# Concept Decompositions for Large Sparse Text Data using Clustering

Inderjit S. Dhillon (inderjit@cs.utexas.edu)
*Department of Computer Science, University of Texas, Austin, TX 78712, USA*

Dharmendra S. Modha (dmodha@almaden.ibm.com)
*IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA*

Original Version: February 14, 1999
Revised Version: October 26, 2000

**Abstract.** Unlabeled document collections are becoming increasingly common and available; mining such data sets represents a major contemporary challenge. Using words as features, text documents are often represented as high-dimensional and sparse vectors–a few thousand dimensions and a sparsity of 95 to 99% is typical. In this paper, we study a certain *spherical k-means* algorithm for clustering such document vectors. The algorithm outputs $k$ disjoint clusters each with a *concept vector* that is the centroid of the cluster normalized to have unit Euclidean norm. As our first contribution, we empirically demonstrate that, owing to the high-dimensionality and sparsity of the text data, the clusters produced by the algorithm have a certain "fractal-like" and "self-similar" behavior. As our second contribution, we introduce *concept decompositions* to approximate the matrix of document vectors; these decompositions are obtained by taking the least-squares approximation onto the linear subspace spanned by all the concept vectors. We empirically establish that the approximation errors of the concept decompositions are close to the best possible, namely, to truncated singular value decompositions. As our third contribution, we show that the concept vectors are localized in the word space, are sparse, and tend towards orthonormality. In contrast, the singular vectors are global in the word space and are dense. Nonetheless, we observe the surprising fact that the linear subspaces spanned by the concept vectors and the leading singular vectors are quite close in the sense of small principal angles between them. In conclusion, the concept vectors produced by the spherical $k$-means algorithm constitute a powerful sparse and localized "basis" for text data sets.

**Keywords:** concept vectors, fractals, high-dimensional data, information retrieval, $k$-means algorithm, least-squares, principal angles, principal component analysis, self-similarity, singular value decomposition, sparsity, vector space models, text mining

## 1. Introduction

Large sets of text documents are now increasingly common. For example, the World-Wide-Web contains nearly 1 billion pages and is growing rapidly (www.alexa.com), the IBM Patent server consists of more than 2 million patents (www.patents.ibm.com), the Lexis-Nexis databases contain more than 2 billion documents (www.lexisnexis.com). Furthermore, an immense amount of text data exists on private corporate intranets, in archives of media companies, and in scientific and technical publishing houses. In this context, applying machine learning and statistical algorithms such as clustering, classification, principal component analysis, and discriminant analysis to text data sets is of great practical interest. In this paper, we focus on clustering of text data sets.

Clustering has been used to discover "latent concepts" in sets of unstructured text documents, and to summarize and label such collections. Clustering is inherently useful in organizing and searching large text collections, for example, in automatically building an ontology like Yahoo! (www.yahoo.com). Furthermore, clustering is useful for compactly summarizing, disambiguating, and navigating the results retrieved by a search engine such as AltaVista (www.altavista.com). Conceptual structure generated by clustering is akin to the "Table-of-Contents" in *front* of books, whereas an inverted index such as AltaVista is akin to the "Indices" at the *back* of books; both provide complementary information for navigating a large body of information. Finally, clustering is useful for personalized information delivery by providing a setup for routing new information such as that arriving from newsfeeds and new scientific publications. For experiments describing a certain syntactic clustering of the whole web and its applications, see (Broder et al., 1997). We have used clustering for visualizing and navigating collections of documents in (Dhillon et al., 1998). Various classical clustering algorithms such as the $k$-means algorithm and its variants, hierarchical agglomerative clustering, and graph-theoretic methods have been explored in the text mining literature; for detailed reviews, see (Rasmussen, 1992; Willet, 1988). Recently, there has been a flurry of activity in this area, see (Boley et al., 1998; Cutting et al., 1992; Hearst and Pedersen, 1996; Sahami et al., 1999; Schütze and Silverstein, 1997; Silverstein and Pedersen, 1997; Vaithyanathan and Dom, 1999; Zamir and Etzioni, 1998).

A starting point for applying clustering algorithms to unstructured text data is to create a *vector space model* for text data (Salton and McGill, 1983). The basic idea is (a) to extract unique content-bearing words from the set of documents and treat these words as *features* and (b) to represent each document as a vector of certain weighted word frequencies in this feature space. Observe that we may regard the vector space model of a text data set as a *word-by-document matrix* whose rows are words and columns are document vectors. Typically, a large number of words exist in even a moderately sized set of documents–a few thousand words or more are common. Hence, the document vectors are very *high-dimensional*. However, typically, most documents contain many fewer words, 1-5% or less, in comparison to the total number of words in the entire document collection. Hence, the document vectors are very *sparse*. Understanding and exploiting the structure and statistics of such vector space models is a major contemporary scientific and technological challenge.

We shall assume that the document vectors have been normalized to have unit $L^2$ norm, that is, they can be thought of as points on a high-dimensional unit sphere. Such normalization mitigates the effect of differing lengths of documents (Singhal et al., 1996). It is natural to measure "similarity" between such vectors by their inner product, known as *cosine similarity* (Salton and McGill, 1983). In this paper, we will use a variant of the well known "Euclidean" $k$-means algorithm (Duda and Hart, 1973; Hartigan, 1975) that uses cosine similarity (Rasmussen, 1992). We shall show that this algorithm partitions the high-dimensional unit sphere using a collection of great hypercircles, and hence we shall refer to this algorithm as the *spherical k-means* algorithm. The algorithm computes a disjoint partitioning of the document vectors, and, for each partition, computes a centroid normalized to have unit Euclidean norm. We shall demonstrate that these normalized centroids contain valuable semantic information about the clusters, and, hence, we refer to them as *concept vectors*. The spherical $k$-means algorithm has a number of advantages from a computational perspective: it can exploit the sparsity of the text data, it can be efficiently parallelized (Dhillon and Modha, 2000), and converges quickly (to a local maxima). Furthermore, from a statistical perspective, the algorithm generates concept vectors that serve as a "model" which may be used to classify future documents.

In this paper, our **first** focus is to study the structure of the clusters produced by the spherical $k$-means algorithm when applied to text data sets with the aim of gaining novel insights into the distribution of sparse text data in high-dimensional spaces. Such structural insights are a key step towards our **second** focus, which is to explore intimate connections between clustering using the spherical $k$-means algorithm and the problem of matrix approximation for the word-by-document matrices.

Generally speaking, matrix approximations attempt to retain the "signal" present in the vector space models, while discarding the "noise." Hence, they are extremely useful in improving the performance of information retrieval systems. Furthermore, matrix approximations are often used in practice for feature selection and dimensionality reduction prior to building a learning model such as a classifier. In a search/retrieval context, (Deerwester et al., 1990; Berry et al., 1995) have proposed *latent semantic indexing* (LSI) that uses truncated singular value decomposition (SVD) or principal component analysis to discover "latent" relationships between correlated words and documents. Truncated SVD is a popular and well studied matrix approximation scheme (Golub and Van Loan, 1996). Based on the earlier work of (O'Leary and Peleg, 1983) for image compression, (Kolda, 1997) has developed a memory efficient matrix approximation scheme known as semi-discrete decomposition. (Gallant, 1994; Caid and Oing, 1997) have used an "implicit" matrix approximation scheme based on their context vectors. (Papadimitriou et al., 1998) have proposed computationally efficient matrix approximations based on random projections. Finally, (Isbell and Viola, 1998) have used independent component analysis for identifying directions representing sets of highly correlated words, and have used these directions for an "implicit" matrix approximation scheme. As our title suggests, our main goal is to derive a new matrix approximation scheme using clustering.

We now briefly summarize our main contributions:

- In Section 3, we empirically examine the average intra- and inter-cluster structure of the partitions produced by the spherical $k$-means algorithm. We find that these clusters have a certain "fractal-like" and "self-similar" behavior that is not commonly found in low-dimensional data sets. These observations are important in that any proposed statistical model for text data should be consistent with these empirical constraints. As an aside, while claiming no such breakthrough, we would like to point out that the discovery of fractal nature of ethernet traffic has greatly impacted the design, control, and analysis of high-speed networks (Leland et al., 1994).

- In Section 4, we propose a new matrix approximation scheme–*concept decomposition*–that solves a least-squares problem after clustering, namely, computes the least-squares approximation onto the linear subspace spanned by the concept vectors. We empirically establish the surprising fact that the approximation power (when measured using the Frobenius norm) of concept decompositions is comparable to the best possible approximations by truncated SVDs (Golub and Van Loan, 1996). An important advantage of concept decompositions is that they are computationally more efficient and require much less memory than truncated SVDs.

- In Section 5, we show that our concept vectors are localized in the word space, are sparse, and tend towards orthonormality. In contrast, the singular vectors obtained from SVD are global in the word space and are dense. Nonetheless, we observe the surprising fact that the subspaces spanned by the concept vectors and the leading singular vectors are quite close in the sense of small principal angles (Björck and Golub,

1973) between them. Sparsity of the concept vectors is important in that it speaks to the economy or parsimony of the model constituted by them. Also, sparsity is crucial to computational and memory efficiency of the spherical $k$-means algorithm. In conclusion, the concept vectors produced by the spherical $k$-means algorithm constitute a powerful sparse and localized "basis" for text data sets.

Preliminary versions of this work were presented at the 1998 Irregular Conference held in Berkeley, CA (www.nersc.gov/conferences/irregular98/program.html), and at the 1999 SIAM Annual meeting held in Atlanta, GA (www.siam.org/meetings/an99/MS42.htm). Due to space considerations, we have removed some experimental results from this paper; complete details appear in our IBM Technical Report (Dhillon and Modha, 1999). Probabilistic latent semantic analysis (PLSA) of (Hofmann, 1999) views the word-by-document matrix as a co-occurrence table describing the probability that a word is related to a document, and approximates this matrix using the aspect model (Saul and Pereira, 1997). While similar in spirit, concept decompositions are distinct from PLSA. Our framework is *geometric* and is concerned with orthonormal $L^2$ projections, while PLSA is *probabilistic* and is concerned with statistical Kullback-Leibler projections. We examine the nature of large, sparse, high-dimensional text data, and find a certain fractal-like behavior (see Sections 3 and 5). On the other hand, PLSA uses a classical multinomial model to describe the statistical structure of a cluster. We employ the spherical $k$-means algorithm followed by a least-squares approximation step, while PLSA employs an EM-type algorithm. For mining extremely large text data sets, speed is of essence, and our spherical $k$-means+least-squares is generically faster than a corresponding EM-type algorithm. Finally, we explore document clustering, show that matrix approximation power of concept decompositions is close to the truncated SVDs, and compare and contrast concept vectors to singular vectors.

A word about notation: small-bold letters such as $\mathbf{x}$, $\mathbf{m}$, $\mathbf{c}$ will denote column vectors, capital-bold letters such as $\mathbf{X}$, $\mathbf{C}$, $\mathbf{Z}$, $\mathbf{R}$ will denote matrices, and script-bold letters such as $\mathcal{C}$ and $\mathcal{E}$ will denote linear subspaces. Also, $\|\mathbf{x}\|$ will denote the $L^2$ norm of a vector, $\mathbf{x}^T\mathbf{y}$ will denote the usual inner product or dot product of vectors, and, finally, $\|\mathbf{X}\|_F$ will denote the Frobenius norm of a matrix.

## 2. Vector Space Models for Text

In this section, we briefly review how to represent a set of unstructured text documents as a vector space model. The basic idea is to represent each document as a vector of certain weighted word frequencies. In addition, we introduce two text data sets that will be used throughout the paper to present our results.

### 2.1. Parsing and Preprocessing

1. Ignoring case, extract all unique words from the entire set of documents.

2. Eliminate non-content-bearing "stopwords" such as "a", "and", "the", etc. For sample lists of stopwords, see (Frakes and Baeza-Yates, 1992, Chapter 7).

3. For each document, count the number of occurrences of each word.

4. Using heuristic or information-theoretic criteria, eliminate non-content-bearing "high-frequency" and "low-frequency" words (Salton and McGill, 1983). Such words and the stopwords are both known as "function" words. Eliminating function words removes little or no information, while speeding up the computation. Although, in general, the criteria used for pruning function words are ad-hoc; for our purpose, any word that does not help in discriminating a cluster from its neighbors is a function word. We will see in Figures 7 and 12 that when the number of clusters is small, a large fraction of words can be treated as function words; we have selected the function words *independently* of the number of clusters.

5. After above elimination, suppose $d$ unique words remain. Assign a unique identifier between 1 and $d$ to each remaining word, and a unique identifier between 1 and $n$ to each document.

The above steps outline a simple preprocessing scheme. In addition, one may extract word phrases such as "New York," and one may reduce each word to its "root" or "stem", thus eliminating plurals, tenses, prefixes, and suffixes (Frakes and Baeza-Yates, 1992, Chapter 8). We point out, in passing, that an efficient implementation of the above scheme would use lexical analyzers, fast and scalable hash tables or other appropriate data structures.

## 2.2. Vector Space Model

The above preprocessing scheme yields the number of occurrences of word $j$ in document $i$, say, $f_{ji}$, and the number of documents which contain the word $j$, say, $d_j$. Using these counts, we now create $n$ document vectors in $R^d$, namely, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ as follows. For $1 \le j \le d$, set the $j$-th component of document vector $\mathbf{x}_i, 1 \le i \le n$, to be the product of three terms

$$x_{ji} = t_{ji} \times g_j \times s_i, \tag{1}$$

where $t_{ji}$ is the *term weighting component* and depends only on $f_{ji}$, $g_j$ is the *global weighting component* and depends on $d_j$, and $s_i$ is the *normalization component* for $\mathbf{x}_i$. Intuitively, $t_{ji}$ captures the relative importance of a word in a document, while $g_j$ captures the overall importance of a word in the entire set of documents. The objective of such weighting schemes is to enhance discrimination between various document vectors and to enhance retrieval effectiveness (Salton and Buckley, 1988).

There are many schemes for selecting the term, global, and normalization components, for example, (Kolda, 1997) presents 5, 5, and 2 schemes, respectively, for the term, global, and normalization components–a total of $5 \times 5 \times 2 = 50$ choices. From this extensive set, we will use two popular schemes denoted as `txn` and `tfn`, and known, respectively, as *normalized term frequency* and *normalized term frequency-inverse document frequency*. Both schemes emphasize words with higher frequencies, and use $t_{ji} = f_{ji}$. The `txn` scheme uses $g_j = 1$, while the `tfn` scheme emphasizes words with low overall collection frequency and uses $g_j = \log(n/d_j)$. In both schemes, each document vector is normalized to have unit $L^2$ norm, that is,

$$s_i = \left( \sum_{j=1}^{d} (t_{ji} g_j)^2 \right)^{-1/2}. \tag{2}$$

Intuitively, the effect of normalization is to retain only the *direction* of the document vectors. This ensures that documents dealing with the same subject matter (that is, using similar words), but differing in length lead to similar document vectors. For a comparative study of various document length normalization schemes, see (Singhal et al., 1996).

We now introduce two sets of text documents: "CLASSIC3" and "NSF".

**Example 1.A (CLASSIC3)** We obtained the CLASSIC3 data set containing 3893 documents by merging the popular MEDLINE, CISI, and CRANFIELD sets. MEDLINE consists of 1033 abstracts from medical journals, CISI consists of 1460 abstracts from information retrieval papers, and CRANFIELD consists of 1400 abstracts from aeronautical systems papers (ftp://ftp.cs.cornell.edu/pub/smart).

We preprocessed the CLASSIC3 collection by proceeding as in Section 2.1. After removing common stopwords, the collection contained 24574 unique words from which we eliminated 20471 low-frequency words appearing in less than 8 documents (roughly 0.2% of the documents), and 4 high-frequency words appearing in more than 585 documents (roughly 15% of the documents). We were finally left with 4099 words–still a very high-dimensional feature space. We created 3893 document vectors using the `txn` scheme. Each document vector has dimension 4099, however, on an average, each document vector contained only about 40 nonzero components and is more than 99% sparse.

**Example 2.A (NSF)** We obtained the NSF data set by downloading 13297 abstracts of the grants awarded by the National Science Foundation between March 1996 and August 1997 from www.nsf.gov. These grants included subjects such as astronomy, population studies, undergraduate education, materials, mathematics, biology and health, oceanography, computer science, and chemistry.

We preprocessed the NSF collection by proceeding as in Section 2.1. After removing common stopwords, the collection contained 66006 unique words from which we eliminated 60680 low-frequency words appearing in less than 26 documents (roughly 0.2% of the documents), and 28 high-frequency words appearing in more than 1994 documents (roughly 15% of the documents). We were finally left with 5298 words. We created 13297 document vectors using the `tfn` scheme. Each document vector has dimension 5298, however, on an average, each document vector contained only about 61 nonzero components and is roughly 99% sparse.

We stress that in addition to the `txn` and the `tfn` weighting schemes, we have conducted extensive experiments with a number of different schemes. Furthermore, we have also experimented with various cut-off thresholds other than 0.2% and 15% used above. In all cases, the essence of our empirical results has remained the same.

## 3. The Spherical $k$-means Algorithm

In this section, we study how to partition high-dimensional and sparse text data sets such as CLASSIC3 and NSF into disjoint conceptual categories. Towards this end, we briefly formalize the spherical $k$-means clustering algorithm. Moreover, we empirically study the structure of the clusters produced by the algorithm.

### 3.1. COSINE SIMILARITY

It follows from (1) and (2) that the document vectors $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ are points on the unit sphere in $R^d$. Furthermore, for most weighting schemes, all components of the document vectors are nonnegative, hence the document vectors are in fact in the nonnegative "orthant" of $R^d$, namely, $R^d_{\geq 0}$. For these vectors, the inner product is a natural measure of similarity. Given any two unit vectors $\mathbf{x}$ and $\mathbf{y}$ in $R^d_{\geq 0}$, let $0 \leq \theta(\mathbf{x}, \mathbf{y}) \leq \pi/2$ denote the angle between them; then

$$\mathbf{x}^T \mathbf{y} = \|\mathbf{x}\| \, \|\mathbf{y}\| \cos(\theta(\mathbf{x}, \mathbf{y})) = \cos(\theta(\mathbf{x}, \mathbf{y})).$$

Hence, the inner product $\mathbf{x}^T \mathbf{y}$ is often known as the "cosine similarity." Since cosine similarity is easy to interpret and simple to compute for sparse vectors, it is widely used in text mining and information retrieval (Frakes and Baeza-Yates, 1992; Salton and McGill, 1983).

### 3.2. CONCEPT VECTORS

Suppose we are given $n$ document vectors $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ in $R^d_{\geq 0}$. Let $\pi_1, \pi_2, \ldots, \pi_k$ denote a partitioning of the document vectors into $k$ disjoint clusters such that

$$\bigcup_{j=1}^{k} \pi_j = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \text{ and } \pi_j \cap \pi_\ell = \phi \text{ if } j \neq \ell.$$

For each fixed $1 \leq j \leq k$, the *mean vector* or the *centroid* of the document vectors contained in the cluster $\pi_j$ is

$$\mathbf{m}_j = \frac{1}{n_j} \sum_{\mathbf{x} \in \pi_j} \mathbf{x},$$

where $n_j$ is the number of document vectors in $\pi_j$. Note that the mean vector $\mathbf{m}_j$ need not have a unit norm; we can capture its direction by writing the corresponding *concept vector* as

$$\mathbf{c}_j = \frac{\mathbf{m}_j}{\|\mathbf{m}_j\|}.$$

The concept vector $\mathbf{c}_j$ has the following important property. For any unit vector $\mathbf{z}$ in $R^d$, we have from the Cauchy-Schwarz inequality that

$$\sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{z} \leq \sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{c}_j. \tag{3}$$

Thus, the concept vector may be thought of as the vector that is closest in cosine similarity (in an average sense) to all the document vectors in the cluster $\pi_j$.

### 3.3. THE OBJECTIVE FUNCTION

Motivated by (3), we measure the "coherence" or "quality" of each cluster $\pi_j$, $1 \leq j \leq k$, as

$$\sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{c}_j.$$

Observe that if all document vectors in a cluster are identical, then the average coherence of that cluster will have the highest possible value of 1. On the other hand, if the document vectors in a cluster vary widely, then the average coherence will be small, that is, close to 0. Since, $\sum_{\mathbf{x} \in \pi_j} \mathbf{x} = n_j \mathbf{m}_j$ and $\|\mathbf{c}_j\| = 1$, we have that

$$\sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{c}_j = n_j \mathbf{m_j}^T \mathbf{c}_j = n_j \|\mathbf{m_j}\| \mathbf{c_j}^T \mathbf{c_j} = n_j \|\mathbf{m_j}\| = \left\| \sum_{\mathbf{x} \in \pi_j} \mathbf{x} \right\|. \tag{4}$$

This rewriting yields the remarkably simple intuition that the quality of each cluster $\pi_j$ is measured by the $L^2$ norm of the sum of the document vectors in that cluster.

We measure the quality of any given partitioning $\{\pi_j\}_{j=1}^k$ using the following *objective function*:

$$\mathcal{Q}\left(\{\pi_j\}_{j=1}^k\right) = \sum_{j=1}^k \sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{c}_j, \tag{5}$$

Intuitively, the objective function measures the combined coherence of all the $k$ clusters. Such an objective function has also been proposed and studied theoretically in the context of market segmentation problems (Kleinberg et al., 1998).

### 3.4. SPHERICAL $k$-MEANS

We seek a partitioning of the document vectors $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ into $k$ disjoint clusters $\pi_1^\star, \pi_2^\star, \ldots, \pi_k^\star$ that maximizes the objective function in (5), that is, we seek a solution to the following maximization problem:

$$\{\pi_j^\star\}_{j=1}^k = \underset{\{\pi_j\}_{j=1}^k}{\arg\max}\, \mathcal{Q}\left(\{\pi_j\}_{j=1}^k\right). \tag{6}$$

Finding the optimal solution to the above maximization problem is NP-complete (Kleinberg et al., 1998, Theorem 3.1); also, see (Garey et al., 1982). We now discuss an approximation algorithm, namely, the *spherical k-means* algorithm, which is an effective and efficient iterative heuristic.

1. Start with an arbitrary partitioning of the document vectors, namely, $\{\pi_j^{(0)}\}_{j=1}^k$. Let $\{\mathbf{c}_j^{(0)}\}_{j=1}^k$ denote the concept vectors associated with the given partitioning. Set the index of iteration $t = 0$.

2. For each document vector $\mathbf{x}_i, 1 \le i \le n$, find the concept vector closest in cosine similarity to $\mathbf{x}_i$. Now, compute the new partitioning $\{\pi_j^{(t+1)}\}_{j=1}^k$ induced by the old concept vectors $\{\mathbf{c}_j^{(t)}\}_{j=1}^k$:

$$\pi_j^{(t+1)} = \left\{ \mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^n : \mathbf{x}^T \mathbf{c}_j^{(t)} > \mathbf{x}^T \mathbf{c}_\ell^{(t)}, 1 \le \ell \le n, \ell \ne j \right\}, 1 \le j \le k. \tag{7}$$

   In words, $\pi_j^{(t+1)}$ is the set of all document vectors that are closest to the concept vector $\mathbf{c}_j^{(t)}$. If it happens that some document vector is simultaneously closest to more than one concept vector, then it is randomly assigned to one of the clusters. Clusters defined using (7) are known as *Voronoi* or *Dirichlet* partitions.

3. Compute the new concept vectors corresponding to the partitioning computed in (7):

$$\mathbf{c}_j^{(t+1)} = \mathbf{m}_j^{(t+1)} / \|\mathbf{m}_j^{(t+1)}\|, \; 1 \le j \le k, \tag{8}$$

   where $\mathbf{m}_j^{(t+1)}$ denotes the centroid or the mean of the document vectors in cluster $\pi_j^{(t+1)}$.

4. If some "stopping criterion" is met, then set $\pi_j^\dagger = \pi_j^{(t+1)}$ and set $\mathbf{c}_j^\dagger = \mathbf{c}_j^{(t+1)}$ for $1 \le j \le k$, and exit. Otherwise, increment $t$ by 1, and go to step 2 above.

An example of a stopping criterion is: Stop if

$$\left| \mathcal{Q}\left(\{\pi_j^{(t)}\}_{j=1}^k\right) - \mathcal{Q}\left(\{\pi_j^{(t+1)}\}_{j=1}^k\right) \right| \le \varepsilon$$

for some suitably chosen $\varepsilon > 0$. In words, stop if the "change" in objective function after an iteration of the algorithm is less than a certain threshold. We now establish that the spherical $k$-means algorithm outlined above never decreases the value of the objective function.

**Lemma 3.1** *For every $t \ge 0$, we have that*

$$\mathcal{Q}\left(\{\pi_j^{(t)}\}_{j=1}^k\right) \le \mathcal{Q}\left(\{\pi_j^{(t+1)}\}_{j=1}^k\right).$$

**Proof:**

$$\begin{aligned}
\mathcal{Q}\left(\{\pi_j^{(t)}\}_{j=1}^k\right) = \sum_{j=1}^k \left( \sum_{\mathbf{x} \in \pi_j^{(t)}} \mathbf{x}^T \mathbf{c}_j^{(t)} \right) &= \sum_{j=1}^k \left( \sum_{\ell=1}^k \sum_{\mathbf{x} \in \pi_j^{(t)} \cap \pi_\ell^{(t+1)}} \mathbf{x}^T \mathbf{c}_j^{(t)} \right) \\
&\le \sum_{j=1}^k \left( \sum_{\ell=1}^k \sum_{\mathbf{x} \in \pi_j^{(t)} \cap \pi_\ell^{(t+1)}} \mathbf{x}^T \mathbf{c}_\ell^{(t)} \right) \\
&= \sum_{\ell=1}^k \left( \sum_{j=1}^k \sum_{\mathbf{x} \in \pi_j^{(t)} \cap \pi_\ell^{(t+1)}} \mathbf{x}^T \mathbf{c}_\ell^{(t)} \right) \\
&= \sum_{\ell=1}^k \sum_{\mathbf{x} \in \pi_\ell^{(t+1)}} \mathbf{x}^T \mathbf{c}_\ell^{(t)} \\
&\le \sum_{\ell=1}^k \sum_{\mathbf{x} \in \pi_\ell^{(t+1)}} \mathbf{x}^T \mathbf{c}_\ell^{(t+1)} = \mathcal{Q}\left(\{\pi_j^{(t+1)}\}_{j=1}^k\right),
\end{aligned}$$

where the first inequality follows from (7) and the second inequality follows from (3). □

Intuitively, the above proof says that the spherical $k$-means algorithm exploits a duality between the concept vectors and partitioning: the concept vectors $\{\mathbf{c}_j^{(t)}\}_{j=1}^k$ induce a partitioning $\{\pi_j^{(t+1)}\}_{j=1}^k$ which in turn implies better concept vectors $\{\mathbf{c}_j^{(t+1)}\}_{j=1}^k$.

**Corollary 3.1** *The following limit exists:*

$$\lim_{t\to\infty} \mathcal{Q}\left(\{\pi_j^{(t)}\}_{j=1}^k\right).$$

**Proof:** We have from Lemma 3.1 that the sequence of numbers $\left\{\mathcal{Q}\left(\{\pi_j^{(t)}\}_{j=1}^k\right)\right\}_{t\geq 0}$ is increasing. Furthermore, for every $t \geq 0$, it follows from (4) and the triangle inequality that

$$\mathcal{Q}\left(\{\pi_j^{(t)}\}_{j=1}^k\right) = \sum_{j=1}^k n_j^{(t)} \|\mathbf{m}_j^{(t)}\| \leq \sum_{j=1}^k n_j^{(t)} = n.$$

Thus, we have an increasing sequence of numbers that is bounded from above by a constant. Hence, the sequence converges, and the limit exists. □

Corollary 3.1 says that if the spherical $k$-means algorithm is iterated indefinitely, then the value of the objective function will eventually converge. However, it is important to realize that the corollary does not imply that the underlying partitioning $\{\pi_j^{(t)}\}_{j=1}^k$ converges. We refer the reader interested in more general convergence results to (Pollard, 1982; Sabin and Gray, 1986).

The spherical $k$-means algorithm (like other gradient ascent schemes) is prone to local maximas. Nonetheless, the algorithm yielded reasonable results for the experimental results reported in this paper. A key to the algorithm is a careful selection of the starting partitioning $\{\pi_j^{(0)}\}_{j=1}^k$, for example, (a) one may randomly assign each document to one of the $k$ clusters or (b) one may first compute the concept vector for the entire document collection and obtain $k$ starting concept vectors by randomly perturbing this vector, and use the Voronoi partitioning corresponding to this set of $k$ concept vectors. Furthermore, one can try several initial partitionings and select the best (in terms of the largest objective function) amongst these trials. In this paper, we tried exactly one initial partitioning according to the strategy (b) above.

### 3.5. Experimental Results

Before we undertake an empirical study of the spherical $k$-means algorithm, we present the following example to persuade the reader that the algorithm indeed produces meaningful clusters.

**Example 1.B (confusion matrix)** We clustered the CLASSIC3 data set into $k = 3$ clusters. The following "confusion matrix" shows that the clusters $\pi_1^\dagger$, $\pi_2^\dagger$, and $\pi_3^\dagger$ produced by the algorithm can be reasonably identified with the MEDLINE, CISI, and CRANFIELD data sets, respectively.

| | $\pi_1^\dagger$ | $\pi_2^\dagger$ | $\pi_3^\dagger$ |
|---|---|---|---|
| MEDLINE | 1004 | 18 | 11 |
| CISI | 5 | 1440 | 15 |
| CRANFIELD | 4 | 16 | 1380 |

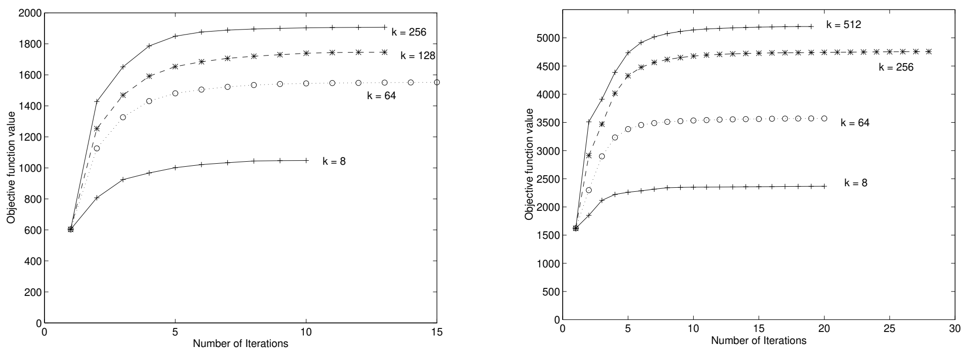



*Figure 1.* Value of the objective function $Q$ versus the number of $k$-means iterations for the CLASSIC3 (left panel) and the NSF (right panel) data sets.

We can conclude from the above table that the algorithm indeed "discovers" the class structure underlying the CLASSIC3 data set. Also, see Figure 7 for the top ten words describing each of the three clusters $\pi_1^\dagger$, $\pi_2^\dagger$, and $\pi_3^\dagger$.

**Example 2.B** We clustered the NSF data set into $k = 10$ clusters. See Figure 12 for the top seven words corresponding to four of the ten clusters. For the top seven words corresponding to the remaining clusters, see (Dhillon and Modha, 1999). These words constitute an anecdotal evidence of the coherence of the clusters produced by the algorithm.

We now empirically validate Lemma 3.1 and Corollary 3.1.

**Example 1.C (objective function never decreases)** We clustered the CLASSIC3 data set into $k = 8$, 64, 128, and 256 clusters. For each clustering, in Figure 1 (left panel), we plot the value of the objective function versus the number of $k$-means iterations. It can be seen from the figure that for a fixed $k$, as the number of iterations increases, the value of the objective never decreases, and, in fact, quickly converges. Furthermore, we can see that the larger the number of clusters $k$ the larger the value of the objective function.

**Example 2.C (objective function never decreases)** We repeat Example 1.C for the NSF data set in Figure 1 (right panel).

In low dimensions, we may picture a cluster of data points as a "nice" and "round" cloud of points with the mean in the center. We now show that such intuitions do not directly carry over to high-dimensional sparse text data sets. Specifically, we examine the intra- and inter-cluster structure produced by the spherical $k$-means algorithm.

Suppose that we have clustered the document vectors into $k$ clusters $\{\pi_j^\dagger\}_{j=1}^k$, and let $\{\mathbf{c}_j^\dagger\}_{j=1}^k$ and $\{n_j^\dagger\}_{j=1}^k$, respectively, denote the corresponding concept vectors and the number of document vectors. We can obtain an insight into the structure of a cluster by the distribution of the cosine similarities within the cluster. We do this as follows. For $\pi_j^\dagger$, compute the $n_j^\dagger$ numbers:

$$\mathcal{A}_j = \left\{ \mathbf{x}^T \mathbf{c}_j^\dagger : \mathbf{x} \in \pi_j^\dagger \right\}.$$

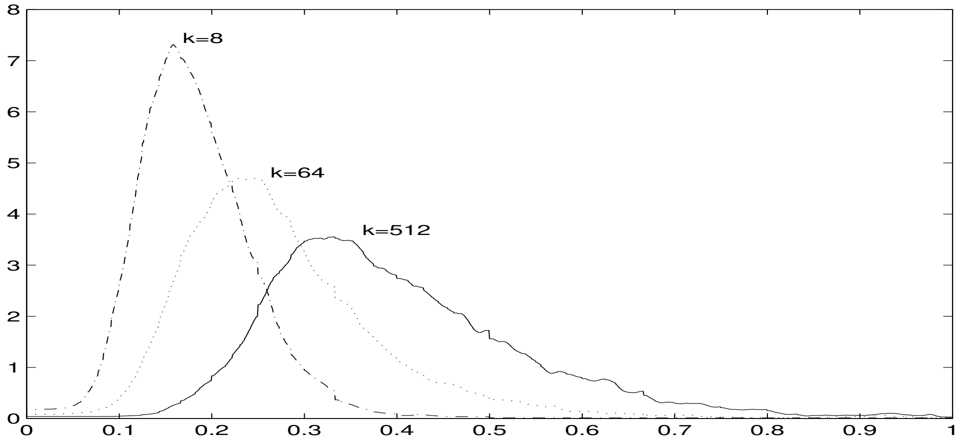



*Figure 2.* Average *intra*-cluster estimated probability density functions for $k = 8$, 64, and 512 clusters of the NSF data set.

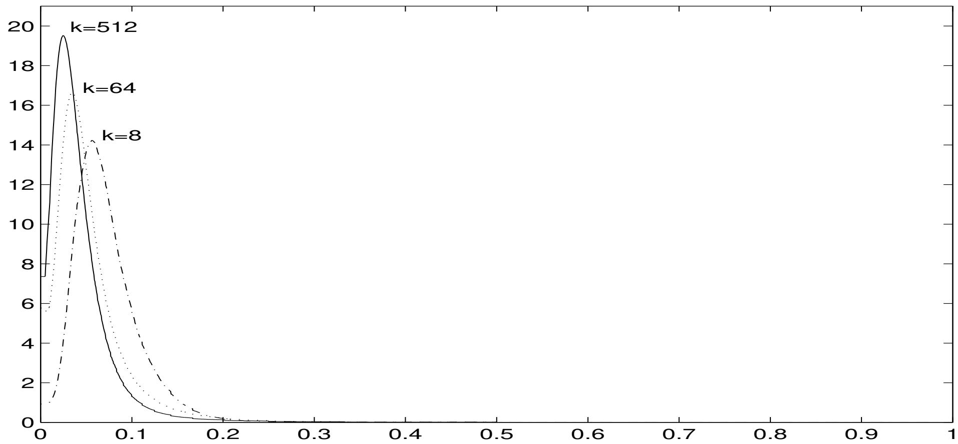



*Figure 3.* Average *inter*-cluster estimated probability density functions for $k = 8$, 64, and 512 clusters of the NSF data set.

Since usually there are many clusters, we can obtain an insight into the *average intra-cluster structure* by computing the $n$ numbers:

$$\bigcup_{j=1}^{k} \mathcal{A}_j \tag{9}$$

and by plotting an estimated probability density function (pdf) of these numbers. We recall that a pdf compactly and completely captures the entire statistics underlying a set of data points. Also, a pdf is a nonnegative function with unit area. In this paper, we estimate the pdf of a set of one-dimensional data points using the unweighted mixture of histogram method of (Rissanen et al., 1992).

**Example 2.D (Average intra-cluster structure)** We clustered the NSF data set into $k = 8$, 64, and 512 clusters. In Figure 2, we plot the estimated pdfs of the $n = 13297$ numbers in (9) for these three clusterings. Note that the range of the $x$-axis is $[0,1]$ since we are plotting cosine similarities. For $k = 8$, the majority of the mass of the estimated pdf lies in the interval $[0,0.4]$; hence, it follows that *there are virtually no document vectors even close to the corresponding concept vector*! Consequently, the concept vectors can hardly be pictured as surrounded by data points, in fact, there is a large empty space between the document vectors and the corresponding concept vectors. Now, observe that as $k$ is increased from 8 to 64 and then to 512, the mass of the estimated pdfs progressively shifts to the right towards 1, that is, with increasing $k$ the clusters become more coherent, and document vectors become progressively closer to the corresponding concept vectors. As $k$ increases, the empty space around the concept vectors progressively shrinks.

In spite of the above behavior, as Examples 1.B and 2.B show, the algorithm does tend to find meaningful clusters. We reconcile these facts by studying the *average inter-cluster structure*, that is, by computing the cosine similarity between a document vector and concept vectors corresponding to all clusters that *do not contain* the document. We compute the following $n(k-1)$ numbers:

$$\bigcup_{j=1}^{k} \left\{ \mathbf{x}^T \mathbf{c}_j^\dagger : \mathbf{x} \notin \pi_j^\dagger \right\}, \tag{10}$$

and plot an estimated pdf of these numbers.

**Example 2.E (Average inter-cluster structure)** We use the same three clusterings with $k = 8$, 64, and 512 as in Example 2.D. In Figure 3, we plot the estimated pdfs for $13297 \times 7$, $13297 \times 63$, and $13297 \times 511$ numbers in (10) corresponding to $k = 8$, 64, and 512, respectively. For $k = 8$, by comparing the inter-cluster estimated pdf in Figure 3 to the intra-cluster estimated pdf in Figure 2, we can see that the corresponding peaks are separated from each other and that the former assigns more mass towards 0 and the latter assigns more mass towards 1. Thus, although the document vectors within a cluster are not that close to the corresponding concept vector, the document vectors outside the cluster are even further. In a sense, it is this relative difference that makes it possible to meaningfully cluster sparse data. By comparing the intra- and inter-cluster estimated pdfs for $k = 8$, 64, and 512 in Figures 2 and 3, we can see that, as $k$ increases, the overlap between corresponding inter- and intra-cluster estimated pdfs steadily decreases. Furthermore, as $k$ is increased from 8 to 64 and then to 512, the mass of the estimated inter-cluster pdfs in Figure 3 progressively shifts to the left towards 0.

Examples 2.D and 2.E show that the clusters of high-dimensional sparse data have properties not commonly observed in low-dimensional data sets. In light of these examples, it follows that modeling the multidimensional distributions of document vectors within a cluster is likely to be a daunting problem, for example, a Gaussian distribution around the mean is hardly a tenable model. We used $k = 8 \times 1$, $64 = 8 \times 8$, and $512 = 8 \times 64$ in Examples 2.D and 2.E to illustrate that average intra- and inter-cluster structures exhibit the same behavior at various scalings or resolutions of the number of clusters. The only essential difference between these structures is the progressive movement of the intra-cluster pdfs towards 1 and the corresponding movement of the inter-cluster pdfs towards 0. We believe that these facts point to a certain "fractal-like" or "self-similar" nature of high-dimensional sparse text data that needs to be further studied.

### 3.6. Comparison with the Euclidean $k$-means Algorithms

We now point out the difference between our objective function (5) and the following more traditional objective function (Duda and Hart, 1973; Hartigan, 1975):

$$\sum_{j=1}^{k} \sum_{\mathbf{x} \in \pi_j} \|\mathbf{x} - \mathbf{m}_j\|^2, \tag{11}$$

where $\mathbf{m}_j$ is the mean vector associated with the cluster $\pi_j$. The objective function in (11) measures the sum of the squared Euclidean distances between document vectors and the closest mean vectors. The above objective function can be minimized using the well known Euclidean $k$-means algorithm (Duda and Hart, 1973). The Euclidean $k$-means algorithm is very similar to the spherical $k$-means algorithm. It can be obtained by replacing the partitioning (7) by:

$$\pi_j^{(t+1)} = \left\{ \mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^n : \|\mathbf{x} - \mathbf{m}_j^{(t)}\|^2 < \|\mathbf{x} - \mathbf{m}_\ell^{(t)}\|^2, 1 \le \ell \le n \right\}, 1 \le j \le k. \tag{12}$$

and by computing the mean vector $\mathbf{m}_j^{(t)}$ in step (3) instead of the concept vector $\mathbf{c}_j^{(t)} = \mathbf{m}_j^{(t)} / \|\mathbf{m}_j^{(t)}\|$.

We now contrast the cluster boundaries corresponding to the objective functions in (5) and (11). By (7), the boundary between any two clusters associated with the spherical $k$-means algorithm, say, $\pi_j$ and $\pi_\ell$, is the locus of all points satisfying:

$$\mathbf{x}^T(\mathbf{c}_j - \mathbf{c}_\ell) = 0.$$

This locus is a hyperplane passing through the origin. The intersection of such a hyperplane with the unit sphere is a great hypercircle. Thus, the spherical $k$-means algorithm partitions the unit sphere using a collection of great hypercircles. Similarly, by (12), the boundary between any two clusters associated with the Euclidean $k$-means algorithm, say, $\pi_j$ and $\pi_\ell$, is the locus of all points satisfying:

$$\mathbf{x}^T(\mathbf{m}_j - \mathbf{m}_\ell) = \frac{1}{2}(\mathbf{m}_j^T \mathbf{m}_j - \mathbf{m}_\ell^T \mathbf{m}_\ell).$$

This locus is a hyperplane whose intersection with the unit sphere will not, in general, be a great hypercircle. Since we use cosine similarity as a measure of closeness it is more natural to partition the document vectors using great hypercircles, and hence, we use the objective function $\mathcal{Q}$.

Observe that we can write the objective function $\mathcal{Q}$ in (5) equivalently as

$$\mathcal{Q}\left(\{\pi_j\}_{j=1}^k\right) = \sum_{j=1}^{k} \sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{c}_j = 2n - 2\sum_{j=1}^{k} \sum_{\mathbf{x} \in \pi_j} \|\mathbf{x} - \mathbf{c}_j\|^2 \equiv 2n - 2\mathcal{F}\left(\{\pi_j\}_{j=1}^k\right). \tag{13}$$

Hence, it follows that the maximization problem in (6) is equivalent to the following minimization problem:

$$\{\pi_j^\star\}_{j=1}^k = \arg\min_{\{\pi_j\}_{j=1}^k} \mathcal{F}\left(\{\pi_j\}_{j=1}^k\right). \tag{14}$$

In the next section, we show that (14) can be naturally thought of as a matrix approximation problem.

## 4. Matrix Approximations using Clustering

Given $n$ document vectors $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ in $R^d$, we define a $d \times n$ *word-by-document matrix* as

$$\mathbf{X} = [\mathbf{x}_1\, \mathbf{x}_2\, \ldots\, \mathbf{x}_n],$$

that is, as a matrix whose columns are the document vectors. The spherical $k$-means algorithm is designed to cluster document vectors into disjoint partitions. It is not explicitly designed to approximate the word-by-document matrix. Nonetheless, there is a surprising and natural connection between clustering and matrix approximation that we explore in this section.

### 4.1. Clustering as Matrix Approximation

Given any partitioning $\{\pi_j\}_{j=1}^k$ of the document vectors $\{\mathbf{x}_i\}_{i=1}^n$ into $k$ clusters, we can approximate a document vector by the closest concept vector. In other words, if a document vector is in cluster $\pi_j$, we can approximate it by the concept vector $\mathbf{c}_j$. Thus, we can define a $d \times n$ matrix approximation $\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_k(\{\pi_j\}_{j=1}^k)$ such that, for $1 \le i \le n$, its $i$-th column is the concept vector closest to the document vector $\mathbf{x}_i$.

A natural question is: How effective is the matrix $\hat{\mathbf{X}}_k$ in approximating the matrix $\mathbf{X}$? We measure the error in approximating $\mathbf{X}$ by $\hat{\mathbf{X}}_k$ using the squared Frobenius norm of the difference matrix:

$$||\mathbf{X} - \hat{\mathbf{X}}_k||_F^2,$$

where for any $p \times q$ matrix $\mathbf{A} = [a_{ij}]$, its *Frobenius norm* (Golub and Van Loan, 1996) is defined as

$$||\mathbf{A}||_F = \sqrt{\sum_{i=1}^p \sum_{j=1}^q |\, a_{ij}\,|^2}.$$

Observe that we can write the matrix approximation error as

$$||\mathbf{X} - \hat{\mathbf{X}}_k||_F^2 = \sum_{j=1}^k \sum_{\mathbf{X} \in \pi_j} ||\mathbf{x} - \mathbf{c}_j||^2 = \mathcal{F}\left(\{\pi_j\}_{j=1}^k\right). \tag{15}$$

It follows from (13), (14), and (15) that maximizing the objective function $\mathcal{Q}$ can also be thought of as a constrained matrix approximation problem.

The matrix approximation $\hat{\mathbf{X}}_k$ has rank at most $k$. In particular, the matrix approximation $\hat{\mathbf{X}}_k^\dagger$ corresponding to the final partitioning $\{\pi_j^\dagger\}_{j=1}^k$ has rank at most $k$. In the next subsection, we compare the approximation power of $\hat{\mathbf{X}}_k^\dagger$ to that of the best possible rank-$k$ approximation to the word-by-document matrix $\mathbf{X}$.

### 4.2. Matrix Approximation using Singular Value Decompositions

Singular value decomposition (SVD) has been widely studied in information retrieval and text mining, for example, in latent semantic indexing (Berry et al., 1995; Deerwester et al., 1990). Let $r$ denote the rank of the $d \times n$ word-by-document matrix $\mathbf{X}$. Following (Golub and Van Loan, 1996), we define the SVD of $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{U}\blacksquare\mathbf{V}^T,$$

where $\mathbf{U}$ is the $d \times r$ orthogonal matrix of *left-singular* vectors, $\mathbf{V}$ is the $n \times r$ orthogonal matrix of *right-singular* vectors, and $\blacksquare$ is the $r \times r$ diagonal matrix of positive *singular* values $(\sigma_1, \sigma_2, \ldots, \sigma_r)$ arranged in decreasing order of their magnitude. Note that we can write

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\blacksquare^2\mathbf{U}^T, \ \ \mathbf{X}^T\mathbf{X} = \mathbf{V}\blacksquare^2\mathbf{V}^T.$$

Hence, the columns of $\mathbf{U}$ and $\mathbf{V}$ define the orthonormal eigenvectors associated with the $r$ nonzero eigenvalues of $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$, respectively, while the diagonal elements of $\blacksquare$ are nonnegative square roots of the nonzero eigenvalues of $\mathbf{X}\mathbf{X}^T$ and of $\mathbf{X}^T\mathbf{X}$. In statistical terminology, the columns of $\mathbf{U}$ are known as principal components[1].

For $1 \le k \le r$, let $\mathbf{U}_k$ and $\mathbf{V}_k$ be obtained by deleting the last $(r-k)$ columns, respectively, from $\mathbf{U}$ and $\mathbf{V}$, and let $\blacksquare_{\mathbf{k}}$ be obtained by deleting the last $(r-k)$ rows and columns of $\blacksquare$. The $d \times n$ matrix

$$\bar{\mathbf{X}}_k = \mathbf{U}_k\blacksquare_k\mathbf{V}_k^T = \mathbf{U}_k(\mathbf{U}_k^T\mathbf{U}_k)^{-1}\mathbf{U}_k^T\mathbf{X} = \mathbf{U}_k\mathbf{U}_k^T\mathbf{X} \tag{16}$$

is known as the *k-truncated SVD* of the matrix $\mathbf{X}$, and has rank equal to $k$. As written above, $\bar{\mathbf{X}}_k$ can be thought of as a least-squares approximation of the matrix $\mathbf{X}$ onto the column space of the matrix $\mathbf{U}_k$.

**Example 1.D (singular values)** In Figure 4 (left panel), we plot the largest 256 singular values of the CLASSIC3 word-by-document matrix.

**Example 2.F (singular values)** In Figure 4 (right panel), we plot the largest 235 singular values of the NSF word-by-document matrix.

We mention that all experiments which report results on SVDs including the above examples use Michael Berry's SVDPACKC (www.netlib.org/svdpack/index.html). It can be seen from Figure 4 that the singular values decrease continuously without any obvious sudden drop. Hence, there is no natural cut-off point; in practice, for various information retrieval experiments, a truncation value of $k$ between 100-300 is normally used.

The following well known result (Golub and Van Loan, 1996) establishes that the $k$-truncated SVD is the best rank-$k$ approximation to $\mathbf{X}$ in the squared Frobenius norm. Hence, $k$-truncated SVDs are the baseline against which all other rank-$k$ matrix approximations should be measured.

[1] Precisely speaking, to obtain principal components we should subtract the mean $(1/n)\sum_{i=1}^{n}\mathbf{x}_i$ from every document vector.

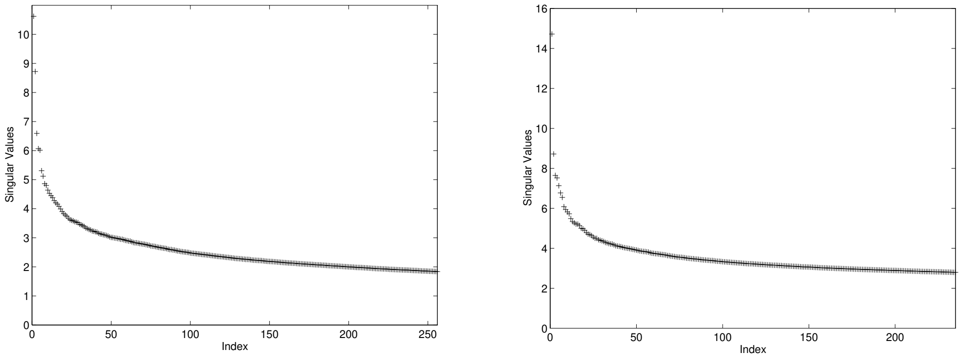



*Figure 4.* The largest 256 and 235 singular values of the CLASSIC3 (left panel) and the NSF (right panel) word-by-document matrices.

**Lemma 4.1** *For any $d \times n$ matrix $\mathbf{Y}$ such that rank$(\mathbf{Y}) \leq k \leq r$, the following holds:*

$$\sum_{\ell=k+1}^{r} \sigma_\ell^2 = \|\mathbf{X} - \bar{\mathbf{X}}_k\|_F^2 \leq \|\mathbf{X} - \mathbf{Y}\|_F^2.$$

Lemma 4.1 holds for arbitrary matrices $\mathbf{Y}$ with rank less than or equal to $k$. In particular, it holds for the matrix approximation $\hat{X}_k^\dagger$ corresponding to the final partitioning $\{\pi_j^\dagger\}_{j=1}^k$ produced by the spherical $k$-means algorithm, that is,

$$\sum_{\ell=k+1}^{r} \sigma_\ell^2 \leq \|\mathbf{X} - \hat{\mathbf{X}}_k^\dagger\|_F^2.$$

We now empirically validate and examine this lower bound.

**Example 1.E (comparing $\bar{\mathbf{X}}_k$ and $\hat{\mathbf{X}}_k^\dagger$)** In Figure 5 (left panel), we compare the errors in approximating the CLASSIC3 word-by-document matrix using the $k$-truncated SVDs and the matrices $\hat{\mathbf{X}}_k^\dagger$ for various values of $k$. It can be seen that, for each fixed $k$, the approximation error for the $k$-truncated SVD is significantly lower than that for $\hat{\mathbf{X}}_k^\dagger$.

**Example 2.G (comparing $\bar{\mathbf{X}}_k$ and $\hat{\mathbf{X}}_k^\dagger$)** In Figure 5 (right panel), we repeat Example 1.E for the NSF data set.

**Remark 4.1** Observe that for the NSF data set, in Figure 5 (right panel), we do not give results for more than 235 singular vectors. In trying to compute more singular vectors, the subroutine lasq2() from SVDPACKC ran out of memory on our workstation–an IBM/RS6000 with 256 MBytes of memory. On the other hand, we could easily compute 512 concept vectors for the NSF data set.

It follows from Figure 5 that the matrix approximation $\hat{\mathbf{X}}_k^\dagger$ is very bad! With hindsight, this is to be expected, since $\hat{\mathbf{X}}_k^\dagger$ uses a naive strategy of approximating each document vector

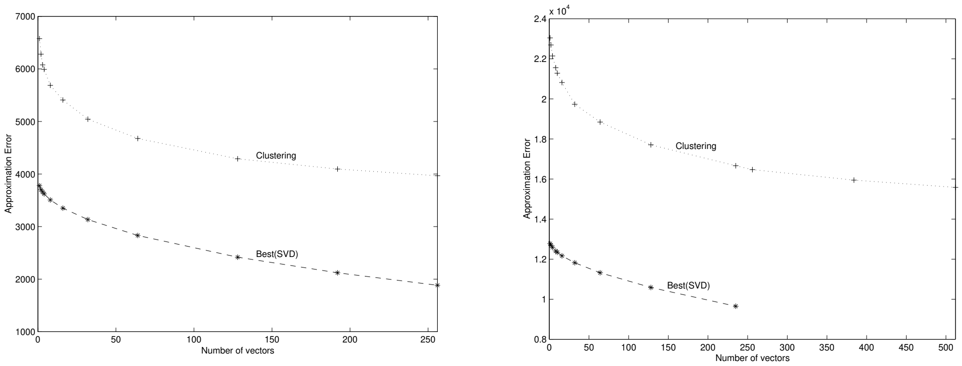



*Figure 5.* Comparing the approximation errors $\|\mathbf{X} - \bar{\mathbf{X}}_k\|_F^2$ and $\|\mathbf{X} - \hat{\mathbf{X}}_k^{\dagger}\|_F^2$ for the CLASSIC3 (left panel) and the NSF (right panel) data sets for various values of $k$.

by the closest concept vector. Hence, there is a significant room for seeking better rank-$k$ matrix approximations.

### 4.3. Concept Decompositions

We now show that by approximating each document vector by a linear combination of the concept vectors it is possible to obtain significantly better matrix approximations.

Let $\{\pi_j\}_{j=1}^k$ denote a partitioning of the document vectors $\{\mathbf{x}_i\}_{i=1}^n$ into $k$ clusters. Let $\{\mathbf{c}_j\}_{j=1}^k$ denote the corresponding concept vectors. Define the *concept matrix* as a $d \times k$ matrix such that, for $1 \le j \le k$, the $j$-th column of the matrix is the concept vector $\mathbf{c}_j$, that is,

$$\mathbf{C}_k = [\mathbf{c}_1\, \mathbf{c}_2\, \ldots\, \mathbf{c}_k].$$

Assuming linear independence of the $k$ concept vectors, it follows that the concept matrix has rank $k$.

For any partitioning of the document vectors, we define the corresponding *concept decomposition* $\tilde{\mathbf{X}}_k$ of the word-by-document matrix $\mathbf{X}$ as the least-squares approximation of $\mathbf{X}$ onto the column space of the concept matrix $\mathbf{C}_k$. We can write the concept decomposition as a $d \times n$ matrix

$$\tilde{\mathbf{X}}_k = \mathbf{C}_k \mathbf{Z}^\star,$$

where $\mathbf{Z}^\star$ is a $k \times n$ matrix that is to be determined by solving the following least-squares problem:

$$\mathbf{Z}^\star = \arg\min_Z \|\mathbf{X} - \mathbf{C}_k \mathbf{Z}\|_F^2. \tag{17}$$

It is well known that a closed-form solution exists for the least-squares problem (17), namely,

$$\mathbf{Z}^\star = (\mathbf{C}_k^T \mathbf{C}_k)^{-1} \mathbf{C}_k^T \mathbf{X}.$$

Although the above equation is intuitively pleasing, it does not constitute an efficient and numerically stable way to compute the matrix $\mathbf{Z}^\star$. Computationally, we use the QR decomposition of the concept matrix (Golub and Van Loan, 1996). The following lemma establishes that the concept decomposition $\tilde{\mathbf{X}}_k$ is a better matrix approximation than $\hat{\mathbf{X}}_k$.

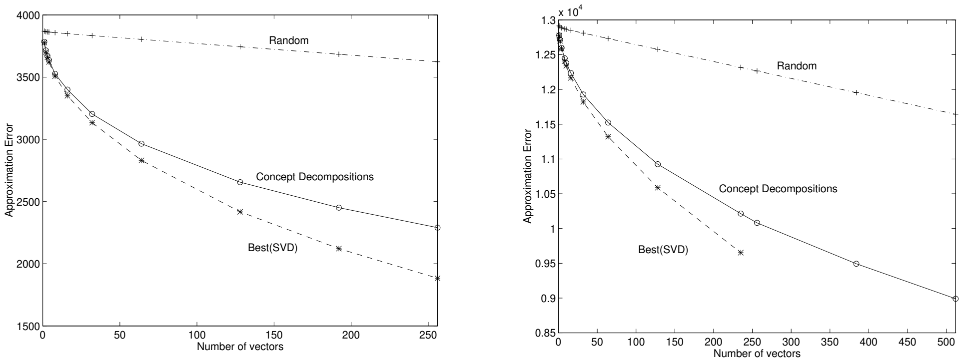



*Figure 6.* Comparing the approximation errors $||\mathbf{X}-\bar{\mathbf{X}}_k||_F^2$, $||\mathbf{X}-\tilde{\mathbf{X}}_k^\dagger||_F^2$, and $||\mathbf{X}-\check{\mathbf{X}}_k||_F^2$ for the CLASSIC3 (left panel) and the NSF (right panel) data sets for various values of $k$.

**Lemma 4.2**

$$\sum_{\ell=k+1}^{r} \sigma_\ell^2 \overset{(a)}{\leq} ||\mathbf{X}-\tilde{\mathbf{X}}_k||_F^2 \overset{(b)}{\leq} ||\mathbf{X}-\hat{\mathbf{X}}_k||_F^2.$$

**Proof:** Since the matrix approximation $\tilde{\mathbf{X}}_k$ has rank $k$, the inequality $(a)$ follows from Lemma 4.1. Now, observe that we may write

$$\hat{\mathbf{X}}_k = \mathbf{C}_k\mathbf{P}_k$$

where $\mathbf{P}_k = [p_{ji}]$ is a $k \times n$ matrix such $p_{ji} = 1$ if the document vector $\mathbf{x}_i$ is in the cluster $\pi_j$ and $p_{ji} = 0$ otherwise. Hence, by (17), the inequality $(b)$ follows. $\square$

Lemma 4.2 holds for any partitioning, in particular, it holds for the concept decomposition

$$\tilde{\mathbf{X}}_k^\dagger = \mathbf{C}_k^\dagger \left[(\mathbf{C}_k^\dagger)^T\mathbf{C}_k^\dagger\right]^{-1}(\mathbf{C}_k^\dagger)^T\mathbf{X}$$

corresponding to the final partitioning $\{\pi_j^\dagger\}_{j=1}^k$ produced by the spherical $k$-means algorithm. We next show that the approximations $\tilde{\mathbf{X}}_k^\dagger$ turn out to be quite powerful; but, before that we introduce "random" matrix approximations.

Let $\mathbf{R}_k$ denote a $d \times k$ matrix whose entries are randomly generated using a uniform distribution on $[0,1]$. For our experiments, we used the rand function of MATLAB. Assuming that the columns of $\mathbf{R}_k$ are linearly independent, we can write the least-squares approximation of $\mathbf{X}$ onto the column space of $\mathbf{R}_k$ as

$$\check{\mathbf{X}}_k = \mathbf{R}_k(\mathbf{R}_k^T\mathbf{R}_k)^{-1}\mathbf{R}_k^T\mathbf{X}.$$

**Example 1.F (comparing $\bar{\mathbf{X}}_k$, $\tilde{\mathbf{X}}_k^\dagger$, and $\check{\mathbf{X}}_k$)** In Figure 6 (left panel), we compare the errors in approximating the CLASSIC3 word-by-document matrix using the $k$-truncated SVDs, the matrices $\hat{\mathbf{X}}_k^\dagger$, and random matrix approximations for various values of $k$. It can be seen from the figure that the approximation errors attained by concept decompositions, $||\mathbf{X} -$

$\tilde{\mathbf{X}}_k^\dagger\|_F^2$, are quite close to those attained by the optimal, $\|\mathbf{X} - \bar{\mathbf{X}}_k\|_F^2$. In comparison, the random matrix approximations are much worse.

**Example 2.H (comparing $\bar{\mathbf{X}}_k$, $\tilde{\mathbf{X}}_k^\dagger$, and $\check{\mathbf{X}}_k$)** In Figure 6 (right panel), we repeat Example 1.F for the NSF data set.

## 5. Concept Vectors and Singular vectors: A comparison

In Section 4, we demonstrated that concept vectors may be used to obtain matrix approximations, namely, concept decompositions, that are comparable in quality to the SVD. In this section, we compare and contrast the two basis sets: (a) concept vectors and (b) singular vectors (columns of $U$ in (16)).

### 5.1. Concept Vectors are Local and Sparse

As before, let $\{\pi_j^\dagger\}_{j=1}^k$ denote a partitioning of the document vectors into $k$ disjoint clusters. For $1 \leq j \leq k$, we now associate a *word cluster* $W_j$ with the document cluster $\pi_j$ as follows. A word $1 \leq w \leq d$ is contained in $W_j$, if the weight of that word in $\mathbf{c}_j$ is larger than the weight of that word in any other concept vector $\mathbf{c}_\ell, 1 \leq \ell \leq k, \ell \neq j$. Precisely, we define

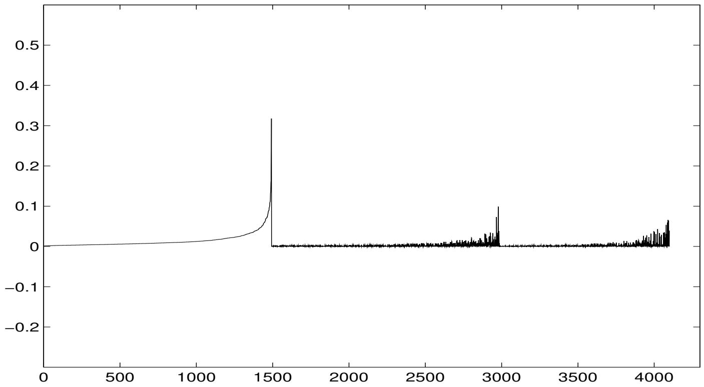
$$W_j = \{w : 1 \leq w \leq d,\ \mathbf{c}_{wj} > \mathbf{c}_{w\ell}, 1 \leq \ell \leq k, \ell \neq j\}.$$

The following examples show that the word clusters can be used for "labeling" a cluster of document vectors.

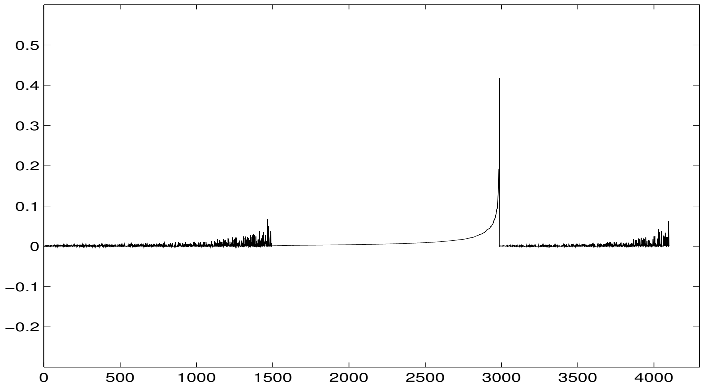
**Example 1.G (word clusters)** Here, we use the same clustering of the CLASSIC3 data set with $k = 3$ as in Example 1.B. On the right hand side of Figure 7 we list the top ten words from each of the three corresponding word clusters. It is clear that each word cluster is essentially localized to only one of the three underlying concepts: MEDLINE, CISI, and CRANFIELD. In contrast, as seen in Figure 8, the top ten words for the three leading singular vectors are distributed across all the three underlying concepts.

**Example 2.I (word clusters)** For this example, we clustered the NSF data set into $k = 10$ clusters. In Figure 12, we display four of the ten concept vectors, and, in Figure 13, we display the four leading singular vectors. Each of the concept vectors and the singular vectors is annotated with top seven words. For plots of the remaining concept vectors and the next 6 singular vectors, see (Dhillon and Modha, 1999).

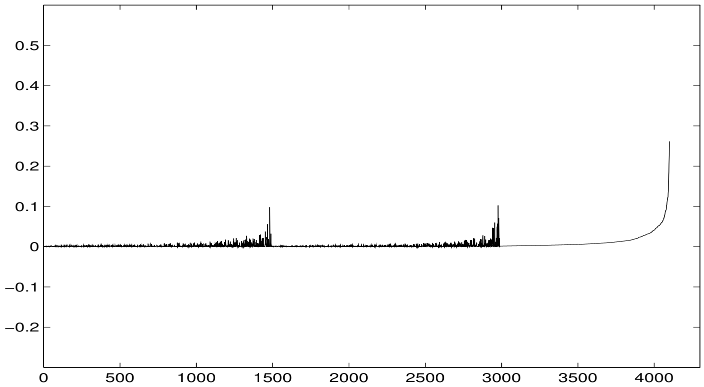
We now define a *total order* on the $d$ words as follows. For $1 \leq j \leq k$, we order the words in the $W_j$ in the increasing order of their respective weight in $\mathbf{c}_j$. Next, we impose an arbitrary order on the word clusters themselves. For example, order the word clusters such that words in $W_j$ precede words in $W_\ell$, if $j < \ell$. We now use such total orders to illustrate the locality of the concept vectors.




| | |
|---|---|
| patients | 0.3178 |
| cells | 0.2429 |
| treatment | 0.1617 |
| normal | 0.1614 |
| growth | 0.1489 |
| blood | 0.1376 |
| children | 0.1310 |
| disease | 0.1151 |
| cell | 0.1120 |
| human | 0.1104 |

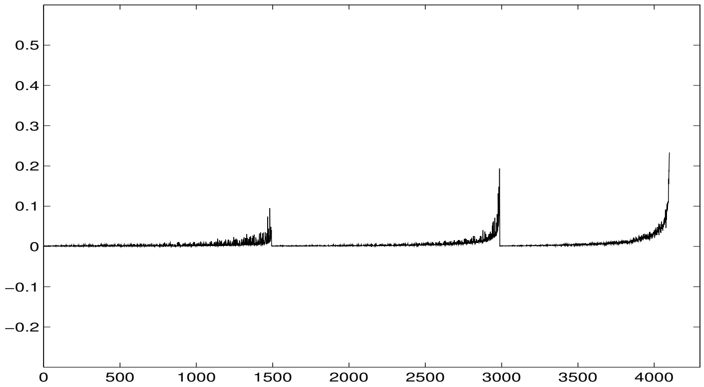



| | |
|---|---|
| library | 0.4171 |
| research | 0.2108 |
| libraries | 0.2009 |
| retrieval | 0.1921 |
| data | 0.1918 |
| systems | 0.1794 |
| science | 0.1637 |
| scientific | 0.1524 |
| book | 0.1345 |
| study | 0.1291 |

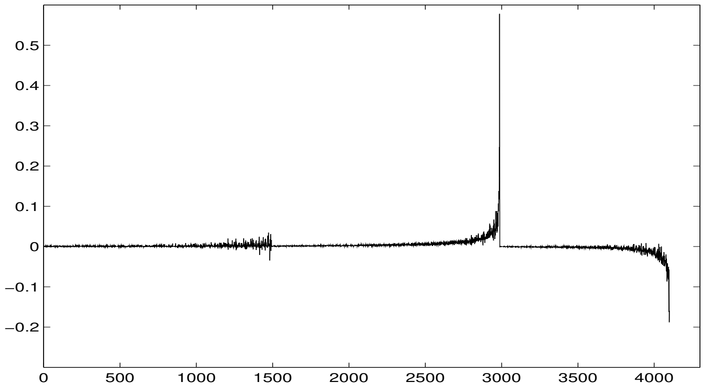



| | |
|---|---|
| pressure | 0.2616 |
| boundary | 0.2263 |
| theory | 0.2046 |
| layer | 0.1846 |
| mach | 0.1790 |
| method | 0.1651 |
| supersonic | 0.1426 |
| heat | 0.1361 |
| shock | 0.1325 |
| effects | 0.1228 |

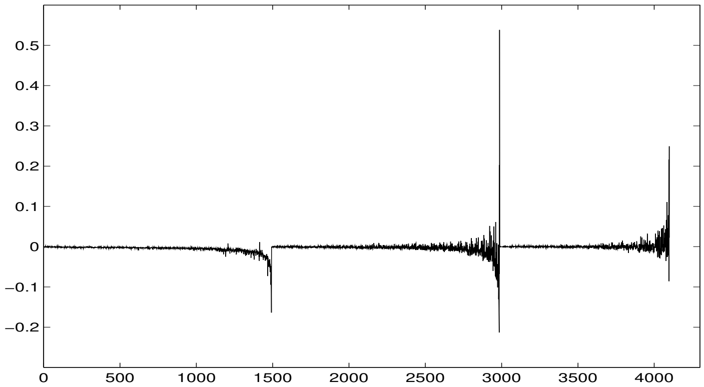
*Figure 7.* The three concept vectors corresponding to a clustering of the CLASSIC3 data set into 3 clusters. For each concept vector, the top ten words with the corresponding weights are shown on the right.




| | |
|---|---|
| pressure | +0.2335 |
| boundary | +0.2234 |
| library | +0.1938 |
| layer | +0.1894 |
| theory | +0.1862 |
| method | +0.1689 |
| mach | +0.1548 |
| data | +0.1484 |
| analysis | +0.1318 |
| shock | +0.1117 |




| | |
|---|---|
| library | +0.5780 |
| libraries | +0.2464 |
| boundary | −0.1879 |
| research | +0.1730 |
| layer | −0.1620 |
| pressure | −0.1597 |
| systems | +0.1379 |
| data | +0.1200 |
| retrieval | +0.1180 |
| science | +0.1144 |




| | |
|---|---|
| library | +0.5384 |
| boundary | +0.2491 |
| layer | +0.2163 |
| retrieval | −0.2132 |
| libraries | +0.2023 |
| patients | −0.1638 |
| data | −0.1534 |
| scientific | −0.1307 |
| systems | −0.1294 |
| cells | −0.1152 |

*Figure 8.* The three leading singular vectors for the CLASSIC3 word-by-document matrix. For each singular vector, the top ten words with the corresponding weights are shown on the right.

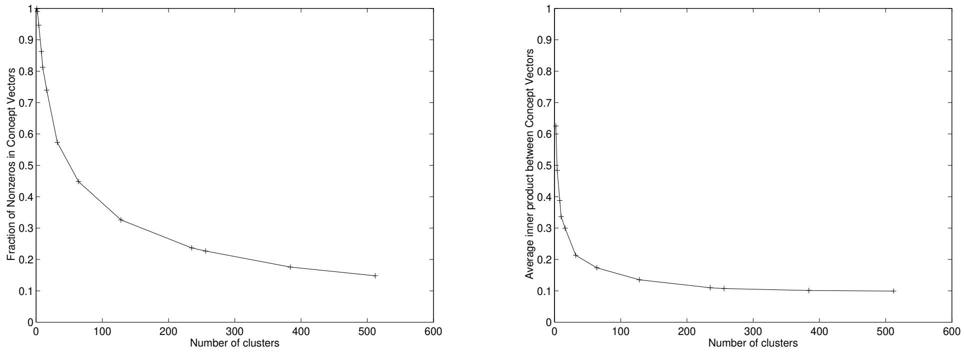



*Figure 9.* The sparsity of the NSF concept matrix $\mathbf{C}_k^\dagger$ (left panel), and the average inner product between concept vectors for the NSF data set (right panel).

**Example 1.H (locality)** Using the same clustering of the CLASSIC3 data set into $k = 3$ clusters as in Example 1.G, in Figure 7, we plot the three concept vectors. For these plots we used the total order on words described above; the boundaries of the three word clusters are evident in the figure and so is the ordering of the words within a word cluster. Figure 7 shows that most of the weight of a concept vector is concentrated in or localized to the corresponding word cluster. Analogously, in Figure 8, we plot the leading three singular vectors of the CLASSIC3 word-by-document matrix by using the same total order on the words. In contrast to the concept vectors, the singular vectors distribute their weight across all the three word clusters. *Thus, the concept vectors are localized, while singular vectors are global in nature. Intuitively speaking, the concept vectors can be compared to wavelets, while the singular vectors can be compared to Fourier series.* Finally, observe that the concept vectors are always nonnegative, whereas the singular vectors can assume both positive and negative values.

**Example 2.J (locality)** We repeat Example 1.H for the NSF data set with $k = 10$ clusters in Figures 12 and 13. We leave the task of comparing and contrasting these figures to the reader as an exercise.

**Example 2.K (sparsity)** In Figure 9 (left panel), we plot the ratio of the number of nonzero entries in all $k$ concept vectors to the total number of entries, namely, $5298 \times k$, for various values of $k$. *We see that as k increases the concept vectors become progressively sparser.* For example, for $k = 512$, the concept vectors are roughly 85% sparse. *In contrast, the singular vectors are virtually completely dense.*

Intuitively, as the number of clusters increases, there are fewer document vectors within each cluster (in an average sense). Recall that the document vector are almost 99% sparse, consequently, the concept vectors are sparse.

For any $k \geq 2$, we write the average inner product between the concept vectors $\{\mathbf{c}_j^\dagger\}_{j=1}^k$ as

$$\frac{2}{k(k-1)} \sum_{j=1}^{k} \sum_{\ell=j+1}^{k} (\mathbf{c}_j^\dagger)^T \mathbf{c}_\ell^\dagger. \tag{18}$$

The average inner product takes a value between $[0,1]$, where a value of 0 corresponds to orthonormal concept vectors and a value of 1 corresponds to identical concept vectors.

**Example 2.L (orthonormality)** For the NSF data set, in Figure 9 (right panel), we plot the average inner product given in (18) versus the number of clusters $k$. As the number of clusters $k$ increases, we see that the average inner product between the concept vectors progressively moves towards 0. *Hence, the concept vectors tend towards "orthonormality." In contrast, the singular vectors are orthonormal.*

So far, we have contrasted concept vectors and singular vectors. Nonetheless, the next subsection shows the surprising fact that the subspaces spanned by these vectors are in fact quite close.

### 5.2. Principal Angles: Comparing Concept and Singular Subspaces

Given $k$ concept vectors $\{\mathbf{c}_k^\dagger\}_{j=1}^k$ corresponding to the final partitioning produced by the $k$-means algorithm, define the corresponding *concept subspace* as

$$\mathcal{C}_k^\dagger = \text{span}\{\mathbf{c}_1^\dagger, \mathbf{c}_2^\dagger, \ldots, \mathbf{c}_k^\dagger\}.$$

Now, for $1 \leq k \leq n$, define the *singular subspace* $\mathcal{S}_k$ as the $k$-dimensional linear subspace spanned by the leading $k$ singular vectors, that is,

$$\mathcal{S}_k = \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}.$$

The closeness of the subspaces $\mathcal{C}_k^\dagger$ and $\mathcal{S}_k$ can be measured by quantities known as principal angles (Björck and Golub, 1973; Golub and Van Loan, 1996). Intuitively, principal angles generalize the notion of an angle between two lines to higher-dimensional subspaces of $R^d$.

Let $\mathcal{F}$ and $\mathcal{G}$ be given subspaces of $R^d$. Assume that

$$p = dim(\mathcal{F}) \geq dim(\mathcal{G}) = q \geq 1.$$

The $q$ *principal angles* $\theta_\ell \in [0, \pi/2]$ between $\mathcal{F}$ and $\mathcal{G}$ are recursively defined for $\ell = 1, 2, \cdots, q$ by

$$\cos\theta_\ell = \max_{\mathbf{f}\in\mathcal{F}} \max_{\mathbf{g}\in\mathcal{G}} \mathbf{f}^T\mathbf{g},$$

subject to the constraints: $\|\mathbf{f}\| = 1, \|\mathbf{g}\| = 1, \mathbf{f}^T\mathbf{f}_j = 0, \mathbf{g}^T\mathbf{g}_j = 0, j = 1, 2, \cdots, \ell-1$. The vectors $(\mathbf{f}_1, \cdots, \mathbf{f}_q)$ and $(\mathbf{g}_1, \cdots, \mathbf{g}_q)$ are called *principal vectors* of the pair of subspaces. Intuitively, $\theta_1$ is the angle between two closest unit vectors $\mathbf{f}_1 \in \mathcal{F}$ and $\mathbf{g}_1 \in \mathcal{G}$, $\theta_2$ is the angle between two closest unit vectors $\mathbf{f}_2 \in \mathcal{F}$ and $\mathbf{g}_2 \in \mathcal{G}$ such that $\mathbf{f}_2$ and $\mathbf{g}_2$ are, respectively, orthogonal to $\mathbf{f}_1$ and $\mathbf{g}_1$, and so on. Write the *average cosine of the principal angles* between the subspaces $\mathcal{F}$ and $\mathcal{G}$ as $(1/q)\sum_{\ell=1}^q \cos\theta_\ell$. See (Björck and Golub, 1973) for an algorithm to compute the principal angles.

**Example 1.I (principal angles)** In the following table, we compare the singular subspace $\mathcal{S}_3$ with various concept subspaces $\mathcal{C}_k^\dagger$ for the CLASSIC3 data set.

| | $\cos\theta_1$ | $\cos\theta_2$ | $\cos\theta_3$ |
|---|---|---|---|
| $\mathcal{C}_3^\dagger$ | 0.996 | 0.968 | 0.433 |
| $\mathcal{C}_4^\dagger$ | 0.996 | 0.989 | 0.557 |
| $\mathcal{C}_8^\dagger$ | 0.997 | 0.992 | 0.978 |
| $\mathcal{C}_{16}^\dagger$ | 0.997 | 0.994 | 0.990 |

Observe that, as $k$ increases, the cosines of all the principal angles tend to 1. In fact, for $k = 16$ the singular subspace $\mathcal{S}_3$ is essentially completely contained in the concept subspace, and even for $k = 3$ the two subspaces virtually share a common two-dimensional subspace spanned by the leading two principal vectors.

**Example 2.M (principal angles)** In the following table, we compare the singular subspace $\mathcal{S}_{10}$ with various concept subspaces $\mathcal{C}_k^\dagger$ for the NSF data set.

| | $\cos\theta_1$ | $\cos\theta_2$ | $\cos\theta_3$ | $\cos\theta_4$ | $\cos\theta_5$ | $\cos\theta_6$ | $\cos\theta_7$ | $\cos\theta_8$ | $\cos\theta_9$ | $\cos\theta_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{C}_{10}^\dagger$ | 0.999 | 0.990 | 0.985 | 0.980 | 0.967 | 0.943 | 0.921 | 0.806 | 0.400 | 0.012 |
| $\mathcal{C}_{16}^\dagger$ | 0.999 | 0.991 | 0.985 | 0.981 | 0.978 | 0.963 | 0.954 | 0.903 | 0.844 | 0.377 |
| $\mathcal{C}_{32}^\dagger$ | 0.999 | 0.993 | 0.990 | 0.985 | 0.984 | 0.976 | 0.974 | 0.973 | 0.952 | 0.942 |

Again, as $k$ increases, the cosines of all the principal angles tend to 1.

We are not interested in comparing *individual* singular subspaces to *individual* concept subspaces, rather we are interested in comparing the *sequence* of singular subspaces $\{\mathcal{S}_k\}_{k\geq 1}$ to the *sequence* of concept subspaces $\{\mathcal{C}_k^\dagger\}_{k\geq 1}$. Since it is hard to directly compare the sequences, in the following example, we compare a fixed singular subspace to various concept subspaces. For results comparing a fixed concept subspace to various singular subspaces, see (Dhillon and Modha, 1999).

**Example 2.N (comparing a singular subspace to various concept subspaces)** For the NSF data set, in Figure 10, we plot the average cosine of the principal angles (left panel) and the cosines of all the principal angles (right panel) between the singular subspace $\mathcal{S}_{64}$ and various concept subspaces. Note that for $k = 512$ the average cosine between $\mathcal{S}_{64}$ and $\mathcal{C}_k^\dagger$ is greater than 0.95 (left panel), and almost all the cosines are greater than 0.9 (right panel). In Figure 11, we furnish similar plots for the singular subspace $\mathcal{S}_{235}$ and various concept subspaces.

In closing, even though individually the concept vectors are very different from the singular vectors, concept subspaces turn out to be quite close to singular subspaces. This result is rather surprising, especially in very high-dimensions.

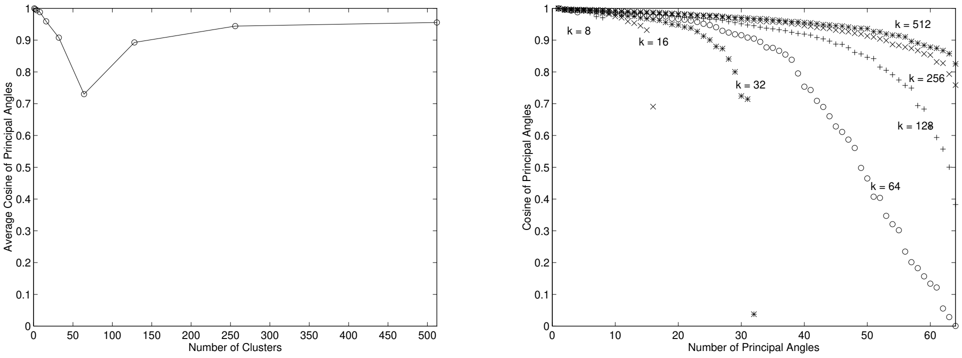



*Figure 10.* Average cosine (left panel) and cosines (right panel) of the principal angles between $\mathcal{S}_{64}$ and various concept subspaces for the NSF data set.

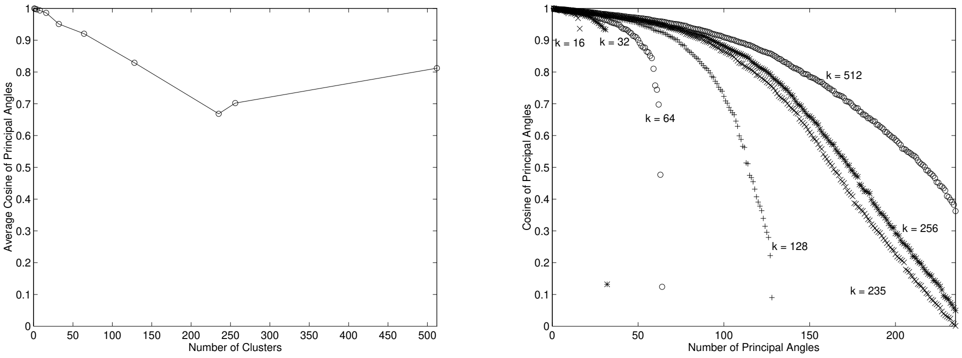



*Figure 11.* Average cosine (left panel) and cosines (righ panel) of the principal angles between $\mathcal{S}_{235}$ and various concept subspaces for the NSF data set.

## 6. Conclusions

In this paper, we have studied vector space models of large document collections. These models are very high-dimensional and sparse, and present unique computational and statistical challenges not commonly encountered in low-dimensional dense data.

Clustering is an invaluable tool to organize a vector space model and the associated document collection. We have used the fast spherical $k$-means clustering algorithm to produce meaningful clusters with good, descriptive labels (see Figures 7 and 12). Geometrically, the spherical $k$-means clustering algorithm partitions the high-dimensional space into Voronoi or Dirichlet regions separated by hyperplanes passing through the origin. Alongwith each cluster, we associate a *concept vector* that provides a compact summary of the cluster.

The spherical $k$-means algorithm seeks high-coherence clusters. We found average cluster coherence to be quite low, that is, in the high-dimensional space there is a large void surrounding each concept vector (see Example 2.D). This behavior is uncommon for low-dimensional, dense data sets; for example, think of the distribution of a Gaussian cloud around its mean. If the ties between document vectors and the nearest concept vectors are so weak, then a natural question is: why is clustering of such data even possible?

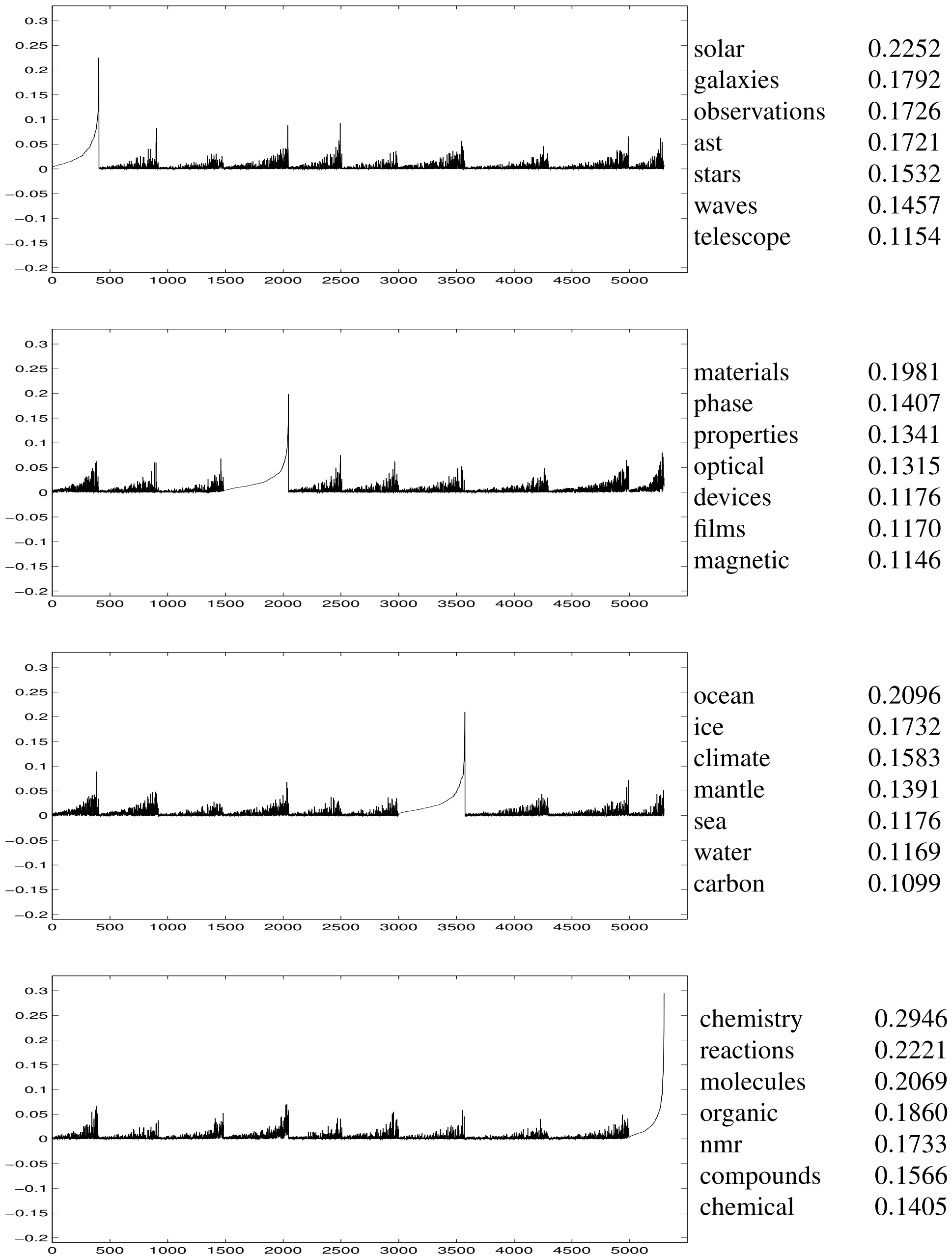



*Figure 12.* Four concept vectors corresponding to a clustering of the NSF data set into 10 clusters. For each concept vector, the top seven words with the corresponding weights are shown on the right.

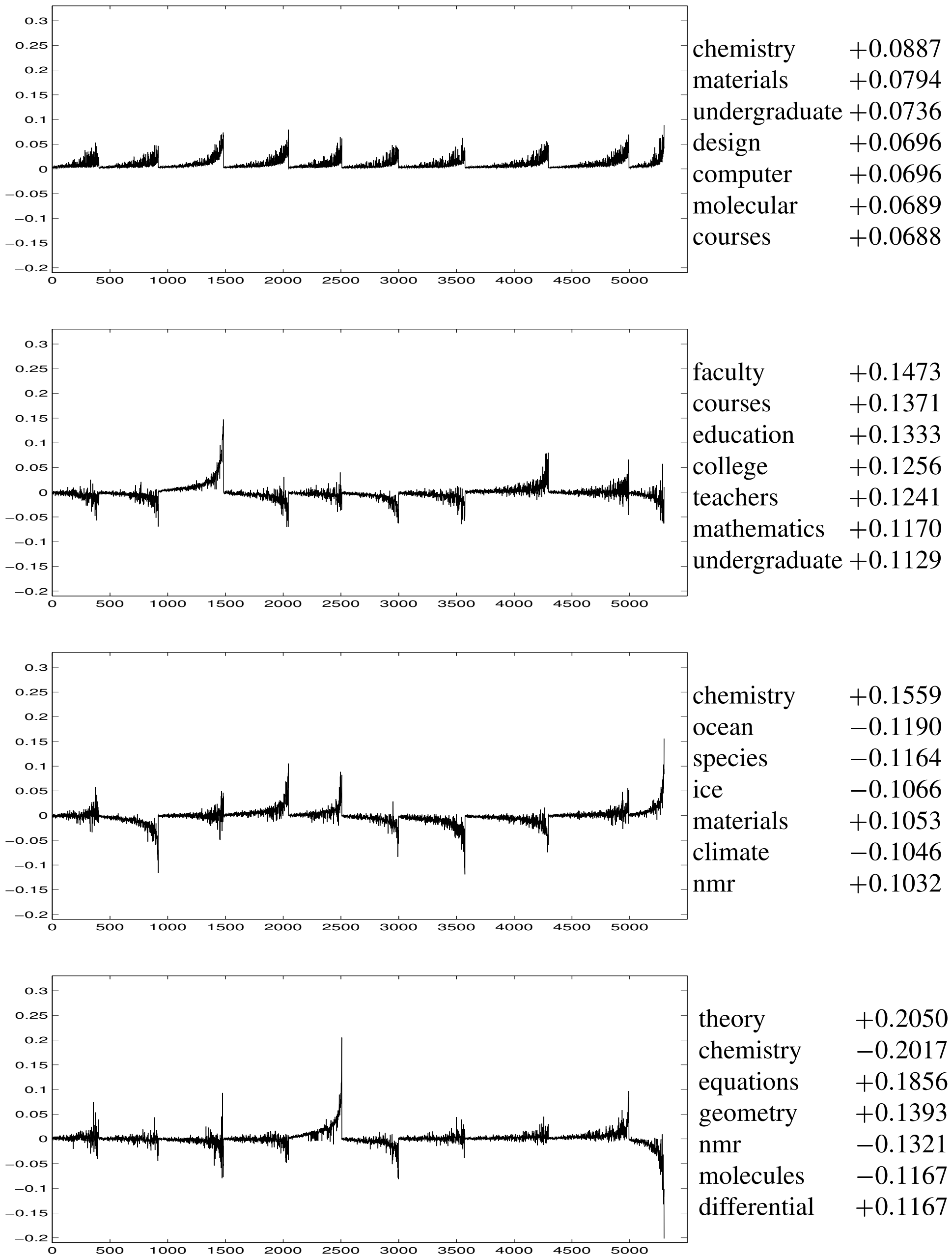



*Figure 13.* The leading four singular vectors for the NSF word-by-document matrix. For each singular vector, the top seven words with the corresponding weights are shown on the right.

We reconcile this paradox by observing that document vectors are indeed close to their nearest concept vector; not in an *absolute sense* but *relative to* their distances from the other concept vectors.

Furthermore, we found the average intra- and inter-cluster structures to be similar at various resolutions. The only essential difference is the progressive separation of the intra- from inter-cluster structure. This prompts an obvious analogy with fractals (Mandelbrot, 1988). Thus, any proposed statistical model for text data should be consistent with this fractal behavior. In fact, it might be meaningful to seek maximum entropy distributions subject to such empirical constraints. Further evidence of self-similarity is provided by our concept vector plots (Figures 7 and 12) that demonstrate that word counts within and outside of each cluster have the same general distribution. It is well known that word count distributions in text collections obey a certain Zipf's law (Zipf, 1949). Our results suggest the possibility that Zipf's law may hold in a recursive fashion, that is, for each cluster within a collection, and for each sub-cluster within a cluster, and so on.

One of our main findings is that concept decompositions that are derived from concept vectors can be used for the more basic task of matrix approximation. Surprisingly, the approximation power of concept decompositions is comparable to that of truncated SVDs (Figure 6). Furthermore, the subspaces spanned by concept vectors are quite close to the subspaces spanned by the singular vectors (Figures 10 and 11). The SVD computation is known to have large time and memory requirements. Thus, our faster and memory-efficient concept decompositions can profitably replace SVDs in many applications such as dimensionality reduction, feature selection, and improved information retrieval.

In spite of the fact that both the concept decompositions and the truncated SVDs possess similar approximation power, their constituent concept vectors and singular vectors are quite unlike each other. In particular, the concept vectors are localized in the word space, while the singular vectors are global in nature (Figures 7, 8, 12, and 13). The locality of concept vectors is extremely useful in labeling (in a human intelligible fashion) the "latent concepts" discovered by the algorithm, for example, see the words on the right-hand side panels in Figures 7 and 12. Furthermore, unlike singular vectors, the concept vectors are sparse (Figure 9), and, hence, constitute a more compact description of the data. In conclusion, the concept vectors constitute a powerful sparse and localized "basis" for text data sets. We cannot resist the temptation to compare the concept vectors to wavelets and the singular vectors to Fourier series.

## References

Berry, M. W., S. T. Dumais, and G. W. O'Brien: 1995, 'Using linear algebra for intelligent information retrieval'. *SIAM Review* **37**(4), 573–595.

Björck, A. and G. Golub: 1973, 'Numerical methods for computing angles between linear subspaces'. *Mathematics of Computation* **27**(123).

Boley, D., M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore: 1998, 'Document categorization and query generation on the World Wide Web using WebACE'. *AI Review*.

Broder, A. Z., S. C. Glassman, M. S. Manasse, and G. Zweig: 1997, 'Syntactic clustering of the web'. Technical Report 1997-015, Digital Systems Research Center.

Caid, W. R. and P. Oing: 1997, 'System and method of context vector generation and retrieval'. US Patent No. 5619709.

Cutting, D. R., D. R. Karger, J. O. Pedersen, and J. W. Tukey: 1992, 'Scatter/Gather: A cluster-based approach to browsing large document collections'. In: *Proc. ACM SIGIR*.

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman: 1990, 'Indexing by Latent Semantic Analysis'. *Journal of the American Society for Information Science* **41**(6), 391–407.

Dhillon, I. S. and D. S. Modha: 1999, 'Concept decompositions for large sparse text data using clustering'. Technical Report RJ 10147 (95022), IBM Almaden Research Center.

Dhillon, I. S. and D. S. Modha: 2000, 'A parallel data-clustering algorithm for distributed memory multiprocessors'. In: M. J. Zaki and C. T. Ho (eds.): *Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence, Volume 1759*. Springer-Verlag, New York, pp. 245–260. Presented at the 1999 Large-Scale Parallel KDD Systems Workshop, San Diego, CA.

Dhillon, I. S., D. S. Modha, and W. S. Spangler: 1998, 'Visualizing Class Structure of Multidimensional Data'. In: S. Weisberg (ed.): *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics*, Vol. 30. Minneapolis, MN, pp. 488–493.

Duda, R. O. and P. E. Hart: 1973, *Pattern Classification and Scene Analysis*. Wiley.

Frakes, W. B. and R. Baeza-Yates: 1992, *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey.

Gallant, S. I.: 1994, 'Methods for generating or revising context vectors for a plurality of word stems'. US Patent No. 5325298.

Garey, M. R., D. S. Johnson, and H. S. Witsenhausen: 1982, 'The complexity of the generalized Lloyd-Max problem'. *IEEE Trans. Inform. Theory* **28**(2), 255–256.

Golub, G. H. and C. F. Van Loan: 1996, *Matrix computations*. Baltimore, MD, USA: The Johns Hopkins University Press.

Hartigan, J. A.: 1975, *Clustering Algorithms*. Wiley.

Hearst, M. A. and J. O. Pedersen: 1996, 'Reexamining the cluster hypothesis: Scatter/Gather on retrieval results'. In: *Proc. ACM SIGIR*.

Hofmann, T.: 1999, 'Probabilistic latent semantic indexing'. In: *Proc. ACM SIGIR*.

Isbell, C. L. and P. Viola: 1998, 'Restructuring Sparse High Dimensional Data for Effective Retrieval'. In: *Advances in Neural Information Processing*, Vol. 11.

Kleinberg, J., C. H. Papadimitriou, and P. Raghavan: 1998, 'A microeconomic view of data mining'. *Data Mining and Knowledge Discovery* **2**(4), 311–324.

Kolda, T. G.: 1997, 'Limited-Memory Matrix Methods with Applications'. Ph.D. thesis, The Applied Mathematics Program, University of Maryland, College Park, Mayland.

Leland, W. E., M. S. Taqqu, W. Willinger, and D. V. Wilson: 1994, 'On the self-similar nature of ethernet traffic'. *IEEE/ACM Transactions on Networking* **2**(1), 1–15.

Mandelbrot, B. B.: 1988, *Fractal Geometry of Nature*. W. H. Freeman & Company.

O'Leary, D. P. and S. Peleg: 1983, 'Digital image compression by outer product expansion'. *IEEE Trans. Communications* **31**, 441–444.

Papadimitriou, C. H., P. Raghavan, H. Tamaki, and S. Vempala: 1998, 'Latent Semantic Indexing: A Probabilistic Analysis'. In: *Proc. Seventeenth ACM-SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems, Seattle, Washington*. pp. 159–168.

Pollard, D.: 1982, 'Quantization and the method of $k$-means'. *IEEE Trans. Inform. Theory* **28**, 199–205.

Rasmussen, E.: 1992, 'Clustering Algorithms'. In: W. B. Frakes and R. Baeza-Yates (eds.): *Information Retrieval: Data Structures and Algorithms*. pp. 419–442.

Rissanen, J., T. Speed, and B. Yu: 1992, 'Density estimation by stochastic complexity'. *IEEE Trans. Inform. Theory* **38**, 315–323.

Sabin, M. J. and R. M. Gray: 1986, 'Global convergence and empirical consistency of the generalized Lloyd algorithm'. *IEEE Trans. Inform. Theory* **32**(2), 148–155.

Sahami, M., S. Yusufali, and M. Baldonado: 1999, 'SONIA: A Service for Organizing Networked Information Autonomously'. In: *Proc. ACM Digital Libraries*.

Salton, G. and C. Buckley: 1988, 'Term-weighting approaches in automatic text retrieval'. *Inform. Proc. & Management* pp. 513–523.

Salton, G. and M. J. McGill: 1983, *Introduction to Modern Retrieval*. McGraw-Hill Book Company.

Saul, L. and F. Pereira: 1997, 'Aggregate and mixed-order Markov models for statistical language processing'. In: *Proc. 2nd Int. Conf. Empirical Methods in Natural Language Processing*.

Schütze, H. and C. Silverstein: 1997, 'Projections for efficient document clustering'. In: *Proc. ACM SIGIR*.

Silverstein, C. and J. O. Pedersen: 1997, 'Almost-constant-time clustering of arbitrary corpus subsets'. In: *Proc. ACM SIGIR*.

Singhal, A., C. Buckley, M. Mitra, and G. Salton: 1996, 'Pivoted Document Length Normalization'. In: *Proc. ACM SIGIR*.

Vaithyanathan, S. and B. Dom: 1999, 'Model selection in unsupervised learning with applications to document clustering'. In: *Proc. 16th Int. Machine Learning Conf., Bled, Slovenia*.

Willet, P.: 1988, 'Recent trends in hierarchic document clustering: a critical review'. *Inform. Proc. & Management* pp. 577–597.

Zamir, O. and O. Etzioni: 1998, 'Web document clustering: A feasibility demonstration'. In: *Proc. ACM SIGIR*.

Zipf, G. K.: 1949, *Human Behavior and the Principle of Least Effort*. Addison Wesley, Reading, MA.