponds to the term frequency in the text document.

1084-4627/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDE.2012.115

IEEE computer society

distance between document vectors. We formally prove the lower bound property of the UL-distance.

We next propose a novel horizontal reduction technique, *Quantization-Reduction* (*Q-Reduction* in short), as a generalized version of UL-Reduction. UL-Reduction uses only two bitmap vectors to approximate a document vector, and thus the difference between the UL-distance and the real distance can be large. To reduce this difference, Q-Reduction represents a text document as multiple bitmap vectors using multiple thresholds. Similar to the UL-distance, we can obtain the *Q-distance* using the multiple bitmap vectors of Q-Reduction. We formally prove that the Q-distance is also a lower bound of the actual distance. By exploiting the Q-distance, we then propose an efficient k-nearest neighbor (k-NN) search algorithm as a solution for efficient similar document search.

The intuition behind UL-Reduction and Q-Reduction is as follows: only a few components of a document vector have notable values while most of them are zero or close to zero. Thus, by representing those notable values using a few bitmap vectors, we can efficiently obtain tight lower bounds of actual distances. Due to its simplicity, the proposed technique can easily be applied to large document sets. In addition, our reduction technique is very suitable for a frequently changing document set since it handles updated document vectors only without a need to reconstruct the entire document set. In other words, the proposed technique processes a document vector without considering the rest of the document vectors (at the instance-level) while the traditional techniques such as LSI or DF consider the entire set of document vectors as a group (at the schema-level) in the reduction process.

The contribution of the paper can be summarized as follows. First, we present a novel notion of vertical and horizontal reductions as a new classification criterion of dimensionality reduction techniques. Second, as the horizontal reduction techniques, we propose UL-Reduction and Q-Reduction and formally prove the lower bound property of the UL-distance and the Q-distance. Third, using Q-Reduction we propose an efficient k-NN search algorithm for similar document search and formally prove its correctness. Finally, through extensive experiments, we show that Q-Reduction is superior to existing reduction techniques.

The rest of the paper is organized as follows. In Section II, we explain the related work on similar document search and dimensionality reduction. In Section III, we present a new point of view on dimensionality reduction. In Section IV, we describe the concept of horizontal reduction and propose UL-Reduction. In Section V, we propose Q-Reduction as a generalization of UL-Reduction and present the k-NN search algorithm using Q-Reduction. In Section VI, we empirically show the superiority of UL-Reduction and Q-Reduction. Finally, we conclude the paper in Section VII.

II. Related Work

A. Similarity Search in Text Classification and Clustering

Text classification organizes text documents into pre-defined classes (or categories) [3], [4]. The k-NN classification is one of the most popular text classification methods [4]. Given a query document, the k-NN classifier first retrieves k text documents most similar to the query from the training set, and then, uses their classes to determine the class to which the query belongs [11], [4]. The k-NN classification, while known as an effective classification method [4], is known inefficient due to excessive computation of similarity [3]. The detailed reasons of inefficiency are (1) that the k-NN classifier computes the similarity measure for all training documents [3], and (2) that computing the similarity measure incurs high cost due to their high dimensionality [7]. Although there have been many attempts to improve the efficiency of k-NN classification, most of them improves it by sacrificing classification effectiveness (i.e., accuracy). This is because they reduce the size of the training set [12] or compute the similarity measure using only a few selected terms [13].

Text clustering groups text documents into meaningful clusters using the traditional clustering techniques such as k-means and k-medoids algorithms [5]. In the k-means algorithm, for example, similarities of all text documents are repeatedly computed against k representative documents until there is no change in all clusters [7]. Hence, as in text classification, the high dimensionality is a major cause of performance degradation in text clustering as well. Although many dimensionality reduction techiques [5], [6] have been proposed to improve the efficiency of text clustering, most of them suffer from the disparity of clustering results before and after reduction.

In this paper we propose a novel reduction technique for text classification and text clustering. For text classification, our technique improves the the performance without any sacrifice on classification accuracy and without any reduction of the training set. For text clustering, our technique provides the same and consistent clustering results before and after reduction.

B. Dimensionality Reduction for Text Documents

There have been many efforts on dimensionality reduction for text documents. Examples include Latent Semantic Indexing (LSI) [8], [3], [6], Random Projection (RP) [14], [6], Locality Preserving Indexing (LPI) [5], Linear Discriminant Analysis (LDA) [15], and Document Frequency (DF) [6], [9].

LSI represents the entire set of documents using a few synthetic terms derived from a large number of original ones after analyzing patterns of their co-occurrence. It considers the semantics such as synonyms and hyponyms and effectively removes noise and redundancy within the document set [6]. However, it is not applicable to a large document set, which has a huge number of terms resulting in high dimensionality, since it uses the singular valued decomposition (SVD) whose time and space complexities are very high ($O(d^2n)$ and $O(dn)$, respectively, where d is the dimensionality and n the number of documents) [10], [6]. Besides, it is not appropriate for dynamic document sets with frequent insertions and deletions because every update basically leads to a whole recomputation of SVD over the entire set of documents. LPI

and LDA are variants of LSI and have similar characteristics to LSI. Readers are referred to [5], [15] for their details.

RP projects all the text documents onto one lower-dimensional space or onto multiple lower-dimensional spaces, which may differ for each class. RP is much simpler than LSI and does not distort text documents significantly [14]. However, mining results of RP are not as accurate as those of LSI because it ignores the semantic information of the document set [14]. DF is a variant of RP and has similar characteristics to RP. Readers are referred to [6], [9] for its details.

Another area where there have been many efforts on dimensionality reduction is time-series mining. Here, dimensionality reduction is used to resolve the high dimensionality problem of time-series data [16], [17], [18], [19], [20], [21]. A time-series is a sequence of real numbers representing values at specific time points. Since a text document can be modeled as a high dimensional vector of term occurrence, if we consider these high-dimensional vectors as time-series data [17], [22], we may use the time-series reduction techniques such as Discrete Fourier Transform (DFT) [16], [17], [20], Piecewise Aggregate Approximation (PAA) [18], [21], and Symbolic Aggregate Approximation (SAX) [19] for text documents. However, they are not suitable for text documents due to lack of correlation among adjacent components. That is, adjacent components of a time-series are strongly correlated with one another; thus, most of its energy can be concentrated into a few coefficients by the time-series reductions [16]. In contrast, adjacent components of a document vector are not related to one another; thus, its energy cannot be summarized as a few coefficients.

In summary, existing reduction techniques have many problems as follows: (1) some (LSI, LPI, and LDA) are too complex to be applied to large document sets, (2) some (RP and DF) produce low accuracy being too simple, and (3) some (DFT, PAA, and SAX) are not applicable to text documents due to the characteristics of text documents. To resolve these problems, we propose a new approach to dimensionality reduction, called *horizontal dimensionality reduction* and show that it makes similar document search very efficient while providing the same mining results before and after reductions.

III. A New View of Dimensionality Reduction: Vertical and Horizontal Reductions

Existing reduction techniques can be classified into *feature extraction* and *feature selection* [3], [6]. Feature extraction represents data objects using a few number of components *newly derived* from a large number of original ones [3], [6]. LSI, LPI, LDA, RP, DFT, PAA, and SAX reviewed in Section II belong to this category. Feature selection represents data objects using a few number of the most relevant (or useful) components *selected* from a large number of original ones [3], [6]. DF and LDA belong to this category. Regardless of the categories, however, all these techniques basically transform the entire set of data objects to a fixed lower-dimensional space. Thus, we can say that they perform the reduction *at the schema-level*. Schema-level reduction projects all data objects onto the same lower-dimensional space by exploiting the relationships among components of the data set. As explained in Section II, however, most existing schema-level reductions are not appropriate for large document sets due to their complex reduction processes. This observation motivates us to devise a new reduction approach that is simple yet effective.

We now propose a new concept of *instance-level* dimensionality reduction. In contrast to schema-level reduction, instance-level reduction projects each data object onto its own lower-dimensional subspace by using its component values. Since instance-level reduction transforms data objects one by one independently, it is very simple, and thus, appropriate for large data sets. It is also well-suited for dynamic document sets where frequent updates occur. We formally define instance-level reduction as *horizontal (dimensionality) reduction* and schema-level reduction as *vertical (dimensionality) reduction*.

Definition 1: Suppose that a function f transforms a d-dimensional vector $A = (a_1, \ldots, a_d)$ to an $d'(< d)$-dimensional vector $A' = (a'_1, \ldots, a'_{d'})$ such that $A' = f(A)$. If f projects all the vectors (i.e., instances) onto the same lower-dimensional space, we call f a *vertical reduction function*, and we say that A is *vertically reduced to* A'. In contrast, if f projects each vector onto its own lower-dimensional subspace possibly different from others, we call f a *horizontal reduction function*, and we say that A is *horizontally reduced to* A'. □

Fig. 1 shows the concept of vertical and horizontal reductions. As shown in the lower part of the figure, all the vectors reduced by the vertical reduction function f_{ver} are in the same lower-dimensional space. In contrast, as shown in the upper-right part of the figure, the individual vectors reduced by the horizontal reduction function f_{hor} are in different lower-dimensional subspaces.

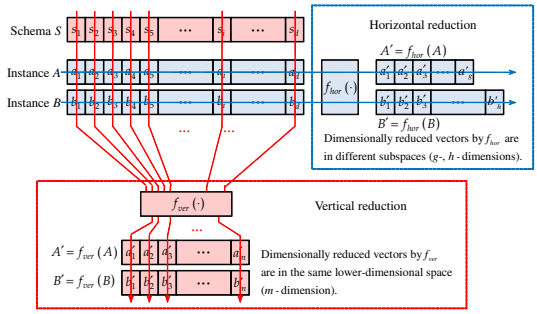


Fig. 1. The concept of vertical and horizontal reductions.

Existing reduction techniques in Section II can be classified into vertical and horizontal reductions as shown in Fig. 2. LSI, LPI, and LDA belong to vertical reduction since they represent all the document vectors using a few number of the same synthetic terms. RP and DF also belong to vertical reduction, however, they also have characteristics of horizontal reduction as described in Section II-B. PAA, SAX, and DFT also belong to vertical reduction because they project all the time-series to the same lower-dimensional space that is composed of a fixed number of features (such as arithmetic means or Fourier coefficients) for all time-series. In summary,

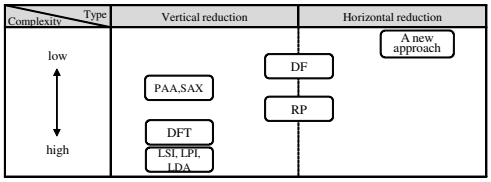


Fig. 2. Classification of existing techniques by vertical and horizontal reductions.

we observe that most existing reduction techniques belong to schema-level vertical reduction, and there are few attempts to exploit instance-level horizontal reduction.

Fig. 2 also shows the complexities of the dimensionality reduction techniques. Complexity is a crucial issue in evaluating reduction techniques because high complexity leads to the excessively long reduction time that is not applicable to large and/or dynamic document sets. For example, LSI is not appropriate for such document sets since it has the complexity of $O(d^2n)$ for computing SVD. From Fig. 2, therefore, we can finally conclude that there have been few efforts on simple but effective horizontal reduction techniques for a huge number of text documents. In the following two sections, we propose new horizontal reduction techniques that are efficient but yet 100% effective.

IV. Horizontal Reduction for Text Documents

A. UL-Reduction: An Implementation of Horizontal Reduction

UL-Reduction, a basic implementation of the horizontal reduction for text documents, is derived from the highly sparse nature of document vectors. UL-Reduction is formally defined in Definition 2.

Definition 2: Given an d-dimensional document vector $A = (a_1, \cdots, a_d)$ of size 1.0 (i.e., unit length) and its two thresholds, *lower* and *upper*, *UL-Reduction* creates two d-dimensional bitmap vectors, named *lower*, *upper vectors*, respectively, $A^l = (a_1^l, \cdots, a_d^l)$ and $A^u = (a_1^u, \cdots, a_d^u)$ by Eq. (1).

$$a_i^l = \begin{cases} 1 & \text{if } 0 \le a_i \le lower \\ 0 & \text{otherwise} \end{cases}$$
$$a_i^u = \begin{cases} 1 & \text{if } upper \le a_i \le 1 \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $0 \le lower < upper$. □

In other words, UL-Reduction divides the range of $[0, 1]$ into three sub-ranges, $[0, lower]$, $(lower, upper)$, and $[upper, 1]$. It then makes two bitmap vectors for the sub-ranges $[0, lower]$ and $[upper, 1]$. Here, we note that UL-Reduction performs the reduction of each document vector independently of other document vectors. Example 1 shows how UL-Reduction works.

Example 1: Fig. 3 shows the bitmap vectors A^l, A^u and B^l, B^u constructed from the 12-dimensional document vectors A and B by UL-Reduction. The *lower* thresholds of A and B are set to be 0.32 and 0.22, respectively; the *upper* thresholds are set to be 0.68 and 0.58, respectively. □

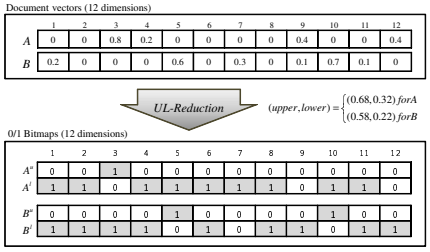


Fig. 3. An example of UL-Reduction .

Using the bitmap vectors of UL-Reduction we compute an approximate similarity between two document vectors. Although the cosine similarity is a widely-used measure in text mining and information retrieval areas [7], [1], [23], we use the Euclidean distance as the similarity measure unless otherwise specified. If document vectors are normalized to the unit length (i.e., 1.0), the cosine similarity has a direct relationship with the Euclidean distance, i.e., $\cos\theta = \sum_{i=1}^{n} a_i \cdot b_i = 1 - \frac{\sum_{i=1}^{n}(a_i - b_i)}{2} = 1 - \frac{D^2(A,B)}{2}$, where θ is the angle between two vectors $A = (a_1, \cdots, a_n)$ and $B = (b_1, \cdots, b_n)$, and D is the Euclidean distance between them. Thus, cosine similarity decreases as the Euclidean distance increases, and *vice versa*. We use the Euclidean distance since it is more intuitive than cosine similarity in similar document search on high dimensional spaces. We now propose a lower bound of the Euclidean distance between two document vectors and call it *UL-distance*.

Definition 3: Given d-dimensional document vectors $A = (a_1, \cdots, a_d)$ and $B = (b_1, \cdots, b_d)$ of size 1.0, let UL-Reduction produce A^l, A^u from A using *lowerA*, *upperA* and B^l, B^u from B using *lowerB*, *upperB*, respectively. Then, the *UL-distance* between A and B, denoted by $D_{ul}(A, B)$, is computed by Eq. (2).

$$D_{ul}(A,B) = \sqrt{n_{AB} \cdot (upperA - lowerB)^2 + n_{BA} \cdot (upperB - lowerA)^2}, \quad (2)$$
$$where\ n_{AB} = |\{i | a_i^u \wedge b_i^l = 1, upperA > lowerB\}|$$
$$and\ n_{BA} = |\{i | b_i^u \wedge a_i^l = 1, upperB > lowerA\}|$$ □

The primary advantage of the UL-distance is fast computation. The UL-distance can be computed by simple bitwise `AND` and count operations according to Definition 3. Here, $(upperA - lowerB)^2$ and $(upperB - lowerA)^2$ are computed only once. Thus, we compute it much faster than the Euclidean distance, which requires relatively complex operations. Example 2 compares the Euclidean distance $D(A, B)$ with UL-distance $D_{ul}(A, B)$ for A and B of Fig. 3.

Example 2: For the document vectors A and B of Fig. 3, the Euclidean distance $D(A, B)$ is 1.39 and the UL-distances $D_{ul}(A, B)$ is 0.59, as follows.

$$D(A,B) = \sqrt{(0-0.2)^2 + (0-0)^2 + (0.8-0)^2 + \cdots + (0-0.7)^2 + (0-0.1)^2 + (0.4-0)^2} = 1.39$$
$$D_{ul}(A,B) = \sqrt{1 \cdot (0.68 - 0.22)^2 + 2 \cdot (0.58 - 0.32)^2} = 0.59$$ □

Faloutsos et al. [17] showed the necessary condition in similarity search for time-series data that the real distance

($D_{real}(\cdot)$) between two time-series in the original space should be equal to or greater than the approximate distance ($D_{approx}(\cdot)$) in the transformed space (i.e., $D_{real}(\cdot) \geq D_{approx}(\cdot)$). In other words, the approximate distance should be a lower bound of the real distance to guarantee no false dismissal. Thus, the UL-distance should also be equal to or less than the real (Euclidean) distance if we use UL-Reduction in similar document search.

Lemma 1: For d-dimensional document vectors A and B of size 1.0, their UL-distance $D_{ul}(A, B)$ is a lower bound of the Euclidean distance $D(A, B)$, i.e., $D_{ul}(A, B) \leq D(A, B)$ holds.

PROOF: Suppose that A and B are reduced to A^l, A^u and B^l, B^u by using *lowerA*, *upperA* and *lowerB*, *upperB*, respectively, by UL-Reduction. Then, the i-th component of UL-distance in Definition 3 is used only when one of the following two conditions holds.

- Condition 1: $a_i^u \wedge b_i^l = 1$ and $upperA > lowerB$.
- Condition 2: $b_i^u \wedge a_i^l = 1$ and $upperB > lowerA$.

Let us consider Condition 1 first. In order to satisfy $a_i^u \wedge b_i^l = 1$, $a_i \in [upperA, 1]$ and $b_i \in [0, lowerB]$ should also be satisfied. In this case, if $upperA > lowerB$ holds, $(upperA - lowerB) \leq (a_i - b_i)$ also trivially holds (see Fig. 4). Similarly, by Condition 2, if $b_i^u \wedge a_i^l = 1$ and $upperB > lowerA$ hold, $(upperB - lowerA) \leq (b_i - a_i)$ also holds. Here, we note that both conditions cannot hold at the same time because a_i^u of A^u and a_i^l of A^l cannot be set to 1 at the same time according to Definition 2. Thus, $n_{AB} + n_{BA} \leq n$. Therefore, Lemma 1 holds by the following equations.

$$
\begin{aligned}
D_{ul}(A, B) &= \sqrt{\begin{array}{l} n_{AB} \cdot (upperA - lowerB)^2 \\ +n_{BA} \cdot (upperB - lowerA)^2 \end{array}} \\
&\leq \sqrt{\begin{array}{l} \sum_{i \in \{i | a_i^u \wedge b_i^l = 1, upperA > lowerB\}} (a_i - b_i)^2 \\ +\sum_{i \in \{i | b_i^u \wedge a_i^l = 1, upperB > lowerA\}} (b_i - a_i)^2 \end{array}} \\
&\leq \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} = D(A, B)
\end{aligned}
$$

□

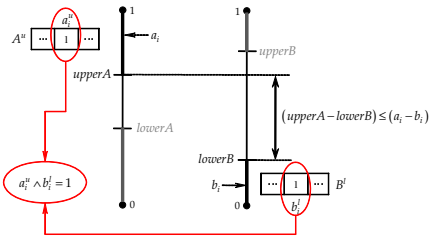


Fig. 4. Lower-bounding property of the UL-distance for the Euclidean distance.

B. Threshold Selection in UL-Reduction

In this section, we discuss how to determine the thresholds, *lower* and *upper*. We need to carefully select the thresholds since they highly influence the tightness of the bound of the UL-distance, and thus, the search performance. More specifically, a loose bound, i.e., a large difference between the UL- and Euclidean distances causes a lot of false alarms requiring that many real distance computations in similarity search [21]. Thus, a loose bound degrades the search performance. To show our selection method, we first define a *base range* of a document vector. Suppose that, for a document vector A, min_A is the smallest non-zero component value, and max_A is the largest one, then we call $[min_A, max_A]$ the *base range* of A. We then determine the thresholds by multiplying the predefined ratios to the base range.

Example 3: Given document vectors A and B of Fig. 3, we obtain the bitmap vectors of Figs. 5(a) and 5(b), where we set the predefined ratios to 10% and 20%, respectively. From Fig. 3, we obtain $min_A = 0.2$, $max_A = 0.8$, $min_B = 0.1$, and $max_B = 0.7$. We then compute the thresholds of Fig. 5(a) by multiplying the ratio of 10% to the base range as follows.

$$
\begin{aligned}
lowerA &= min_A + 0.1 \cdot |max_A - min_A| = 0.26, \\
upperA &= max_A - 0.1 \cdot |max_A - min_A| = 0.74; \\
lowerB &= min_B + 0.1 \cdot |max_B - min_B| = 0.16, \\
upperB &= max_B - 0.1 \cdot |max_B - min_B| = 0.64;
\end{aligned}
$$

Similarly, we can also derive the thresholds of Fig. 5(b) by using the ratio of 20%. □

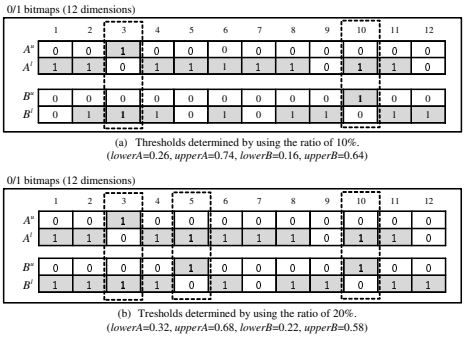


Fig. 5. Examples of determining *lower* and *upper* by the ratio-based method.

Example 4 compares different UL-distances on different bitmap vectors of Figs. 5(a) and 5(b).

Example 4: If the predefined ratio is 10% as in Fig. 5(a), $n_{AB} = n_{BA} = 1$, $D_{ul}(A, B)$ is $\sqrt{1 \cdot (0.74 - 0.16)^2 + 1 \cdot (0.64 - 0.26)^2} = 0.69$; if it is 20% as in Fig. 5(b), $n_{AB} = 1$, $n_{BA} = 2$, $D_{ul}(A, B)$ is $\sqrt{1 \cdot (0.68 - 0.22)^2 + 2 \cdot (0.58 - 0.32)^2} = 0.59$. The components marked by dotted squares contribute to n_{AB} or n_{BA}. □

Advantages of the proposed ratio-based method are that (1) it makes the bitmap construction very fast, and (2) it is easy to understand since it applies the same ratio to all document vectors. However, it fails to obtain a tight enough bound of the UL-distance to improve the k-NN search performance as we demonstrate in Section VI. We may find better thresholds by a more elaborate method, but the bounds of the UL-distance will still be far from optimal because many components that are within the range $(lower, upper)$ are not included in computing the UL-distance. In addition, if the improved method were very complex, we would not fully exploit advantages of horizontal reduction — fast reduction and fast similarity computation. Therefore, for performance improvement, we derive a new

technique that generalizes UL-Reduction instead of finding the optimal thresholds in UL-Reduction.

V. Q-Reduction: A Generalization of UL-Reduction

A. The Concept of Q-Reduction

In this section, we propose *Quantization Reduction* (*Q-Reduction* in short), an advanced implementation of horizontal reduction. Q-Reduction increases the lower bound distance using multiple bitmap vectors. UL-Reduction uses only two bitmap vectors, and its UL-distance can be much smaller than the real distance. To solve this problem, Q-Reduction divides the base range into multiple sub-ranges through the quantization process and constructs multiple bitmap vectors, one for each sub-range. By using these multiple bitmap vectors, the lower bound of Q-Reduction can be much more close to the real distance than UL-distance is.

To use multiple bitmap vectors in Q-Reduction, we need to determine how to divide the base range into multiple sub-ranges. We note the following two observations.

- Observation 1: Only a few components (i.e., terms) of a document vector have large values, which are relatively close to 1.0. Fig. 6 shows the term distribution in a document of RCV1 [24]. We note that only $0.1 \sim 0.2\%$ of $47,219$ components have relatively large values.
- Observation 2: Most components in a document vector have values of zero. After examining $23,149$ documents of RCV1, we note that 99.84% of $47,219$ components have values of zero in one document.

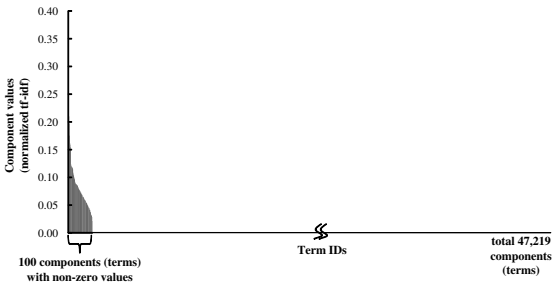


Fig. 6. Distribution of term occurrences in one specific document of RCV1.

Based on these observations, we select a few large component values as thresholds. Q-Reduction constructs several *upper* bitmap vectors and one *lower* bitmap vector. That is, only a few components that significantly affect the real distance are recorded in *upper* bitmap vectors, and more than 99% components whose values are zeros are recorded in the *lower* bitmap vector. We now formally define Q-Reduction as follows.

Definition 4: Let $A = (a_1, \ldots, a_d)$ be an d-dimensional document vector of size 1.0 and max_A be $\max_{i=1}^{d} a_i$. Given the number of sub-ranges m and a set of thresholds $Q = \{q_0, \cdots, q_{m-1}\}$ ($q_0 = 0$, $q_{m-1} = max_A$, $q_k < q_{k+1}$, $k = 0, 1, \ldots, m-2$), *Q-Reduction* creates an d-dimensional *lower* bitmap vector $A^0 = (a_1^0, \ldots, a_d^0)$ and $(m-1)$ d-dimensional *upper* bitmap vectors $A^j = (a_1^j, \ldots, a_d^j)$ $(j = 1, 2, \cdots, m-1)$ by Eq. (3).

$$a_i^0 = \begin{cases} 1 \text{ if } \ a_i = 0 \ (= q_0) \\ 0 \text{ otherwise} \end{cases}$$
$$a_i^j = \begin{cases} 1 \text{ if } \ (1 \le j < m-1) \wedge (q_j \le a_i < q_{j+1}) \\ 1 \text{ if } \ (j = m-1) \wedge (a_i = q_j) \\ 0 \text{ otherwise} \end{cases} \quad (3)$$

□

Bitmap vector A^0 reflects Observation 2, and each bit is set to 1 if the corresponding component is zero. Bitmap vectors A^j reflects Observation 1, and each bit is set to 1 by the following two conditions. The first condition of $(1 \le j < m-1) \wedge (q_j \le a_i < q_{j+1})$ is for selecting the components whose values are in $[q_j, q_{j+1})$. The second condition of $(j = m-1) \wedge (a_i = q_j)$ is for constructing a special bitmap vector for the largest component value that contributes most to the distance between document vectors. In other words, Q-Reduction divides $[0, max_A]$ into the following m sub-ranges and then constructs the bitmap vectors corresponding to those sub-ranges.

$$[q_0 = 0], [q_1, q_2), \cdots, [q_{m-3}, q_{m-2}), [q_{m-2}, q_{m-1}), [q_{m-1} = max_A]$$

Fig. 7 shows an example where each document vector A or B in Example 1 is represented as four bitmap vectors by Q-Reduction.

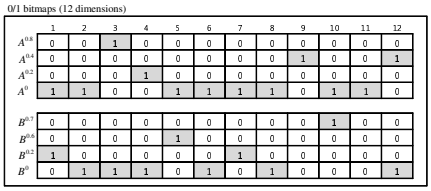


Fig. 7. Example *upper* and *lower* bitmap vectors of Q-Reduction.

Now, we need to determine a set of thresholds for each document vector. We determine the thresholds of a document vector based on its real component values. If an ordered set of non-zero component values of a document vector is given by $V = \{v_1, v_2, v_3, \cdots\}$ $(v_i \ge v_{i+1})$, we determine the ordered set $Q = \{q_0, \cdots, q_{m-1}\}$ as follows. First, q_0 is determined as 0. Second, q_k $(k = 1, \cdots, m-1)$ is determined by $v_{2^{m-1-k}}$ so as to make the higher thresholds reflect the real component values more closely and, at the same time, to decrease the number of sub-ranges†. As the number of documents increases, so does the number of non-zero components, and this may require more bitmap vectors. However, since the number of bitmap vectors logarithmically increase in the number of non-zero components N ($N = 2^{m-1-k}$, thus $m = \log_2 N + 1 + k$), this increase is not significant. Despite its simplicity, this component- value-based threshold determination makes the bounds quite tight as we present in Section V-B. Thus, we believe it is a practically applicable method for Q-Reduction.

† If the number of non-zero components in a document vector is less than 2^{m-2}, we use $Q = \{q_0, q_{k'}, q_{k'+1}, \cdots, q_{m-1}\}$, where $q_{k'}$ is the minimum available, i.e., $v_{2^{m-1-k'}} \in V$ and $v_{2^{m-k'}} \notin V$.

We note that other methods that make the bound tighter can be devised, and we leave it as a future work.

B. Q-Distance: A Lower Bound Distance of Q-Reduction

1) BQ-distance: In Definition 5, we define the Basic Q-distance (BQ-distance). For a tighter bound, we define the Tight Q-distance (TQ-distance) in Section V-B2. We define the BQ-distance using the UL-distance in Definition 3.

Definition 5: Let d-dimensional document vectors $A = (a_1, \ldots, a_d)$ and $B = (b_1, \ldots, b_d)$ of size 1.0 be reduced to $A^0, \cdots, A^{m-1}$ and $B^0, \cdots, B^{m-1}$ by the sets of thresholds $Q^A = \{q_0^A, \cdots, q_{m-1}^A\}$ and $Q^B = \{q_0^B, \cdots, q_{m-1}^B\}$, respectively. We then define the *BQ-distance*, denoted by $D_{bq}(A, B)$, as the distance computed by Eq. (4) using the *lower* and *upper* bitmaps of A and B.

$$\begin{gathered} D_{bq}(A,B) = \sqrt{\textstyle\sum_{j=1}^{m-1} D_j^2(A,B)}\ , \\ \text{where } D_j(A,B) = D_{ul}(A,B). \\ \text{In } D_{ul}(A,B),\ lowerA = q_0^A,\ upperA = q_j^A, \\ lowerB = q_0^B,\ upperB = q_j^B,\ A^l = A^0,\ A^u = A^j, \\ B^l = B^0,\ B^u = B^j, \text{and} 1 \le j \le m-1. \end{gathered} \tag{4}$$ □

The BQ-distance will be much larger than UL-distance due to the following two reasons. First, multiple *upper* bitmap vectors of Q-Reduction reflect more components than a single *upper* bitmap vector of UL-Reduction does. That is, from the viewpoint of UL-Reduction, Q-Reduction increases n_{AB} and n_{BA}. Second, the difference between some $upper$ and the $lower$ thresholds in Q-Reduction are likely to be larger than that of UL-Reduction. This is because Q-Reduction has multiple $upper$ thresholds that are likely to be larger than the $upper$ threshold of UL-Reduction, and those larger $upper$ thresholds of Q-Reduction produce larger differences. That is, from the viewpoint of UL-Reduction, Q-Reduction is likely to increase $(upperA - lowerB)$ and $(upperB - lowerA)$. Example 5 shows BQ-distance $D_{bq}(A, B)$ between A and B of Fig. 7.

Example 5: $D_{bq}(A, B)$, the BQ-distance between A and B of Fig. 7, can be computed as Eq. (5).

$$\begin{aligned} D_{bq}(A,B) &= \sqrt{\{1\cdot(0.8-0)^2 + 1\cdot(0.4-0)^2 + 1\cdot(0.2-0)^2\}} \\ &\quad \overline{+\{1\cdot(0-0.7)^2 + 1\cdot(0-6.1)^2 + 2\cdot(0.2-0)^2\}} \\ &= 1.39 \end{aligned} \tag{5}$$ □

As shown in Eq. (5), the BQ-distance of 1.33 is much closer to the real distance of 1.39 than the UL-distance of 0.69 or 0.59 in Example 4 is. □

The following lemma shows the lower bound property of the BQ-distance.

Lemma 2: For d-dimensional document vectors A and B of size 1.0, their BQ-distance $D_{bq}(A, B)$ is a lower bound of the Euclidean distance $D(A, B)$, i.e., $D_{bq}(A, B) \le D(A, B)$ holds.

PROOF: According to Definition 4, m bitmap vectors are generated from m *disjoint* sub-ranges, and thus, for every i $(= 1, \cdots, d)$, the case of $a_i^j = a_i^k = 1$ never holds if $j \ne k$ $(0 \le j, k \le m-1)$. Hence, the i-th component used for $D_{ul}(A^j, B^j)$ cannot be used for $D_{ul}(A^k, B^k)$ if $j \ne k$. Therefore, the following Eq. (6) holds, and the BQ-distance is a lower bound of the Euclidean distance (see Fig. 8).

$$\begin{aligned} D_{bq}(A,B) &= \sqrt{\textstyle\sum_{j=1}^{m-1}(D_{ul}^2(A^j,B^j))} \\ &= \sqrt{\textstyle\sum_{j=1}^{m-1}\left(\sum_{i\in\{i|a_i^j\wedge b_i^0=1\}}(q_i^A-0)^2 + \sum_{i\in\{i|a_i^0\wedge b_i^j=1\}}(q_i^B-0)^2\right)} \\ &\le \sqrt{\textstyle\sum_{j=1}^{m-1}\left(\sum_{i\in\{i|a_i^j\wedge b_i^0=1\}}(a_i-0)^2 + \sum_{i\in\{i|a_i^0\wedge b_i^j=1\}}(b_i-0)^2\right)} \\ &= \sqrt{\textstyle\sum_{(i,j)\in\{(i,j)|(j=1,\cdots,m-1)\wedge((a_i^j\wedge b_i^0)\vee(a_i^0\wedge b_i^j))\}}(a_i-b_i)^2} \\ &\le \sqrt{\textstyle\sum_{(i,j)\in\{(i,j)|(j=1,\cdots,m-1)\wedge((a_i^j\wedge b_i^0)\vee(a_i^0\wedge b_i^j))\}}(a_i-b_i)^2} \\ &\quad \overline{+\sum_{(i,j)\notin\{(i,j)|(j=1,\cdots,m-1)\wedge((a_i^j\wedge b_i^0)\vee(a_i^0\wedge b_i^j))\}}(a_i-b_i)^2} \\ &= \sqrt{\textstyle\sum_{i=1}^{d}(a_i-b_i)^2} = D(A,B) \end{aligned} \tag{6}$$ □

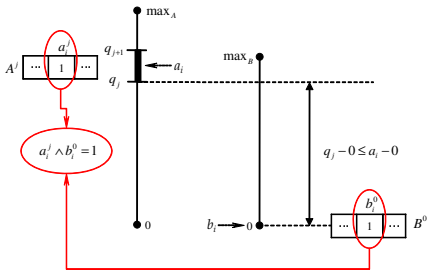


Fig. 8. Lower-bounding property of the Q-distance for the Euclidean distance.

2) TQ-distance: In this section, we propose the Tight Q-distance (TQ-distance), a lower bound distance tighter than the BQ-distance. The TQ-distance uses real component values in computing the lower bound while the BQ-distance uses the thresholds, and thus, the TQ-distance becomes larger than the BQ-distance.

The BQ-distance of Definition 5 can be simplified as follows. First, $upperA > lowerB$ and $upperB > lowerA$ always hold since $upperA = q_j^A$ and $upperB = q_j^B$, and $lowerA = q_0^A = 0$ and $lowerB = q_0^B = 0$ by Definition 4. Second, since $D_j(A, B) = D_{ul}(A, B)$, the UL-distance in Definition 3 and the BQ-distance in Definition 5 are simplified as in Eqs. (7) and (8), respectively.

$$\begin{gathered} D_{ul}(A,B) = \sqrt{n_{ABj}\cdot(q_j^A)^2 + n_{BAj}\cdot(q_j^B)^2}\ , \\ \text{where } n_{ABj} = |\{i|a_i^j \wedge b_i^0 = 1\}|,\ n_{BAj} = |\{i|b_i^j \wedge a_i^0 = 1\}|, \\ \text{and } 1 \le j \le m-1. \end{gathered} \tag{7}$$

$$\begin{aligned} D_{bq}(A,B) &= \sqrt{\textstyle\sum_{j=1}^{m-1} D_j^2(A,B)} \\ &= \sqrt{\textstyle\sum_{j=1}^{m-1} D_{ul}^2(A,B)} \\ &= \sqrt{\textstyle\sum_{j=1}^{m-1}\left(n_{ABj}\cdot(q_j^A)^2 + n_{BAj}\cdot(q_j^B)^2\right)}\ . \end{aligned} \tag{8}$$

According to Eq. (8), the BQ-distance is determined by n_{AB}, n_{BA}, and threshold q_j. Threshold q_j, however, represents all component values that are within the sub-range $[q_j, q_{j+1})$. Thus, we can increase BQ-distance if we use their real values rather than the thresholds.

To use real component values, we employ the following two ordered lists. First, the k-th component list B_k that includes a threshold q_k ($\in Q$, a set of thresholds defined in Section V-A) is described in Eq. (9). Second, a list of cumulative squared sums S_k is defined in Eq. (10).

$$\begin{aligned} B_k &= (v | q_k \le v < q_{k+1},\ v \in V) \\ &= (v_{k,1}(=q_k), v_{k,2}, \cdots, v_{k,l}), \\ &\text{where } l = |B_k| \text{ and } V \text{ is an ordered set of} \\ &\text{non-zero component values of a document vector.} \end{aligned} \tag{9}$$

$$S_k = \{s_1 = v_{k,1}^2, s_2 = s_1 + v_{k,2}^2, \cdots, s_l = s_{l-1} + v_{k,l}^2\} \tag{10}$$

S_k is precalculated and stored together with the bitmaps.

Definition 6 defines the TQ-distance using the ordered lists B_k and S_k.

Definition 6: Two d-dimensional document vectors $A = (a_1, \ldots, a_d)$ and $B = (b_1, \ldots, b_d)$ of size 1.0 are reduced to $A^0, \cdots, A^{m-1}$ and $B^0, \cdots, B^{m-1}$ by using the sets of thresholds $Q^A = \{q_0^A, \cdots, q_{m-1}^A\}$ and $Q^B = \{q_0^B, \cdots, q_{m-1}^B\}$, respectively. Let B_j^A and B_j^B be the j-th component lists, and S_j^A and S_j^B the lists of cumulative squared sums. We then define the *TQ-distance*, denoted by $D_{tq}(A, B)$, as in Eq. (11) using the *lower* and *upper* bitmap vectors of A and B.

$$D_{tq}(A,B) = \sqrt{\sum_{j=1}^{m-1} (s_{n_{AB_j}} + s_{n_{BA_j}})}\ \text{, where} \tag{11}$$

$$s_{n_{AB_j}} = \begin{cases} n_{AB_j}\text{-th element of } S_j^A & \text{if } n_{AB_j} \ge 1 \\ 0 & \text{if } n_{AB_j} = 0 \end{cases}$$

$$s_{n_{BA_j}} = \begin{cases} n_{BA_j}\text{-th element of } S_j^B & \text{if } n_{BA_j} \ge 1 \\ 0 & \text{if } n_{BA_j} = 0 \end{cases}$$

□

The TQ-distance is closer to the real distance than the BQ-distance even though both use the same bitmap vectors. This is because the TQ-distance uses the real component values (the exact cumulative squared sums) while the BQ-distance the thresholds, i.e., $n_{AB_j} \cdot q_k^2 \le s_{n_{AB_j}} (= \sum_{a=1}^{n_{AB_j}} v_{k,a}^2)$.

Example 6: The TQ-distance $D_{tq}(A, B)$ of Fig. 7 is computed as in Eq. (12) if $m = 4$.

$$D_{tq}(A,B) = \frac{\sqrt{\{(0.8)^2 + (0.4)^2 + (0.2)^2\}}}{+\{(0.7)^2 + (0.6)^2 + (0.2)^2 + (0.3)^2\}} = 1.35 \tag{12}$$

As shown in Eq. (12), the TQ-distance of 1.35 is tighter than the BQ-distance of 1.33 in Example 5. □

Lemma 3: For d-dimensional document vectors A and B of size 1.0, their TQ-distance $D_{tq}(A, B)$ is a lower bound of the Euclidean distance $D(A, B)$, i.e., $D_{tq}(A, B) \le D(A, B)$ holds.

PROOF: $|B_j^A| = |S_j^A|$ and $|B_j^B| = |S_j^B|$ hold by Eqs. (9) and (10), respectively. As shown in Eq. (11), n_{AB_j} (n_{BA_j}) is the number of components whose bitmaps A^j and B^0 (B^j and A^0) are both 1. Thus, $0 \le n_{AB_j} \le |S_j^A|$ and $0 \le n_{BA_j} \le |S_j^B|$. Moreover, $s_{n_{AB_j}}$ ($1 \le n_{AB_j} \le |B_j^A|$) and $s_{n_{BA_j}}$ ($1 \le n_{BA_j} \le |B_j^B|$), which are n_{AB_j}-th and n_{BA_j}-th elements of S_j^A and S_j^B, are sums of the first n_{AB_j} and n_{BA_j} elements of B_j^A and B_j^B, respectively. Thus, they are always equal to or less than real component values rendering the TQ-distance a lower bound of the real distance. □

C. k-NN Search Algorithm

Fig. 9 shows the k-NN search algorithm using Q-Reduction. Its basic concept comes from Corollary 1, which states that "we can exclude (i.e., prune) the document vector whose Q-distance to the query vector is larger than the Euclidean distance of the k-th similar document seen so far" [**?**]. We use the term 'Q-distance' here as a generic term for the TQ-distance and the BQ-distance.

Algorithm *k*-NN-Search

// Assumption: all document and query vectors have been normalized to size 1 in advance.

Input
(1) *DB*: a database containing document vectors with their Q-Reduced bitmaps;
(2) *Q*: the query vector;
(3) *k*: the number of similar vectors returned;

Output
kNNHeap: a *max heap* containing *k*-NN document vectors from the query vector *Q*;

Algorithm
1. Construct bitmaps Q^j (*j*=0,1,...,*m*-1) from *Q* by Q-Reduction;
2. Retrieve initial *k* document vectors {*D*} from *DB* and compute their Euclidean distances {d_E} to *Q*;
3. Initialize *kNNHeap* and insert <*ID*, d_E> for each document in {*D*} into *kNNHeap*;
   // *ID*: the document identifier.
   // d_E: the Euclidean distance of a document to *Q*.
4. <ID_{curr_max}, d_{curr_max}> = <*ID*, d_E> in the root node of *kNNHeap*;
5. **foreach** document vector *A* in *DB* except the initial *k* document vectors **do**
6. $d_q = D_q(Q,A)$; /* Q-distance */
7. **if**($d_q < d_{curr\_max}$) **then begin**
8. Retrieve the document vector *A* from *DB*;
9. $d_E = D(Q,A)$; /* Euclidean distance */
10. **if**($d_E < d_{curr\_max}$) then **begin**
11. Replace the root of *kNNHeap* with <*ID*, d_E> of *A* and readjust *kNNHeap*;
12. <ID_{curr_max}, d_{curr_max}> = <*ID*, d_E> in the root node of *kNNHeap*;
13. **end-if**
14. **end-if**
15. **end-for**

Fig. 9. k-NN search algorithm using UL-Reduction.

Corollary 1: Let the Euclidean distance between the query vector Q and the k-th similar document vector seen so far be d_{so-far}, Q-distance between a certain document vector A and Q be $D_q(Q, A)$, and their Euclidean distance be $D(Q, A)$. Then, Eq. (13) holds.

$$D_q(Q,A) > d_{so-far} \Rightarrow D(Q,A) > d_{so-far} \tag{13}$$

PROOF: By Lemmas 2 and 3, $D_q(Q, A) \le D(Q, A)$. □

Algorithm k-NN-Search in Fig. 9 finds the k most similar documents in one-pass using Q-distance and Linear-Time Top-k Sort [25]. We now explain the algorithm in detail. The inputs to the algorithm are a database DB that stores all document vectors, a query vector Q, and the number k of similar documents to be returned. We assume that each document vector in DB is stored with its m bitmap vectors as well as the thresholds (or the list S_k) for Q-Reduction. The outputs are k document vectors that are most similar to the query vector Q to be returned in *kNNHeap*. The *kNNHeap* is a k-sized maximum heap [26] that finds k closest (similar)

document vectors with the time complexity of $O(n)$ [‡]. In the algorithm, we first construct m bitmap vectors of the query vector Q (Line 1). We then retrieve k document vectors from *DB* and compute their real (Euclidean) distances to the query vector Q (Line 2). We store the identifiers of the document vectors together with their real distances in the result list *kNNHeap* (Line 3). Hereafter, *kNNHeap* maintains k most similar document vectors seen so far. For the document vector in the root node of *kNNHeap*, we denote its real distance to Q as d_{curr_max} and its identifier as ID_{curr_max} (Lines 4 and 12). In Lines 5 to 15, we investigate the rest of the document vectors in *DB* and find the final k most similar documents through the filtering process that exploits Q-distance. We first access bitmap vectors and exclude the document whose Q-distance d_q is larger than d_{curr_max} without accessing its real document vector by Corollary 1 (Lines 6 and 7). On the other hand, if $d_q \leq d_{curr\_max}$, we retrieve the document vector and compute its real (Euclidean) distance as d_E (Lines 8 and 9). If $d_E < d_{curr\_max}$, we then replace the root node of *kNNHeap* with the document vector A and readjust *kNNHeap* (Lines 10 to 13). At the completion of the algorithm, we get the k most similar document vectors stored in *kNNHeap*.

Theorem 1: Algorithm k-NN-Search correctly finds the k documents most similar to the query vector, i.e., it correctly finds the k document vectors whose Euclidean distances to the query vector are the k largest.

PROOF: *kNNHeap* always contains the k most similar document vectors seen so far from Corollary 1 and Lines 7 to 12 of the algorithm. □

VI. PERFORMANCE EVALUATION

A. Experimental Data and Environment

We use two real document sets, RCV1 [24] and Enron Emails in Bag of Words (EMAILS in short) [27], that are widely used in text mining. RCV1 comprises $804,414$ documents with $47,219$ terms while EMAILS $39,861$ documents with $28,102$ terms. As the weight of terms, RCV1 uses the term frequency-inverse document frequency (tf-idf) [24], while EMAILS the term frequency (tf) [27]. We normalize all the document vectors to size 1.0 as mentioned in Section IV. In storing document vectors, we consider non-zero components only. More precisely, for each vector, we store a list of <identifier, weight> pairs of non-zero components instead of a full high-dimensional vector.

In the experiments, we measure the costs of preprocessing (i.e., dimensionality reduction) and k-NN search. For preprocessing, we compare Q-Reduction (*Q-R*) using the TQ-distance with LSI and RP, which are widely used vertical reduction techniques for text documents. For k-NN search, we compare the following reduction techniques that obtain exact k-NN results from the *original* document vectors: 1) UL-Reduction (*UL-R*), 2) Q-Reduction using BQ-distance (*Q-R(BQ)*), 3) Q-Reduction using TQ-distance (*Q-R(TQ)*), 4) DFT, 5) SAX, and 6) no reduction, i.e., a sequential scan (*SEQ-SCAN*). Here, we consider only non-zero bytes in maintaining bitmap vectors. That is, for each bitmap vector, we store its non-zero bytes with their corresponding offsets in disk and dynamically reconstruct it in main memory when it is read from disk. We implement all reduction techniques and k-NN algorithms using C. We conduct all the experiments on a Pentium-4 3.2GHz Linux (Version 2.6.23) PC with 2GB of main memory.

For preprocessing, we measure the dimensionality reduction time and storage cost. We compare our Q-R with LSI and RP by measuring reduction time and size of reduced vectors. For k-NN search, we measure the number of document vectors accessed from the database and the wall clock time. In our k-NN search algorithm, accessing a document vector occurs only when the lower bound cannot prune the document vector. Hence, the number of document vectors accessed evaluates how effective (i.e., tight) the lower bound is, and the wall clock time evaluates the overall efficiency of the search algorithm – including the lower bound computation.

B. Preprocessing Cost

Experiment 1: Dimensionality reduction time and storage cost We set the dimension of the reduced vectors to $1,500$ for LSI and RP and the number m of sub-ranges to 5 for Q-R. The reason why we choose these numbers is that, for fair comparison of reduction techniques, we want to make their storage spaces for the reduced vectors to be similar to each other. We do not use the entire $39,861$ documents of EMAILS but instead use randomly selected $6,000$ documents since LSI cannot handle the entire set of documents due to its heavy SVD processing overhead. The results of RCV1 are very similar to those of EMAILS, and for brevity we explain EMAILS only for the preprocessing cost. For LSI, we use *las2*, the fastest SVD implementation of the most widely used package SVDPACK [28].

Table I shows the relative dimensionality reduction time and the relative storage space for the reduced vectors. Q-R performs the reduction 59.1 and 80.9 times faster than LSI and RP, respectively. The main reason for this speed up is that Q-R is much simpler due to its instance-level nature while LSI and RP are quite complex due to their schema-level nature. The storage spaces are made similar in all three techniques for fair comparison.

TABLE I
THE RELATIVE DIMENSIONALITY REDUCTION TIME AND STORAGE COST OF $6,000$ DOCUMENT VECTORS REDUCED.

Experimental method	Reduction time	Storage space for reduced vectors
LSI/Q-R	59.1	1.0
RP/Q-R	80.9	1.0

Fig. 10 shows the dimensionality reduction time as the number of documents is varied. Here, we observe that LSI and RP take significantly more time than Q-R due to the complex SVD process and the complex matrix multiplication,

[‡] A patent of the linear-time top-k sort algorithm using a maximum (or minimum) heap has been applied [25].

respectively. If the number of documents reaches tens of thousands, we cannot practically use LSI and RP due to this excessive preprocessing time. For this reason, most existing reduction techniques including LSI and RP use sampling to reduce the number of documents and perform similarity search using the sampled documents only. However, since the sampled documents cannot fully reflect the entire document set, the exact results of similarity search is sacrificed for performance. In contrast, our Q-R uses the entire document set and obtain the exact results.

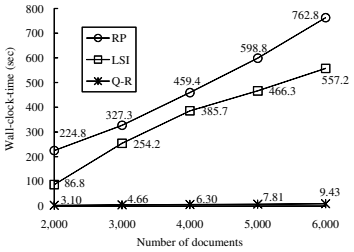


Fig. 10. Dimensionality reduction time as the number of documents is varied.

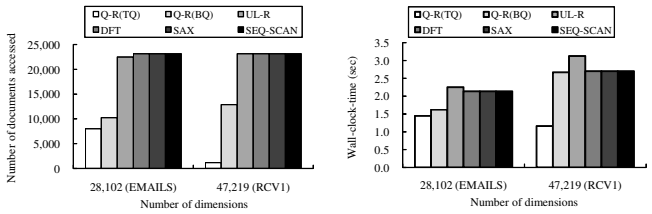
We also measure the dimensionality reduction time for inserting new documents to the document set. The results show that Q-R requires only a marginal time because it suffices to reduce the inserted documents only, but LSI and RP take much longer time than that of Q-R since they must reduce the entire set of documents including the new ones. The same phenomenon also occurs when deleting documents from the document set. Due to the excessive update time, we cannot use LSI and RP for dynamic sets with frequent updates. In contrast, our Q-R is very suitable for dynamic sets since its update time is negligible.

C. k-NN Search Cost

In this section, we discuss the performance of k-NN search that finds the *exact* k-NN results from the original document set. We compare UL-R, Q-R(BQ), Q-R(TQ), DFT, SAX, and SEQ-SCAN. As described in Experiment 1, LSI and RP cannot obtain the exact k-NN results since they handle the reduced set of document vectors only (rather than the original set of document vectors). Besides, LSI and RP are not suitable for dynamic document sets. In contrast, the six techniques above can obtain exact k-NN results and are suitable for dynamic document sets. In other words, the k-NN results of LSI and RP are not the same as those of the other six techniques. Since LSI and RP are inherently different from the six techniques, we exclude them from the k-NN search performance comparisons.

We set the experimental parameters as follows. First, for Q-R, we set the number of sub-ranges to nine for RCV1 and five for EMAILS since these numbers result in the best search performance, especially in search time. Second, to determine the thresholds (i.e., *lower* and *upper*) of UL-R, we apply the ratio of 10% to the base range. Third, we use six features for DFT as has been done in Faloutsos et al. [17]. Fourth, for SAX, we set the number of alphabets to ten and the reduced dimensionality to 16, as has been done in Lin et al. [19]. We use 20 documents randomly selected from each dataset as the input queries and average the result of those queries.

Experiment 2: k-NN search performance with different dimensionalities Fig. 11 shows the k-NN search performance on two datasets, RCV1 and EMAILS, whose dimensionalities are different from each other; the dimensionality of the former ($= 47,219$) is almost double that of the latter ($= 28,102$). We use the training set ($23,149$ document vectors) of RCV1 and $23,149$ document vectors randomly selected from EMAILS. We set the number k to be ten. Fig. 11(a) shows the number of documents accessed from the database; Fig. 11(b) the elapsed time of k-NN search.



(a) Number of documents accessed (Total 23,149 documents; k=10)

(b) k-NN search time (Total 23,149 documents; k=10)

Fig. 11. k-NN search performance by varying dimensionality.

In Fig. 11, we note that overall trends of EMAILS and RCV1 are very similar to each other. As shown in Fig. 11, Q-R(TQ) shows the best result in both document sets. Compared with SEQ-SCAN, Q-R(TQ) reduces the number of documents accessed by 19.6 times for RCV1 and 2.9 times for EMAILS and improves the k-NN search time by 5.5 times for RCV1 and 1.6 times for EMAILS[§]. We also note that the performance improvement by UL-R is very small and actually negligible. This is because the UL-distance is far from the Euclidean distance as we explained in Section V-A. In Fig. 11, DFT and SAX do not improve the performance at all. This is because, as we explained in Section II-B, these techniques are not effective in dimensionality reduction of text documents.

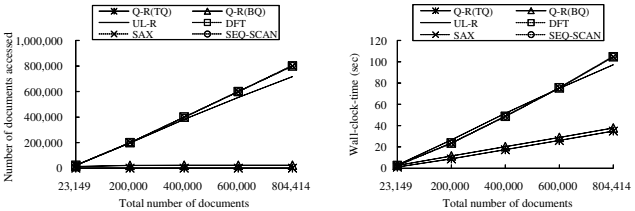
Even though overall trends for EMAILS and RCV1 are similar, all techniques except Q-R(TQ) show a slightly different result for each set. That is, as shown in Fig. 11(b), five techniques except Q-R(TQ) show a longer search time in RCV1 than in EMAILS. This happens because the dimensionality of RCV1 is larger than that of EMAILS, and the time for computing real distances and/or lower bounds is longer in RCV1. In the case of Q-R(TQ), however, the search time in RCV1 is shorter than that of EMAILS. This is because, as shown in Fig. 11(a), Q-R(TQ) prunes much more documents in RCV1 than in EMAILS. The main reason why we have a larger pruning effect in RCV1 is that the number of sub-ranges of RCV1 ($= 9$) is larger than that of EMAILS ($= 5$). The number of the component values included in the list of cumulative squared sums S_k increases as the number of thresholds (i.e., sub-ranges) increases. Thus, the tightness between the real distance and the TQ-distance in RCV1 is

[§] The reason why the improvement of k-NN search time is smaller than that of the number of documents accessed is that the time for computing Q-distances is relatively larger than that of other lower bound distances.

much larger than that in EMAILS.

As mentioned earlier in this section, we use nine sub-ranges for RCV1 and five for EMAILS. The reason why they have different optimal values is as follows. The number of components having non-zero values in a document of RCV1 is larger than that of EMAILS. Further, since RCV1 uses tf-idf as the component values while EMAILS tf[¶] (i.e., RCV1 has more unique values). Hence, RCV1 requires more sub-ranges than EMAILS to achieve high performance.

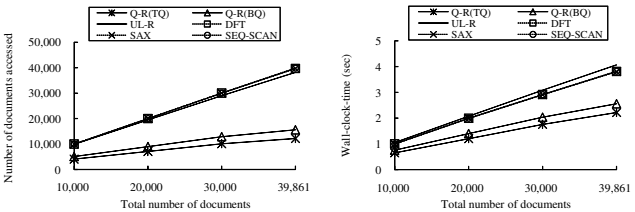
Experiment 3: k-NN search performance by varying the size of the document set We measure the k-NN search performance as the number of documents is varied. In Fig. 12, we vary the number of documents in RCV1 from $23,149$ (the training set) to $804,414$. In Fig. 13, we vary the number in EMAILS from $10,000$ to $39,861$. We note that, in Figs. 12 and 13, overall performance trends of RCV1 and EMAILS are similar to each other. Regardless of the document sets, our Q-R(TQ) outperforms other techniques significantly. In particular, the performance improvement of Q-R(TQ) increases as the size of document set increases. For example, in Fig. 12(a), Q-R(TQ) with $23,149$ documents improves the number of documents accessed by 19.6 times compared with SEQ-SCAN, while that with $804,414$ by 521.6 times. The main reason of this improvement is that the ratio of the documents pruned for a fixed k increases as the size of the document set increases.



(a) Number of documents accessed (47,219 dimensions; k=10)

(b) k-NN search time (47,219 dimensions; k=10)

Fig. 12. k-NN search performance as the size of the document set is varied (RCV1).



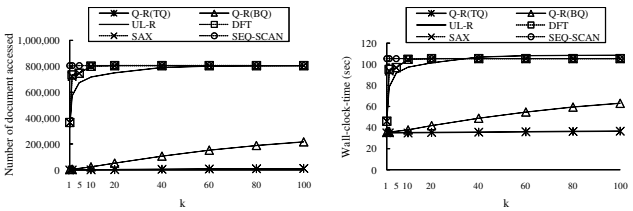
(a) Number of documents accessed (28,102 dimensions; k=10)

(b) k-NN search time (28,102 dimensions; k=10)

Fig. 13. k-NN search performance as the size of the document set is varied (EMAILS).

Experiment 4: k-NN search performance by varying k Figs. 14 and 15 show the k-NN search performance as k is varied. Larkey et al. [29] and Yang [11], [30] used 20 and 25

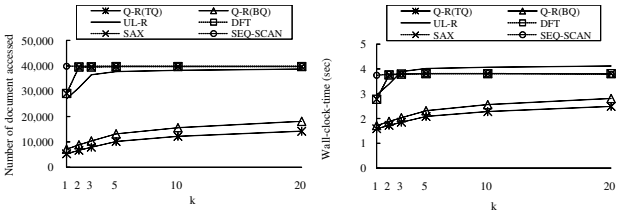
[¶] The number of unique component values in a document is usually larger when using tf-idf and smaller when using tf. The reason is, when using tf-idf, two components with the same tf value may have different idf values resulting in different tf-idf values.



(a) Number of documents accessed (Total 804,414 documents; 47,219 dimensions)

(b) k-NN search time (Total 804,414 documents; 47,219 dimensions)

Fig. 14. k-NN search performance as k is varied (RCV1).



(a) Number of documents accessed (Total 39,861 documents; 28,102 dimensions)

(b) k-NN search time (Total 39,861 documents; 28,102 dimensions)

Fig. 15. k-NN search performance as k is varied (EMAILS).

as the maximum value of k, respectively. Using these values as references, we vary k for RCV1, a larger set ($801,414$ documents), from 1 to 100 and k for EMAILS, a smaller set ($39,861$ documents), from 1 to 20. In these figures, the results of SEQ-SCAN are naturally constant since it needs to access all the documents for any value of k. The results of the other techniques including UL-R and Q-R increase as k increases since they prune less number of documents. Specifically, in Algorithm k-NN-Search of Fig. 9, as k increases, so do the current pruning distance, d_{curr_max}, and the number of pruned documents whose lower bound is larger than d_{curr_max}. UL-R, DFT, and SAX do not show a meaningful improvement since they are not effective in dimensionality reduction of text documents. The results of Q-R(TQ) increase only very slightly because the TQ-distance is a very tight lower bound maximizing the pruning effect.

From the results of Experiments 1 to 4, we conclude the following. From Experiment 1, we conclude that Q-Reduction is more appropriate for large document sets than LSI and RP since Q-Reduction, which uses instance-level reduction, reduces documents much faster than LSI or RP, which uses schema-level reduction. We also conclude that Q-Reduction works much better for dynamic document sets with frequent updates since Q-Reduction needs to consider only the updated documents while LSI and RP should consider the entire document set for every update. From Experiments 2, 3, and 4, we observe that Q-Reduction improves the k-NN search performance significantly compared with existing reduction techniques by reducing the number of document accesses. The major reason for this improvement is that the Q-distance is a tighter lower bound of the real Euclidean distance than those of other reduction techniques. Therefore, we finally conclude that Q-Reduction is the best dimensionality reduction technique that produces the exact results over large-scale document sets with frequent updates.

VII. Conclusions

In this paper, we have proposed new dimensionality reduction techniques for efficient similar search in large-scale document sets with frequent updates. Similar document search is one of the most important technical issues for text classification and clustering, and the dimensionality reduction technique is considered indispensable for efficient search. First, we have presented a new viewpoint of classifying the reduction techniques into *vertical* and the *horizontal* reductions – identifying that most existing techniques belonged to vertical reduction. We have also shown that the previous reduction techniques are not adequate for large-scale document sets with frequent updates. Second, we have presented UL-Reduction as an implementation of horizontal reduction for text documents, and then, proposed Q-Reduction as its generalization. We have formally proved that UL-distance and Q-distance are lower bounds of the real distance. Third, we have proposed the k-NN search algorithm using Q-Reduction and formally proved its correctness. Fourth, we have shown the superiority of Q-Reduction through extensive experiments with real data sets. Compared with the well-known LSI and RP, Q-Reduction reduces the preprocessing (reduction) time and the update time by one or two orders of magnitude. For k-NN search, Q-Reduction with TQ-distance beats all the existing reduction techniques. In particular, it outperforms the sequential scan by one to three orders of magnitude. Therefore, we conclude that the concept of horizontal reduction is crucial for fast similar search in large-scale document sets with frequent updates.

Acknowledgment

This work was partially supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2011-0020498). This work was partially supported by the Defense Acquisition Program Administration and the Agency for Defense Development under the contract (UD060048AD). We are indebted to the anonymous reviewers for their incisive comments, which helped make the paper more complete. We would like to acknowledge great help of Jun-Sung Kim in implementing the algorithm.

References

[1] A. Stavrianou, P. Andritsos, and N. Nicoloyannis, "Overview and semantic issues of text mining," *ACM SIGMOD Record*, vol. 36, no. 3, pp. 23–34, 2007.

[2] M. Hearst, "Untangling text data mining," in *Proc. the 32nd Annual Meeting of the Association for Computational Linguistics*, College Park, Maryland, Jun. 1999, pp. 3–10.

[3] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1–47, 2002.

[4] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proc. the 22nd Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, Berkeley, California, Aug. 1999, pp. 42–49.

[5] D. Cai, X. He, and J. Han, "Document clustering using locality preserving indexing," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 12, pp. 1624–1637, Dec. 2005.

[6] B. Tang, M. Shepherd, and E. Milios, "Comparing and combining dimension reduction techniques for efficient text clustering," in *Proc. Int'l Workshop on Feature Selection for Data Mining: Interfacing Machine Learning and Statistics*, Newport Beach, California, Apr. 2005, pp. 17–26.

[7] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd ed. Morgan Kaufmann, 2006.

[8] S. Deerwester, T. Dumais, W. Furnas, K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 319–407, Sep. 1990.

[9] Y. Yang and J. Pederson, "A comparative study on feature selection in text categorization," in *Proc. the 14th Int'l Conf. on Machine Learning*, Nashville, Tennessee, Jul. 1997, pp. 412–420.

[10] B. Liu, *Web Data Mining*. Springer, 2007.

[11] Y. Yang, "Expert network: Effective and efficient learning from human decisions in text categorization and retrieval," in *Proc. the 17th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, Dublin, Ireland, Jul. 1994, pp. 13–22.

[12] D. Wilson and T. Martinez, "An integrated instance-based learning algorithm," *Computational Intelligence*, vol. 16, no. 1, pp. 1–28, Feb. 200.

[13] E. Han, G. Karypis, , and V. Kumar, "Text categorization using weight adjusted k-nearest neighbor classification," in *Proc. the 5th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD)*, Hong Kong, China, Apr. 2001, pp. 53–65.

[14] E. Bingam and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *Proc. the 7th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, San Francisco, California, Aug. 2001, pp. 245–250.

[15] D. Cai, X. He, and J. Han, "Srda: An efficient algorithm for large-scale discriminant analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 1–12, Jan. 2008.

[16] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Proc. the 4th Int'l Conf. on Foundations of Data Organization and Algorithms*, Chicago, Illinois, Oct. 1993, pp. 69–84.

[17] C. Faloutsos, M. Ranganathan, , and Y. Manolopoulos, "Fast subsequence matching in time series databases," in *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, Minneapolis, Minnesota, Jun. 1994, pp. 419–429.

[18] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[19] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 107–144, 2007.

[20] Y. Moon, K. Whang, and W. Loh, "Duality-based subsequence matching in time-series databases," in *Proc. the 17th IEEE Int'l Conf. on Data Engineering*, Heidelberg, Germany, Apr. 2001, pp. 263–272.

[21] K. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary l_p norms," in *Proc. the 26th Int'l Conf. on Very Large Data Bases*, Berkeley, California, Sep. 2000, pp. 385–394.

[22] T. Yang and D. Lee, "T^3: On mapping text to time series," in *Proc. the 3rd Alberto Mendelzon Int'l Workshop on Foundations of Data Management*, Arequipa, Peru, May 2009.

[23] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Proc. AAAI 2002 Workshop on Artificial Intelligence for Web*, Austin, Texasa, Jul. 2000, pp. 58–64.

[24] D. Lewis, Y. Yang, T. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, Apr. 2004.

[25] K. Whang, M. Kim, and J. Lee, "Linear-time top-k sort," Korean Patent filed as Appl. No. 10-2011-0 037 332 (granted), U.S. Patent being applied.

[26] E. Horowitz, S. Sahni, and A.-F. S., *Fundamentals of Data Structures in C*, 2nd ed. W.H. Freeman, Sep. 1992.

[27] A. Asuncion and D. Newman. (2007) UCI machine learning repository. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[28] M. Berry, T. Do, G. O'Brien, V. Krishna, and S. Varadhan. (2007) SVDPACKC (version 1.0) user's guide. [Online]. Available: http://www.netlib.org/svdpack/index.html

[29] L. Larkey and W. Croft, "Combining classifiers in text categorization," in *Proc. the 19th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, Zurich, Switzerland, Aug. 1996, pp. 289–297.

[30] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, no. 1-2, pp. 69–90, Apr. 1999.