Concept Decompositions for Large Sparse Text Data using Clustering

Inderjit S. Dhillon and Dharmendra S. Modha
IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120-6099
emails: {dhillon,dmodha}@almaden.ibm.com

Original Version: February 14, 1999
Revised Version: August 5, 1999

Abstract

Unstructured text documents 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 20 million patents (www.patents.ibm.com), the Lexis-Nexis databases contain more than 1 billion documents (www.lexisnexis.com). Furthermore, 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. 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. 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.

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, 1999], 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 revolutionized 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]. Throughout our experimental study, we observed that concept decompositions 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).

In this paper, our main goal is to harness clustering for approximating the word-by-document matrix. After this paper was submitted, the recent work of [Hofmann, 1999], or [H] for short, was brought to our attention by one of the referees. [H] views the word-by-document matrix as a co-occurrence table describing the probability that a word is related to a document, and approximates the co-occurrence table by treating the latent class mixture model as a matrix approximation scheme. While similar in spirit, our work is distinct from [H]:

- Our framework is *geometric* and is concerned with orthonormal L^2 projections, while the framework in [H] 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, [H] assumes a classical multinomial model to describe the statistical structure of a cluster.

- We employ the spherical k-means-type algorithm followed by least-squares approximation step, while [H] employs a 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. Along quite different lines, [H] employs mixture decompositions for enhancing precision in information retrieval.

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 $\boldsymbol{\mathcal{C}}$ and $\boldsymbol{\mathcal{E}}$ 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 13, 31, and 32 that when number of clusters are small a large fraction of words are in fact function words. In this paper, we have selected the function words *independently* of the number of clusters, since, for illustration purposes, we plan to use the same collection of words across various different clusterings. For more formal attempts at determining the function words, see [Vaithyanathan and Dom, 1999].

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 $1 \leq j \leq d$, set the j-th component of document vector $\mathbf{x}_i$, $1 \leq i \leq n$, to be the product of three terms

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

where t_{ji} is the *term weighting component* and is based only on f_{ji}, g_j is the *global weighting component* and is based 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,

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

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 constant.

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) 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 such 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; 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 π_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 π_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 $R^d_{\geq 0}$, we have from the Cauchy-Schwarz inequality that

$$\sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{z} \leq \sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{c}_j. \tag{2}$$

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 π_j.

3.3 The Objective Function

Motivated by (2), we measure the "coherence" or "quality" of each cluster π_j, $1 \le j \le k$, 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

$$\sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{c}_j = n_j \mathbf{m_j}^T \mathbf{c}_j = n_j \|\mathbf{m_j}\| \mathbf{c_j}^T \mathbf{c_j} = n_j \|\mathbf{m_j}\| = \left\| \sum_{\mathbf{x} \in \pi_j} \mathbf{x} \right\|. \tag{3}$$

This rewriting yields the remarkably simple intuition that the quality of each cluster π_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*:

$$\mathcal{Q}\left(\{\pi_j\}_{j=1}^k\right) = \sum_{j=1}^k \sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T \mathbf{c}_j, \tag{4}$$

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 (4), that is, we seek a solution to the following maximization problem:

$$\{\pi_j^\star\}_{j=1}^k = \arg\max_{\{\pi_j\}_{j=1}^k} \mathcal{Q}\left(\{\pi_j\}_{j=1}^k\right). \tag{5}$$

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 $\{\mathbf{c}_j^{(t)}\}_{j=1}^k$:

$$\pi_j^{(t+1)} = \left\{\mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^n : \mathbf{x}^T \mathbf{c}_j^{(t)} \ge \mathbf{x}^T \mathbf{c}_\ell^{(t)}, 1 \le \ell \le n\right\}, \ 1 \le j \le k. \tag{6}$$

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 (6) are known as *Voronoi* or *Dirichlet* partitions.

3. Compute the new concept vectors corresponding to the partitioning computed in (6):

$$\mathbf{c}_j^{(t+1)} = \mathbf{m}_j^{(t+1)}/\|\mathbf{m}_j^{(t+1)}\|, \ 1 \le j \le k, \tag{7}$$

   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

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

for some suitably chosen $\epsilon > 0$. 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 (6) and the second inequality follows from (2). □

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 (3) 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. We can con-

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

Table 1: A confusion matrix whose rows show the distribution of the document vectors of the CLASSIC3 data set across the three clusters: $\pi_1^\dagger$, $\pi_2^\dagger$, and $\pi_3^\dagger$, and whose columns show the distribution of the document vectors across the three classes: MEDLINE, CISI, and CRANFIELD.

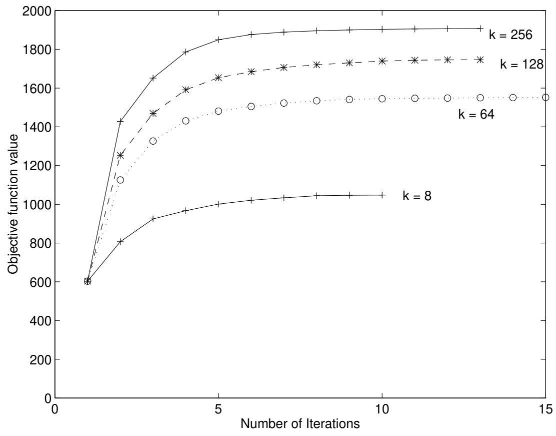


Figure 1: Value of the objective function Q versus the number of k-means iterations for the CLASSIC3 data set.

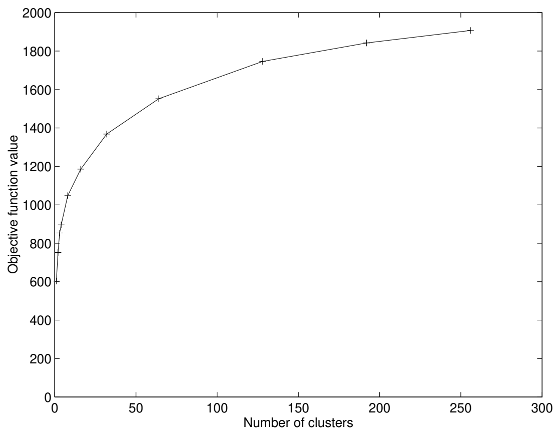


Figure 2: Value of the objective function Q (after the algorithm has converged) versus the number of clusters for the CLASSIC3 data set.

clude from Table 1 that the algorithm indeed "discovers" the class structure underlying the CLASSIC3 data set. Also, see Figure 13 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 Figures 31 and 32 for the top seven words corresponding to each of the ten clusters. 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, 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, it can be seen from Figure 1 that the larger the number of clusters k, the larger the value of the objective function. We further underscore this fact in Figure 2 where we plot the value of the objective function (after the algorithm has converged) as a function of k.

Example 2.C (Objective function never decreases) We repeat Example 1.C for the NSF data set in Figures 3 and 4.

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 proceed to show that such intuitions do not directly carry over to high-dimensional sparse text data sets. Specifically, we examine intra- and inter-cluster structure of the clusters 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, say, $\pi_j^\dagger$, by computing the cosine similarity between

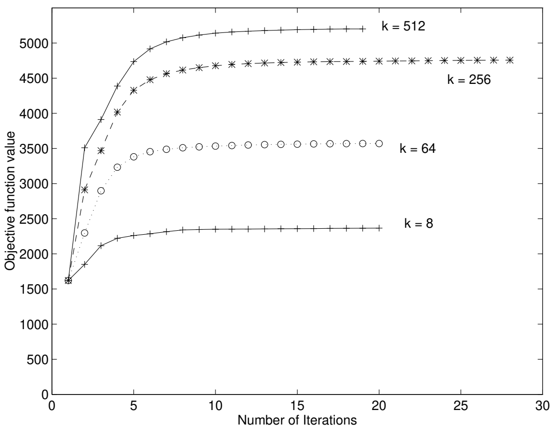


Figure 3: Value of the objective function $\mathcal{Q}$ versus the number of k-means iterations for the NSF data set.

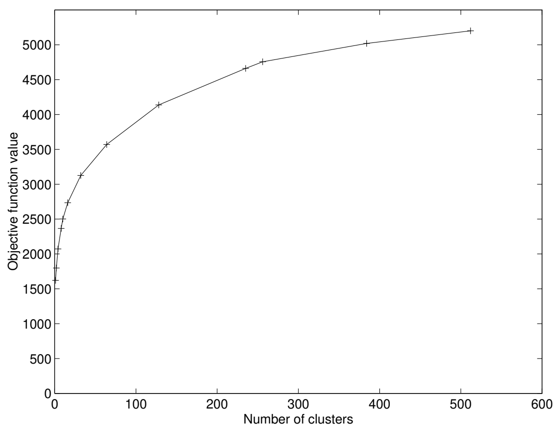


Figure 4: Value of the objective function $\mathcal{Q}$ (after the algorithm has converged) versus the number of clusters for the NSF data set.

every document vector and its concept vector $\mathbf{c}_j^\dagger$, that is, by computing the following $n_j^\dagger$ numbers:

$$\left\{ \mathbf{x}^T \mathbf{c}_j^\dagger : \mathbf{x} \in \pi_j^\dagger \right\},$$

and by plotting an estimated probability density function (pdf) of these numbers. Such plots can be thought of as a projection of the document vectors onto the one-dimensional subspace spanned by the concept vector. Since usually there are many clusters, we can obtain an insight into the *average intra-cluster structure* by computing the following n numbers:

$$\bigcup_{j=1}^{k} \left\{ \mathbf{x}^T \mathbf{c}_j^\dagger : \mathbf{x} \in \pi_j^\dagger \right\}, \tag{8}$$

and by plotting an estimated 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 5, we plot the estimated pdfs of the $n = 13297$ numbers in (8) for these three clusterings. Observe that the inner product between two unit vectors in $R^d_{\geq 0}$ lies in the interval $[0, 1]$–the inner product is 0 if the two vectors are orthogonal and is 1 if the two vectors are identical. Hence, we plot the estimated pdfs only over the interval $[0, 1]$. For $k = 8$, observe that 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 huge 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 and more coherent, and document vectors in a cluster

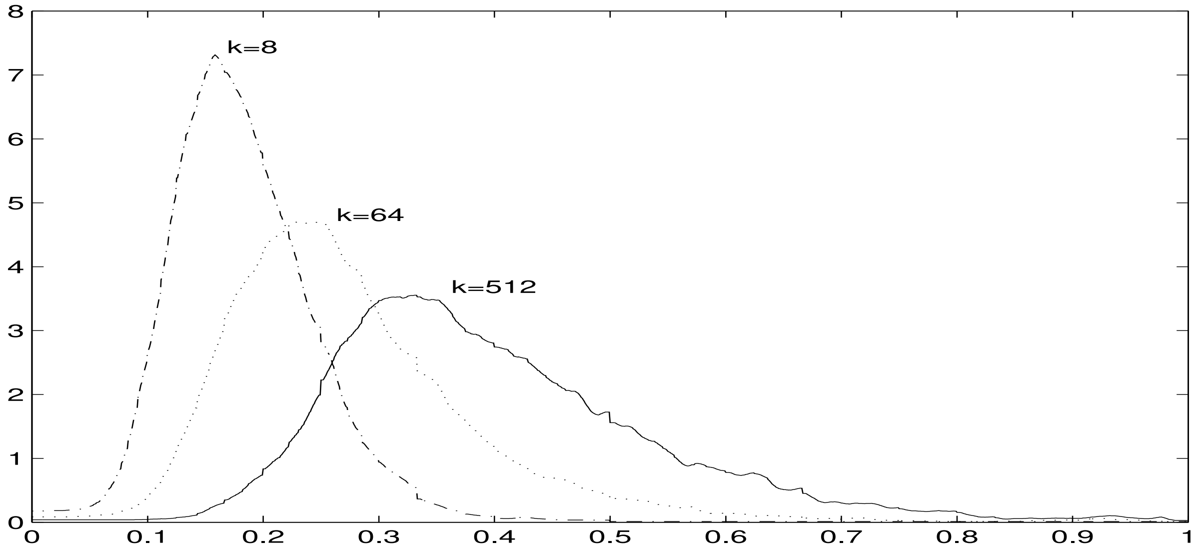


Figure 5: Average *intra*-cluster estimated probability density functions for $k = 8$, 64, and 512 clusters of the NSF data set.

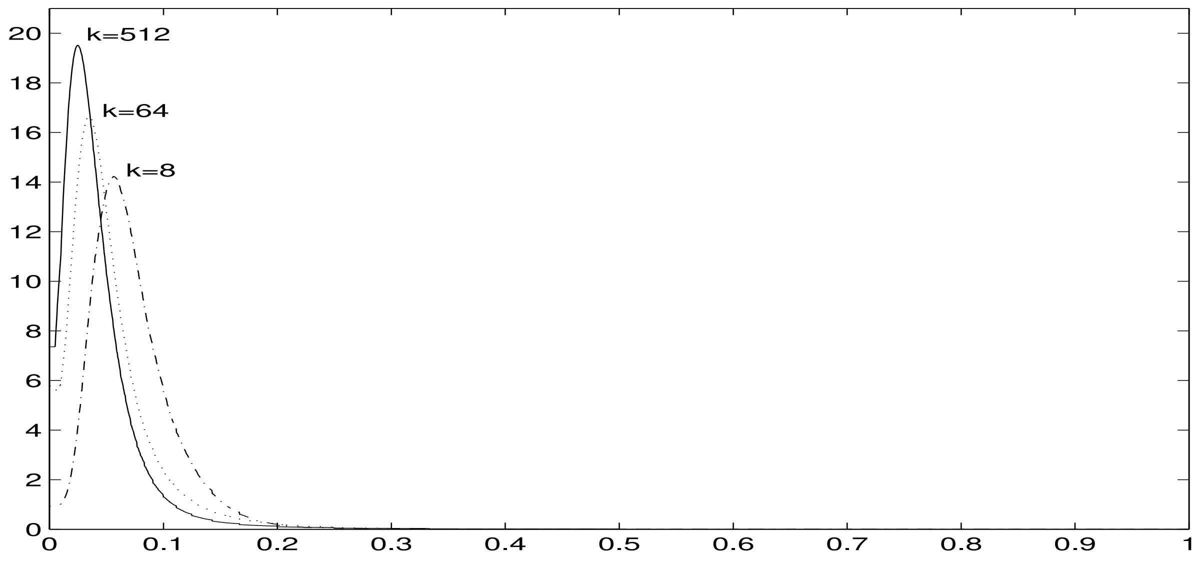


Figure 6: Average *inter*-cluster estimated probability density functions for $k = 8$, 64, and 512 clusters of the NSF data set.

become progressively closer to the corresponding concept vectors. In other words, as k increases, the empty space around the concept vectors progressively shrinks.

In spite of the above example, as Examples 1.B and 2.B show the algorithm does tend to find meaningful clusters. We now 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:

$$\bigcup_{j=1}^{k} \left\{ \mathbf{x}^T \mathbf{c}_j^\dagger : \mathbf{x} \notin \pi_j^\dagger \right\}, \tag{9}$$

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 6, we plot the estimated pdfs for 13297×7, 13297×63, and 13297×511 numbers in (9) corresponding to $k = 8$, 64, and 512, respectively. For $k = 8$, by comparing the inter-cluster estimated pdf in Figure 6 to the intra-cluster estimated pdf in Figure 5, 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, as noted in Example 2.D, although the document vectors in a cluster are not that close to the corresponding concept vector, the document vectors not in a cluster are even further. In a sense, this makes clustering of sparse data work. Also, comparing the intra- and inter-cluster estimated pdfs for $k = 8$, 64, and 512 in Figures 5 and 6, we can see that, as k increases, the overlap between corresponding inter- and intra-cluster estimated pdfs steadily decreases. Furthermore, observe that as k is increased from 8 to 64 and then to 512, the mass of the estimated inter-cluster pdfs in Figure 6 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 are essentially the same 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 estimated pdfs towards 1 and the corresponding progressive movement of the inter-cluster estimated pdfs towards 0. We believe that these facts point to a certain "fractal-like" or "self-similar" nature of high-dimensional sparse data that we do not yet completely comprehend.

3.6 Comparison with the Euclidean k-means Algorithms

We now point out the difference between our objective function (4) and the following more traditional objective function [Duda and Hart, 1973, Hartigan, 1975] used for measuring the quality of clusterings:

$$\sum_{j=1}^{k} \sum_{\mathbf{x} \in \pi_j} \|\mathbf{x} - \mathbf{m}_j\|^2, \tag{10}$$

where $\mathbf{m}_j$ is the mean vector associated with the cluster π_j. The objective function in (10) 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 (6) by:

$$\pi_j^{(t+1)} = \left\{ \mathbf{x} \in \{\mathbf{x}_i\}_{i=1}^n : \|\mathbf{x} - \mathbf{m}_j^{(t)}\|^2 \leq \|\mathbf{x} - \mathbf{m}_\ell^{(t)}\|^2, 1 \leq \ell \leq n \right\}, \ 1 \leq j \leq k. \tag{11}$$

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 characterize the shape of the clusters obtained by minimizing the objective function $\mathcal{Q}$ and the objective function in (10). It follows from (6) that the boundary between any two clusters associated with the spherical k-means algorithm, say, π_j and π_ℓ, 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, it follows from (11) that the boundary between any two clusters associated with the Euclidean k-means algorithm, say, π_j and π_ℓ, 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 an arbitrary hyperplane and its 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 (4) equivalently as

$$\mathcal{Q}\left(\{\pi_j\}_{j=1}^k\right) = \sum_{j=1}^k \sum_{\mathbf{x} \in \pi_j} \mathbf{x}^T\mathbf{c}_j = 2n - 2\sum_{j=1}^k \sum_{\mathbf{x} \in \pi_j} \|\mathbf{x} - \mathbf{c}_j\|^2 \equiv 2n - 2\mathcal{F}\left(\{\pi_j\}_{j=1}^k\right). \tag{12}$$

Hence, it follows that the maximization problem in (5) is equivalent to the following minimization problem:

$$\{\pi_j^\star\}_{j=1}^k = \underset{\{\pi_j\}_{j=1}^k}{\arg\min} \mathcal{F}\left(\{\pi_j\}_{j=1}^k\right). \tag{13}$$

In the next section, we show that (13) 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 π_j for some $1 \le j \le k$, we can approximate it by the concept vector $\mathbf{c}_j$. Recall that the word-by-document matrix $\mathbf{X}$ is a $d \times n$ matrix whose i-th column, for $1 \le i \le n$, is the document vector $\mathbf{x}_i$. Define a $d \times n$ matrix $\hat{\mathbf{X}}_k = \hat{\mathbf{X}}_k(\{\pi_j\}_{j=1}^k)$ such that, for $1 \le i \le n$, the i-th column of the matrix 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

$$\|\mathbf{X} - \hat{\mathbf{X}}_k\|_F^2 = \sum_{j=1}^{k} \sum_{\mathbf{x} \in \pi_j} \|\mathbf{x} - \mathbf{c}_j\|^2 = \mathcal{F}\left(\{\pi_j\}_{j=1}^k\right). \tag{14}$$

It follows from (12) and (14) that maximizing the objective function $\mathcal{Q}$ can also be thought of as a constrained matrix approximation problem.

In principle, the concept vectors $\{\mathbf{c}_j\}_{j=1}^k$ may be linearly dependent; however, in practice, we seldom found this to be the case. Hence, from now on, we shall assume that they are linearly independent. Under this assumption, for any partitioning $\{\pi_j\}_{j=1}^k$ the corresponding matrix approximation $\hat{\mathbf{X}}_k$ has rank k. In particular, the matrix approximation $\hat{\mathbf{X}}_k^\dagger$ corresponding to the final partitioning $\{\pi_j^\dagger\}_{j=1}^k$ produced by the spherical k-means algorithm has rank k. In the next subsection, we compare the approximation power of $\hat{\mathbf{X}}_k^\dagger$ to that of the best rank-k approximations to the word-by-document matrix $\mathbf{X}$.

4.2 Matrix Approximation using Singular Value Decompositions

Singular value decomposition (or SVD, for short) is well known in information retrieval and text mining, and has been widely studied for 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 SVD of $\mathbf{X}$ as

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

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 $\boldsymbol{\Sigma}$ 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. 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. The diagonal elements of $\boldsymbol{\Sigma}$ are nonnegative square roots of the eigenvalues of $\mathbf{X}\mathbf{X}^T$ and of $\mathbf{X}^T\mathbf{X}$. We can write

$$\mathbf{X}\mathbf{X}^T = \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^T, \quad \mathbf{X}^T\mathbf{X} = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T.$$

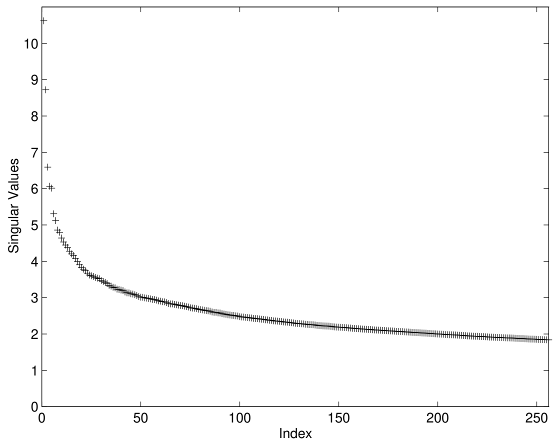


Figure 7: The largest 256 singular values of the CLASSIC3 word-by-document matrix.

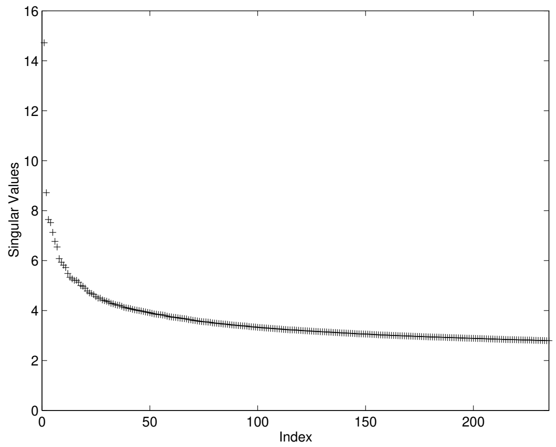


Figure 8: The largest 235 singular values of the NSF word-by-document matrix.

In statistical terminology, the columns of $\mathbf{U}$ are known as principal components[1]. The columns of $\mathbf{U}$ are known as the *left-singular vectors*; for brevity, we shall refer to them as simply the singular vectors.

For $1 \leq k \leq r$, 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 $\mathbf{\Sigma_k}$ be obtained deleting the last $(r-k)$ rows and columns of $\mathbf{\Sigma}$. The $d \times n$ matrix

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

is known as a k-*truncated SVD* of the matrix $\mathbf{X}$, and has rank equal to k. It 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 7, we plot the largest 256 singular values of the CLASSIC3 word-by-document matrix.

Example 2.F (singular Values) In Figure 8, 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 Examples 1.D and 2.F) 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 $(1/n)\sum_{i=1}^{n} x_i$ from every document vector.

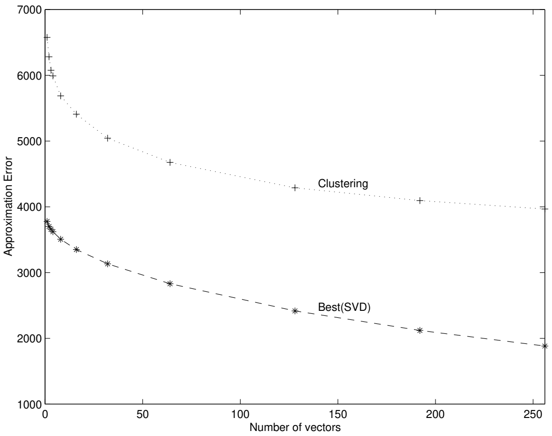


Figure 9: Comparing the approximation errors $\|X - \bar{X}_k\|_F^2$ and $\|X - \hat{X}_k^\dagger\|_F^2$ for the CLASSIC3 data set for various values of k.

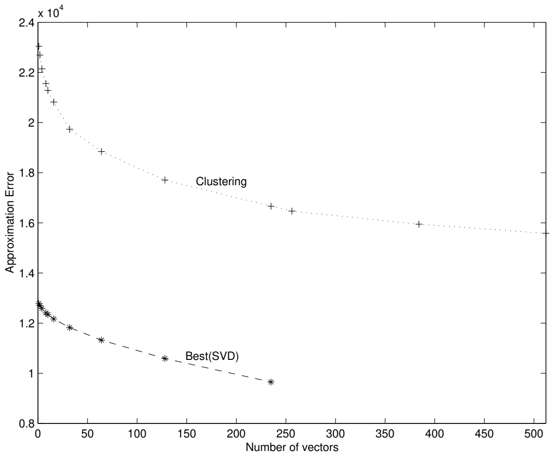


Figure 10: Comparing the approximation errors $\|X - \bar{X}_k\|_F^2$ and $\|X - \hat{X}_k^\dagger\|_F^2$ for the NSF data set for various values of k.

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

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

Lemma 4.1 holds for arbitrary matrices 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,

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

We now empirically validate and examine this lower bound.

Example 1.E (comparing $\bar{X}_k$ and $\hat{X}_k^\dagger$) In Figure 9, we compare the errors in approximating the CLASSIC3 word-by-document matrix using the k-truncated SVDs and the matrices $\hat{X}_k^\dagger$ 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 $\hat{X}_k^\dagger$.

Example 2.G (comparing $\bar{X}_k$ and $\hat{X}_k^\dagger$) In Figure 10, we repeat Example 1.E for the NSF data set.

Remark 4.1 Observe that for the NSF data set, in Figure 10, 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 find 512 concept vectors for the NSF data set.

Remark 4.2 Observe that (12) establishes that the upper curves in Figures 9 and 10 are intimately tied to Figures 2 and 4, respectively.

It follows from Figures 9 and 10 that the matrix approximation errors $||\mathbf{X} - \bar{\mathbf{X}}_k||_F^2$ and $||\mathbf{X} - \hat{\mathbf{X}}_k^\dagger||_F^2$ are not close to each other; hence, there is a significant room for seeking better rank-k matrix approximations than $\hat{\mathbf{X}}_k^\dagger$. With hindsight, this is to be expected, since $\hat{\mathbf{X}}_k^\dagger$ uses a naive strategy of approximating each document vector by the closest concept vector. In the next subsection, we show that if we approximate each document vector by a linear combination of the concept vectors then it is possible to obtain significantly better matrix approximations.

4.3 Concept Decompositions

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 $1 \leq j \leq k$, the j-th column of the matrix is the concept vector $\mathbf{c}_j$, that is,

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

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 approximations 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:

$$\mathbf{Z}^\star = \arg\min_{\mathbf{Z}} ||\mathbf{X} - \mathbf{C}_k \mathbf{Z}||_F^2. \tag{15}$$

It is well known that a closed-form solution exists for the least-squares problem (15), 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 actually 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 $\hat{\mathbf{X}}_k$.

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 π_j and $p_{ji} = 0$ otherwise. However, the matrix $\mathbf{P}_k$ need not be the solution of the least-squares minimization problem (15), hence the inequality (b) follows. $\square$

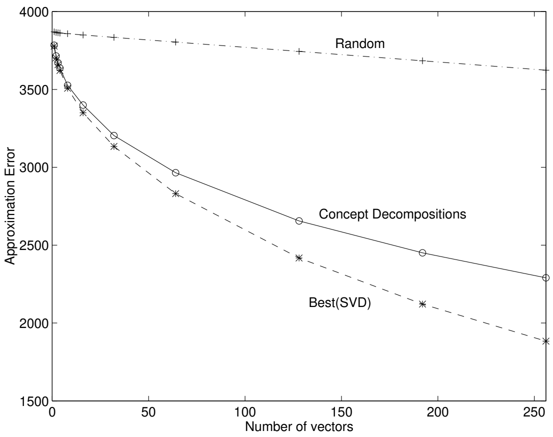


Figure 11: Comparing the approximation errors $||\mathbf{X} - \bar{\mathbf{X}}_k||_F^2$, $||\mathbf{X} - \tilde{\mathbf{X}}_k^\dagger||_F^2$, and $||\mathbf{X} - \check{\mathbf{X}}_k||_F^2$ for the CLASSIC3 data set for various values of k.

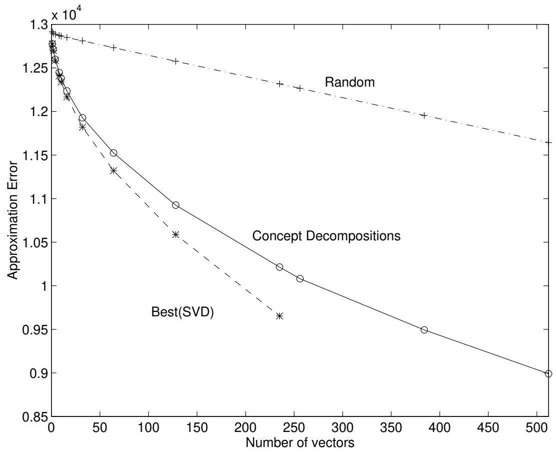


Figure 12: Comparing the approximation errors $||\mathbf{X} - \bar{\mathbf{X}}_k||_F^2$, $||\mathbf{X} - \tilde{\mathbf{X}}_k^\dagger||_F^2$, and $||\mathbf{X} - \check{\mathbf{X}}_k||_F^2$ for the NSF data set for various values of k.

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. Before, we empirically validate Lemma 4.2, 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 11, 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 approximation error $||\mathbf{X} - \tilde{\mathbf{X}}_k^\dagger||_F^2$ is quite close to the optimal $||\mathbf{X} - \bar{\mathbf{X}}_k||_F^2$, and is significantly better than $||\mathbf{X} - \check{\mathbf{X}}_k||_F^2$. Finally, comparing Figures 9 and 11, it follows that the approximation error $||\mathbf{X} - \tilde{\mathbf{X}}_k^\dagger||_F^2$ is a major improvement over the approximation error $||\mathbf{X} - \hat{\mathbf{X}}_k^\dagger||_F^2$.

Example 2.H (comparing $\bar{\mathbf{X}}_k$, $\tilde{\mathbf{X}}_k^\dagger$, and $\check{\mathbf{X}}_k$) In Figure 12, we repeat Example 1.F for the NSF data set.

5 Concept Vectors and Singular vectors: A comparison

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 $1 \le j \le k$, we now associate a *word cluster* W_j with the document cluster π_j as follows. A word $1 \le w \le d$ 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

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

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 13 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. On the right hand side of Figure 14 we list the top ten words for the leading three singular vectors of the CLASSIC3 word-by-document matrix. However, in this case, the words are distributed across the three underlying concepts, and are not localized.

Example 2.I (word clusters) In Figures 31, 32, 33, and 34, we repeat Example 1.G for the NSF data set with $k = 10$ clusters and with the leading 10 singular vectors.

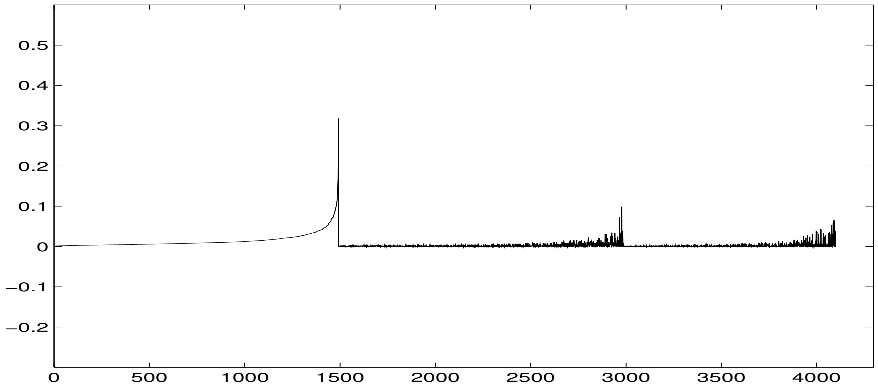
We now define a *total order* on the d words as follows. For $1 \leq j \leq k$, we order the words in the W_j in the increasing order of their respective weight in $\mathbf{c}_j$. Next, we impose an arbitrary order on the word clusters themselves. For example, order the word clusters such that words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

Example 1.H (locality of concept vectors) Using the same clustering of the CLASSIC3 data set into $k = 3$ clusters as in Example 1.G, in Figure 13, 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 in evident in the figure and so is the ordering of the words within a word cluster. It can be seen from the figure that most of the weight of a concept vector is concentrated in or localized to the corresponding word cluster. Analogously, in Figure 14, 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 more than one word clusters, and hence are global. Finally, observe that the concept vectors are always nonnegative, whereas the singular vectors can assume both positive and negative values.

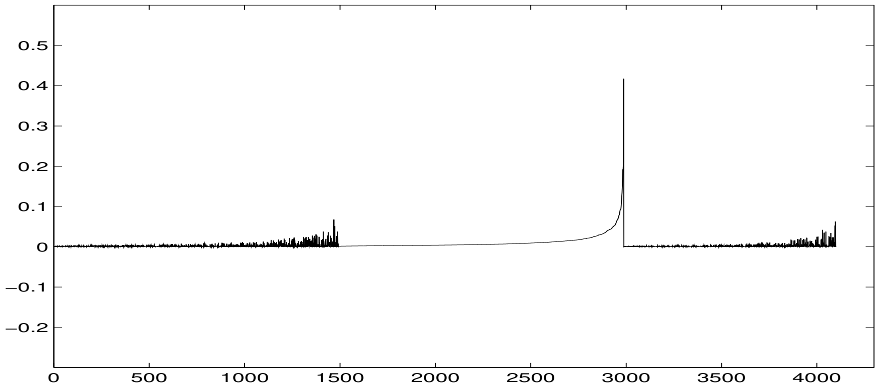
Example 2.J (locality of concept vectors) We repeat Example 1.H for the NSF data set with $k = 10$ clusters and $k = 10$ top singular vectors in Figures 31, 32, 33, and 34. We leave the task of comparing and contrasting these figures to the reader as an exercise.

Intuitively speaking, it follows from the above examples that the concept vectors are local to each word cluster and can be compared to wavelets, while the singular vectors are global across the word clusters and can be compared to Fourier series.

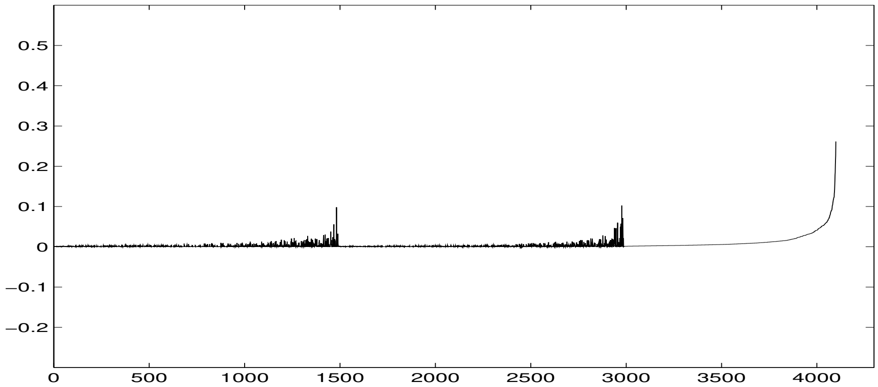
Observe that for a fixed k, the concept matrix $\mathbf{C}_k^\dagger$ (corresponding to the final partitions produced by the spherical k-means algorithm) has $d \times k$ entries. We now empirically study the number of nonzero entries in the concept matrix as the number of clusters is increased.



patients	0.317878
cells	0.242986
treatment	0.161797
normal	0.161455
growth	0.148922
blood	0.137679
children	0.131082
disease	0.115177
cell	0.112006
human	0.110402

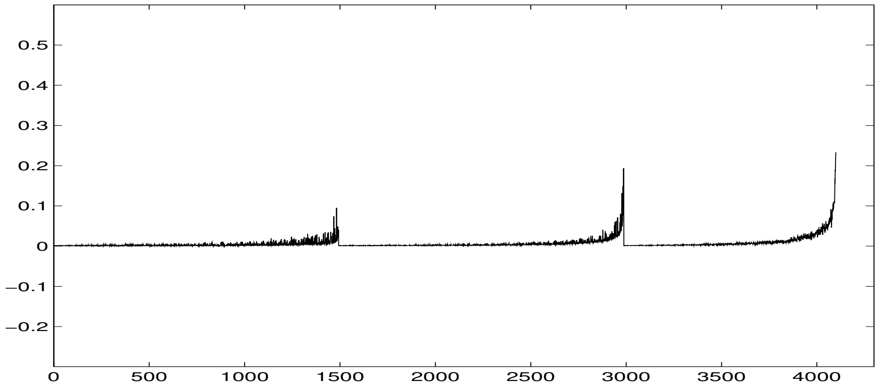


library	0.417128
research	0.210852
libraries	0.200998
retrieval	0.192128
data	0.191829
systems	0.179494
science	0.163754
scientific	0.152427
book	0.134523
study	0.129123

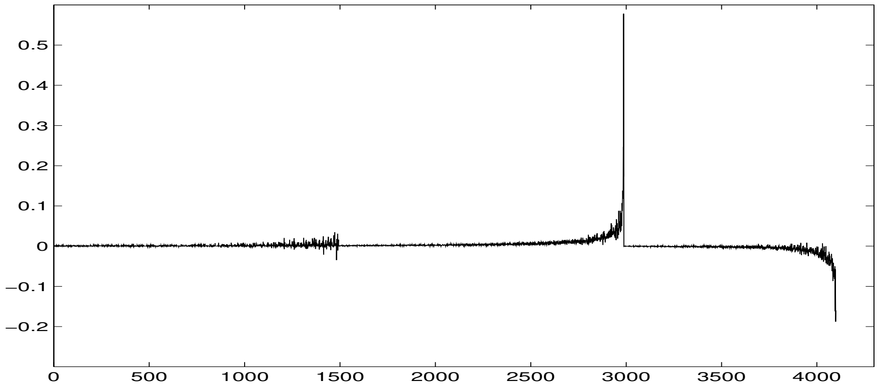


pressure	0.261630
boundary	0.226353
theory	0.204605
layer	0.184686
mach	0.179034
method	0.165160
supersonic	0.142664
heat	0.136141
shock	0.132581
effects	0.122806

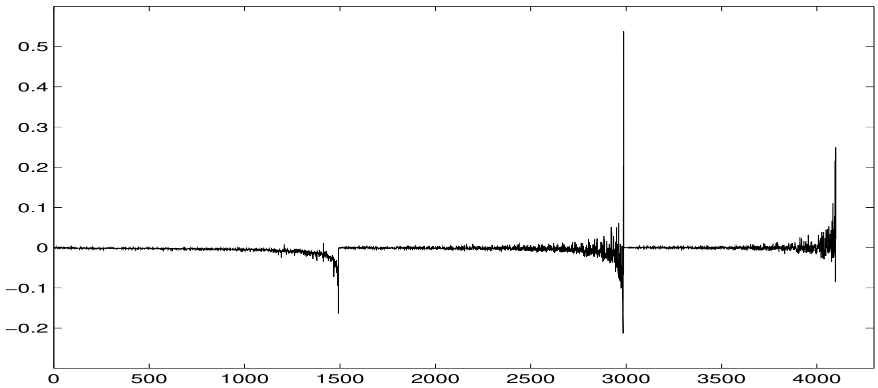
Figure 13: The three concept vectors corresponding to a clustering of the CLASSIC3 data set into 3 clusters. For each concept vector, the top ten words with the corresponding weights are shown on the right.



pressure	+0.233539
boundary	+0.223440
library	+0.193893
layer	+0.189404
theory	+0.186251
method	+0.168910
mach	+0.154866
data	+0.148474
analysis	+0.131805
shock	+0.111746



library	+0.578096
libraries	+0.246430
boundary	−0.187940
research	+0.173021
layer	−0.162006
pressure	−0.159707
systems	+0.137941
data	+0.120045
retrieval	+0.118030
science	+0.114411



library	+0.538484
boundary	+0.249150
layer	+0.216349
retrieval	−0.213215
libraries	+0.202313
patients	−0.163804
data	−0.153484
scientific	−0.130758
systems	−0.129484
cells	−0.115206

Figure 14: 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.

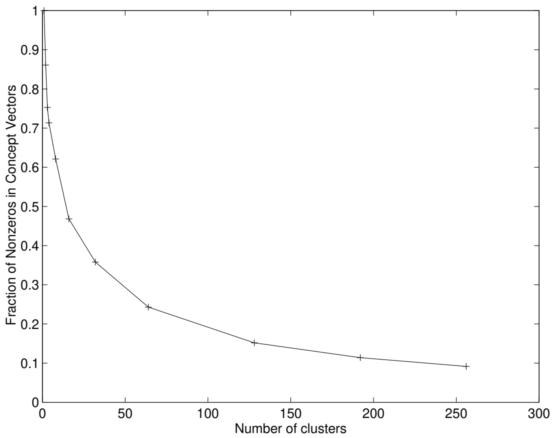


Figure 15: The sparsity of the CLASSIC3 concept matrix $\mathbf{C}_k^\dagger$.

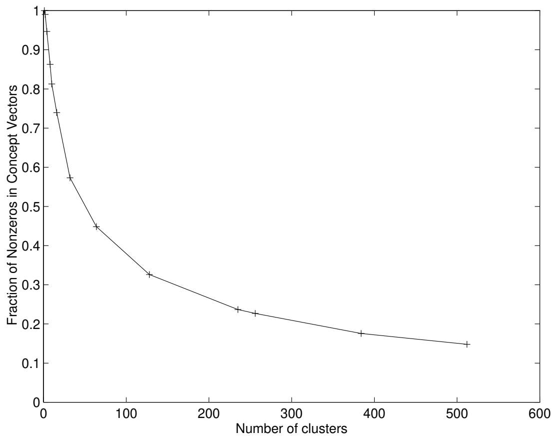


Figure 16: The sparsity of the NSF concept matrix $\mathbf{C}_k^\dagger$.

Example 1.I (sparsity of concept vectors) In Figure 15, we plot the ratio of the number of nonzero entries in the concept matrix to the total number of entries, namely, $4099 \times k$, in the concept matrix for various values of k. It can be seen that as the number of clusters is increased, the concept matrix becomes progressively sparser. For example, for $k = 256$ clusters, the concept matrix is roughly 90% sparse.

Example 2.K (sparsity of concept vectors) In Figure 16, we repeat Example 1.I for the NSF data set.

Examples 1.I and 2.K establish that concept vectors become progressively sparser as the number of clusters is increased. In contrast, the singular vectors are virtually completely dense. Intuitively, as the number of clusters increase, there are fewer and fewer document vectors in a cluster (in an average sense). Now, the document vector are almost 99% sparse, and as their number in a cluster decreases, the total number of nonzero words in the cluster correspondingly decreases, and hence the sparsity of the concept vectors.

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

$$\frac{2}{k(k-1)} \sum_{a=1}^{k} \sum_{b=a+1}^{k} (\mathbf{c}_a^\dagger)^T \mathbf{c}_b^\dagger. \tag{16}$$

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 1.J (average inner product between the concept vectors) For the CLASSIC3 data set, in Figure 17, we plot the average inner product between the concept vectors in (16) versus the number of clusters k. It can be seen that as the number of clusters k increases, the average inner product between the concept vectors progressively moves towards 0.

Example 2.L (average inner product between the concept vectors) In Figure 18, we repeat Example 1.J for the NSF data set.

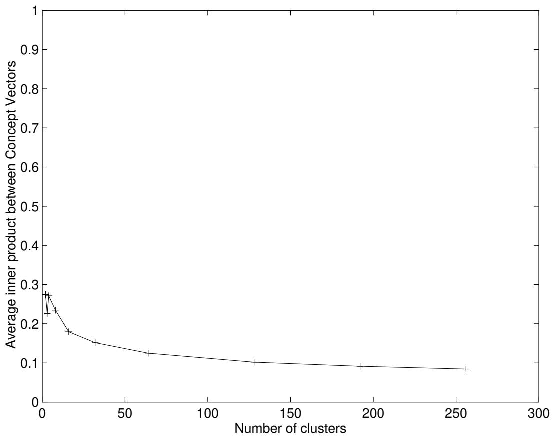


Figure 17: The average inner product between concept vectors for the CLASSIC3 data set.

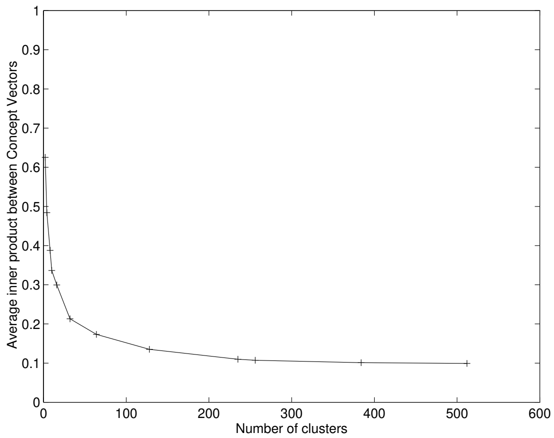


Figure 18: The average inner product between concept vectors for the NSF data set.

It follows from the above examples that as the number of clusters increase, the concept vectors tend towards "orthonormality." In contrast, in general, the singular vectors are orthonormal.

So far, in this section, we have contrasted concept vectors and the singular vectors. Nonetheless, in spite of these contrasts, the next subsection (our final 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

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

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

$$\mathcal{E}_k = \mathrm{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\}.$$

Before we can compare these subspaces, we briefly discuss a general notion of similarity between two subspaces 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 $\mathbb{R}^d$.

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

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

The *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: $\|f\| = 1, \|g\| = 1, f^T f_j = 0, g^T g_j = 0, j = 1, 2, \cdots, \ell - 1$. The vectors $(f_1, \cdots, f_q)$ and $(g_1, \cdots, g_q)$ are called *principal vectors* of the pair of subspaces. Finally, define 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. Intuitively, θ_1 is the angle between two closest unit vectors $f_1 \in \mathcal{F}$ and $g_1 \in \mathcal{G}$, θ_2 is the angle between two closest unit vectors $f_2 \in \mathcal{F}$ and $g_2 \in \mathcal{G}$ such that f_2 and g_2 are, respectively, orthogonal to f_1 and g_1, and so on.

Example 1.K (principal angles) In Table 2, we compare the singular subspace $\mathcal{E}_3$ with various concept subspaces for the CLASSIC3 data set. Observe that, as k increases, the cosines of the all principal angles tend to 1. In fact, for $k = 16$ the singular subspace $\mathcal{E}_3$ 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.

	$\cos \theta_1$	$\cos \theta_2$	$\cos \theta_3$
$\mathcal{C}_3^\dagger$	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

Table 2: Cosines of the principal angles between the singular subspace $\mathcal{E}_3$ and various concept subspaces for the CLASSIC3 data set.

Example 2.M (principal angles) In Table 3, we compare the singular subspace $\mathcal{E}_{10}$ with various concept subspaces for the NSF data set. Observe that, as k increases, the cosines of the all principal angles tend to 1.

	$\cos \theta_1$	$\cos \theta_2$	$\cos \theta_3$	$\cos \theta_4$	$\cos \theta_5$	$\cos \theta_6$	$\cos \theta_7$	$\cos \theta_8$	$\cos \theta_9$	$\cos \theta_{10}$
$\mathcal{C}_{10}^\dagger$	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

Table 3: Cosines of the principal angles between the singular subspace $\mathcal{E}_{10}$ and various concept subspaces for the NSF data set.

We are not interested in comparing individual singular subspaces to individual concept subspaces, rather we are interested in comparing the sequence of the singular subspaces $\{\mathcal{E}_k\}_{k \geq 1}$ to the sequence of the concept subspaces $\{\mathcal{C}_k^\dagger\}_{k \geq 1}$. Since it is hard to directly compare the sequences, in the following examples, (a) we compare a fixed singular subspace to various concept subspaces and (b) we compare a fixed concept subspace to various singular subspaces.

Example 1.L (comparing a singular subspace to various concept subspaces) For the CLASSIC3 data set, in Figures 19 and 20, we plot the average cosine of the principal angles and all the cosines of the principal angles, respectively, between the singular subspace $\mathcal{E}_{64}$ and various concept subspaces. It can be seen from Figure 19 that as k is increased from 1 to 64 the average cosine of principal angles between $\mathcal{E}_{64}$ and $\mathcal{C}_k^\dagger$ progressively decreases. On the other hand, as k is increased from 64 to 256 the average cosine of principal angles between $\mathcal{E}_{64}$ and $\mathcal{C}_k^\dagger$ progressively increases. Also, it can be seen from Figure 20 that as k is increased from 64 to 256 the cosines of the principal angles tend towards 1. In Figures 21 and 22, we plot the average cosine of the principal angles and all the cosines of the principal angles, respectively, between the singular subspace $\mathcal{E}_{256}$ and various concept subspaces.

Example 2.N (comparing a singular subspace to various concept subspaces) In Figures 25, 26, 27, and 28, we repeat Example 1.L for the NSF data set.

Example 1.M (comparing a concept subspace to various singular subspaces) For the CLASSIC3 data set, in Figures 23 and 24, we plot the average cosine of the principal angles and all the cosines of the principal angles, respectively, between the concept subspace $\mathcal{C}_{64}^\dagger$ and various singular subspaces. It can be seen from Figure 23 that as k is increased from 1 to 64 the average cosine of principal angles between $\mathcal{C}_{64}^\dagger$ and $\mathcal{E}_k$ progressively decreases. On the other hand, as k is increased from 64 to 256 the average cosine of principal angles between $\mathcal{C}_{64}^\dagger$ and $\mathcal{E}_k$ progressively increases. Also, it can be seen from Figure 20 that as k is increased from 64 to 256 the cosines of the principal angles tend towards 1.

Example 2.O (comparing a concept subspace to various singular subspaces) In Figures 29 and 30, we repeat Example 1.M 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 when mining low-dimensional and 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 13, 31, and 32). Geometrically, the spherical k-means clustering algorithm partitions the high-dimensional space into Voronoi or Dirichlet regions separated by hyperplanes passing through the origin. Associated with each cluster is its *concept vector* that is its centroid normalized to have unit Euclidean norm.

The spherical k-means algorithm seeks high-coherence clusters, where we define coherence of each cluster to be the average cosine similarity between each document vector and the associated concept vector. 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? We reconcile this paradox by observing that

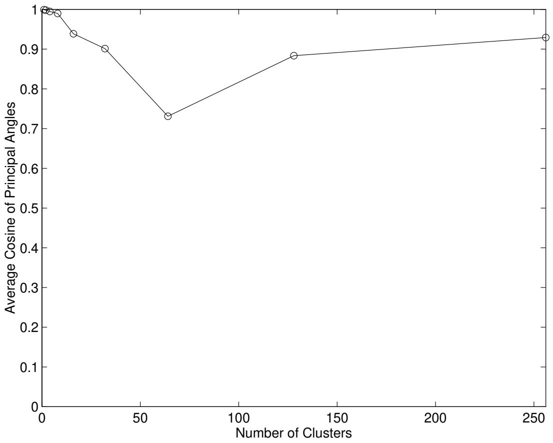


Figure 19: Average cosine of the principal angles between $\mathcal{E}_{64}$ and various concept subspaces for the CLASSIC3 data set.

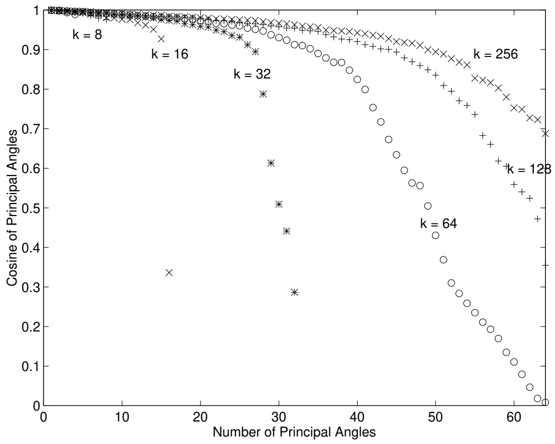


Figure 20: Cosines of the principal angles between $\mathcal{E}_{64}$ and various concept subspaces for the CLASSIC3 data set.

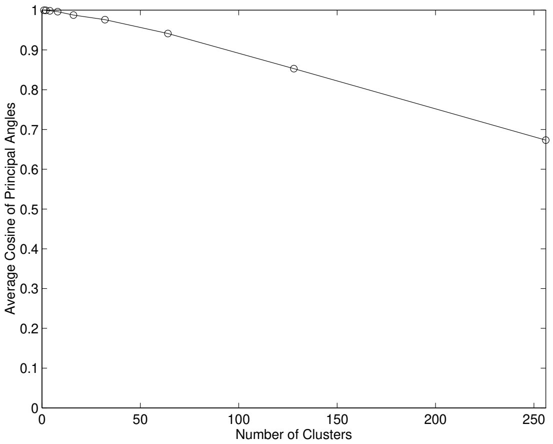


Figure 21: Average cosine of the principal angles between $\mathcal{E}_{256}$ and various concept subspaces for the CLASSIC3 data set.

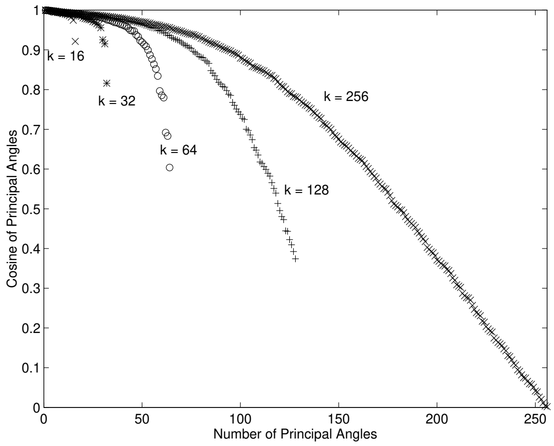


Figure 22: Cosines of the principal angles between $\mathcal{E}_{256}$ and various concept subspaces for the CLASSIC3 data set.

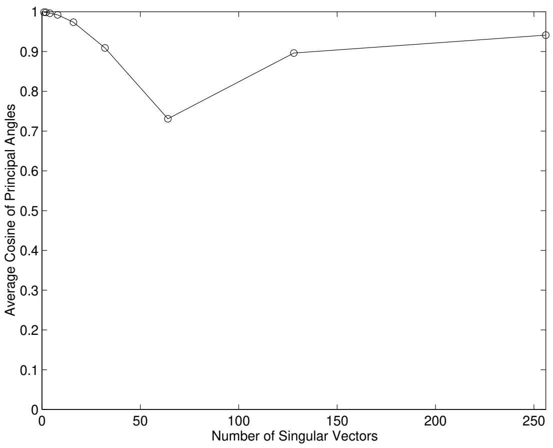


Figure 23: Average cosine of the principal angles between $\mathcal{C}^{\dagger}_{64}$ and various singular subspaces for the CLASSIC3 data set.

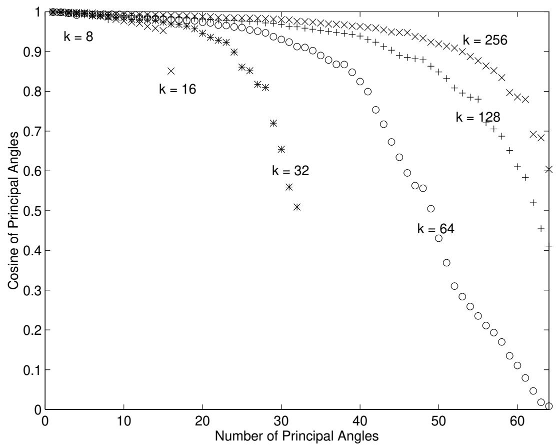


Figure 24: Cosines of the principal angles between $\mathcal{C}^{\dagger}_{64}$ and various singular subspaces for the CLASSIC3 data set.

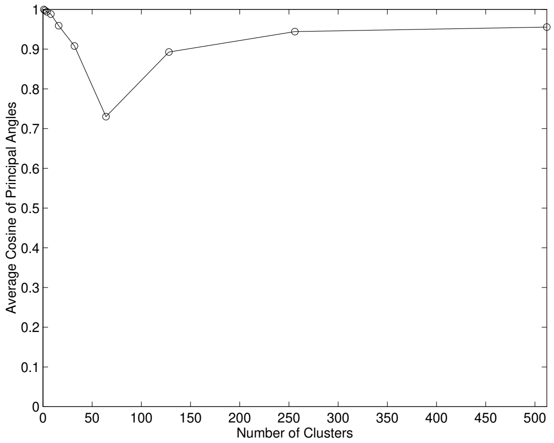


Figure 25: Average cosine of the principal angles between $\mathcal{E}_{64}$ and various concept subspaces for the NSF data set.

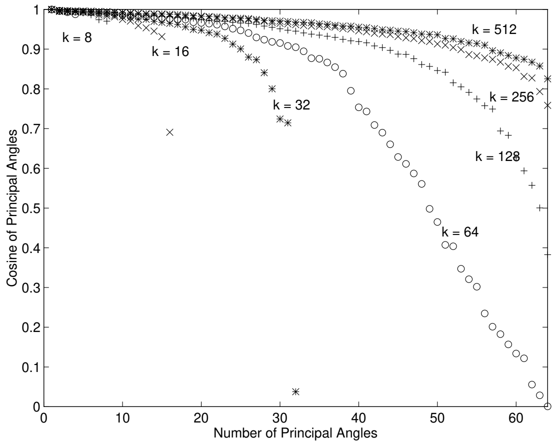


Figure 26: Cosines of the principal angles between $\mathcal{E}_{64}$ and various concept subspaces for the NSF data set.

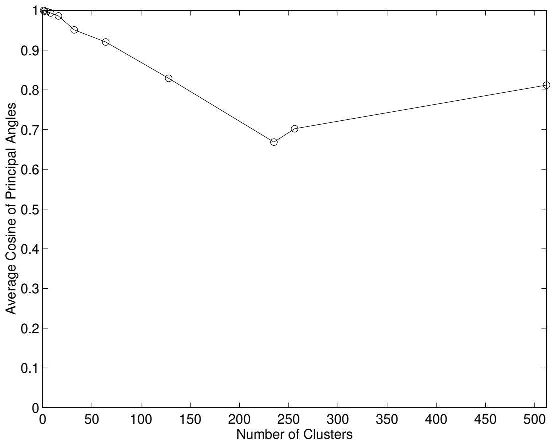


Figure 27: Average cosine of the principal angles between $\mathcal{E}_{235}$ and various concept subspaces for the NSF data set.

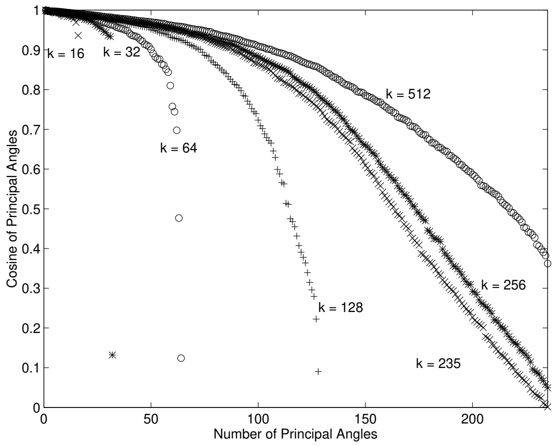


Figure 28: Cosines of the principal angles between $\mathcal{E}_{235}$ and various concept subspaces for the NSF data set.

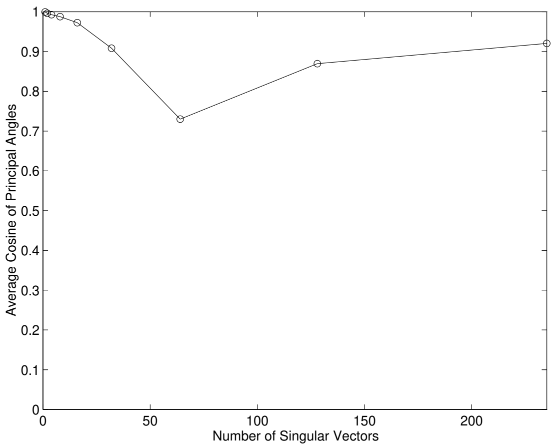


Figure 29: Average cosine of the principal angles between $\mathcal{C}^{\dagger}_{64}$ and various singular subspaces for the NSF data set.

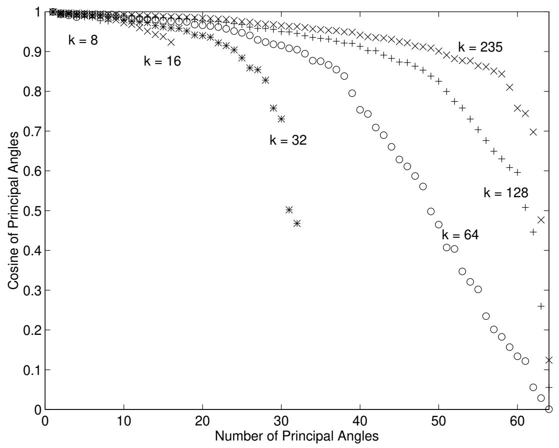
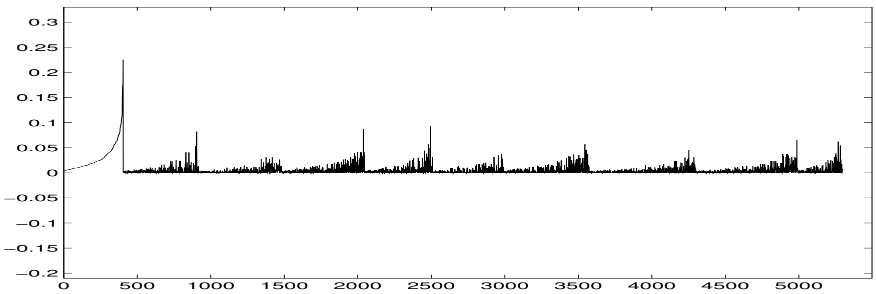
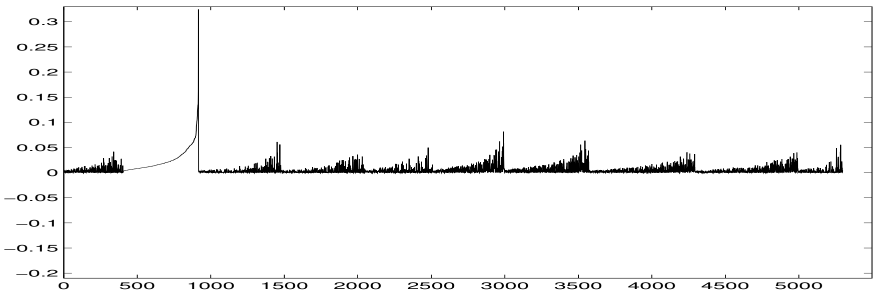


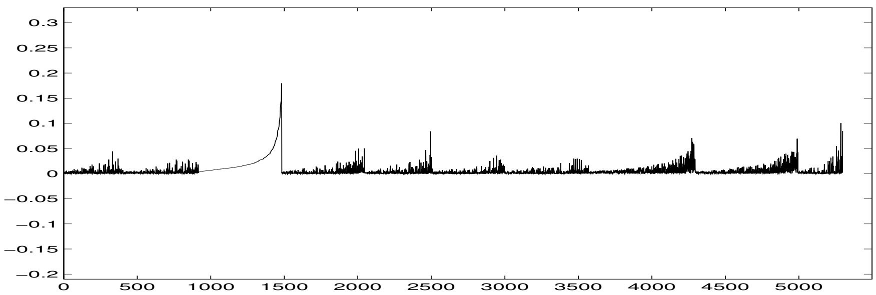
Figure 30: Cosines of the principal angles between $\mathcal{C}^{\dagger}_{64}$ and various singular subspaces for the NSF data set.



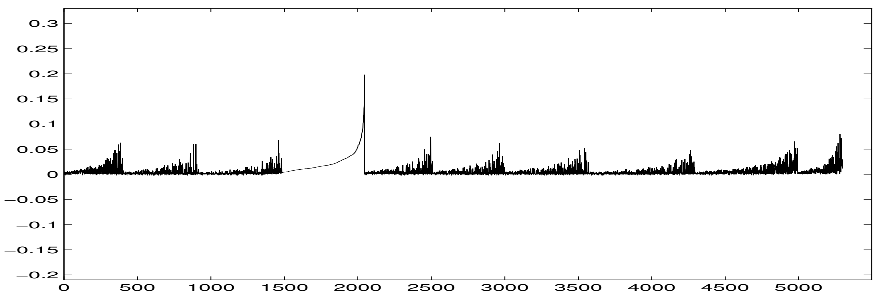
solar	0.225248
galaxies	0.179245
observations	0.172646
ast	0.172102
stars	0.153223
waves	0.145713
telescope	0.115486



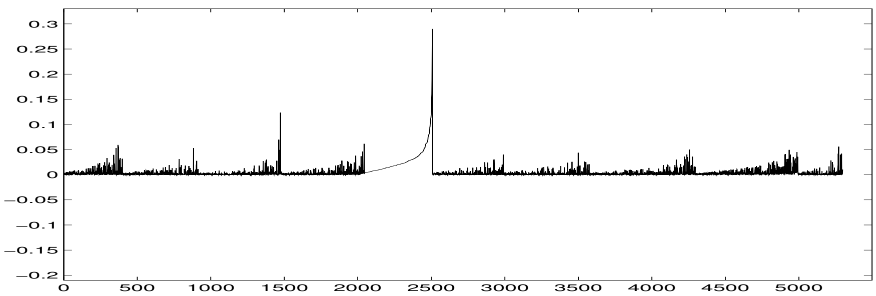
species	0.324644
populations	0.162039
plant	0.146033
genetic	0.135277
population	0.134982
plants	0.128927
evolutionary	0.124207



courses	0.179539
faculty	0.169954
undergraduate	0.155252
education	0.143993
college	0.143863
teachers	0.141829
student	0.139139

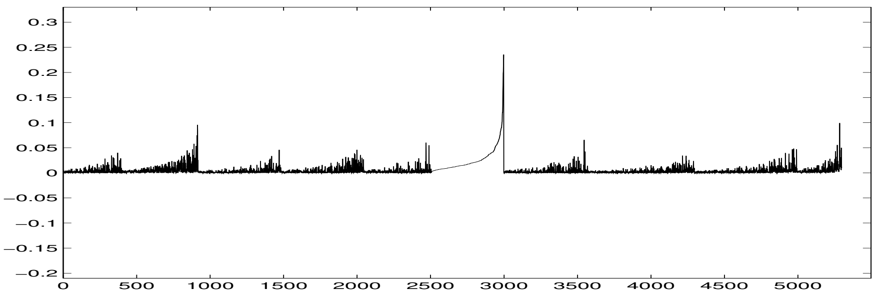


materials	0.198163
phase	0.140753
properties	0.134193
optical	0.131578
devices	0.117619
films	0.117072
magnetic	0.114698

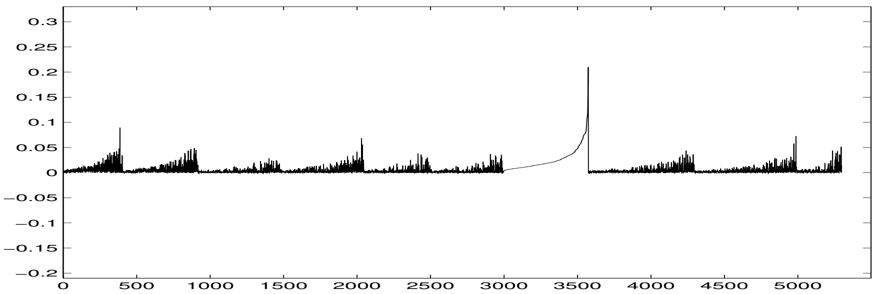


theory	0.289798
equations	0.240976
geometry	0.188458
differential	0.159887
algebraic	0.153347
mathematical	0.149532
spaces	0.135558

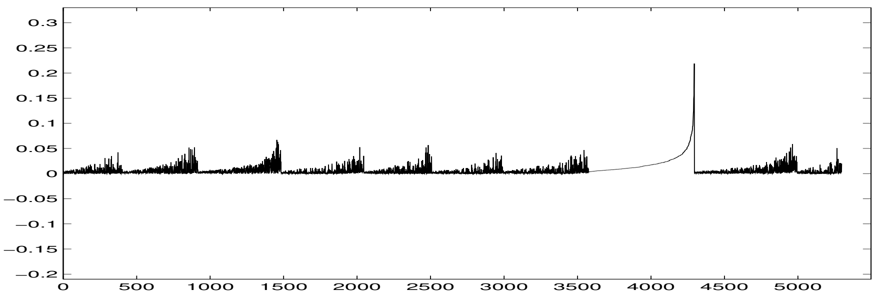
Figure 31: The concept vectors 1-5 corresponding to a clustering of the NSF data set into 10 clusters. For each concept vector, the top seven words with the corresponding weights are shown on the right.



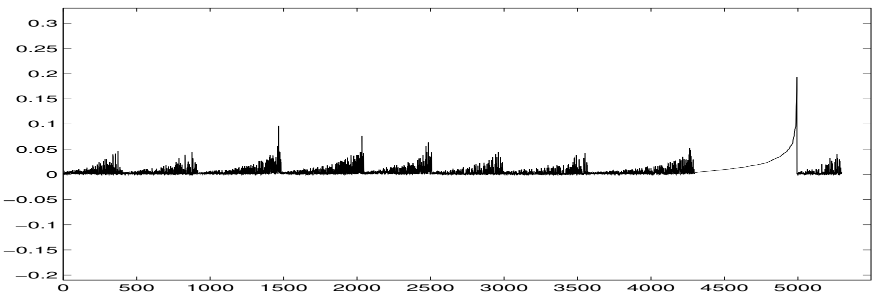
cells	0.235343
protein	0.217492
cell	0.201709
proteins	0.198059
gene	0.173822
genes	0.162417
expression	0.134430



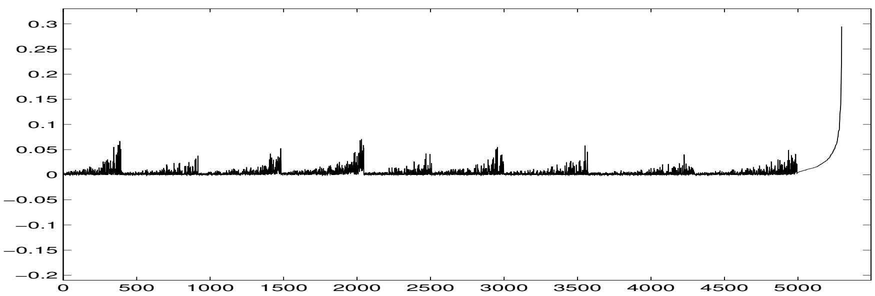
ocean	0.209621
ice	0.173253
climate	0.158356
mantle	0.139116
sea	0.117656
water	0.116900
carbon	0.109983



conference	0.218879
workshop	0.183514
international	0.156905
held	0.152462
social	0.149102
meeting	0.124209
scientists	0.121072



design	0.192935
algorithms	0.165257
control	0.147830
performance	0.141795
software	0.135452
parallel	0.121158
models	0.102727



chemistry	0.294680
reactions	0.222103
molecules	0.206988
organic	0.186004
nmr	0.173389
compounds	0.156620
chemical	0.140527

Figure 32: The concept vectors 6-10 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.

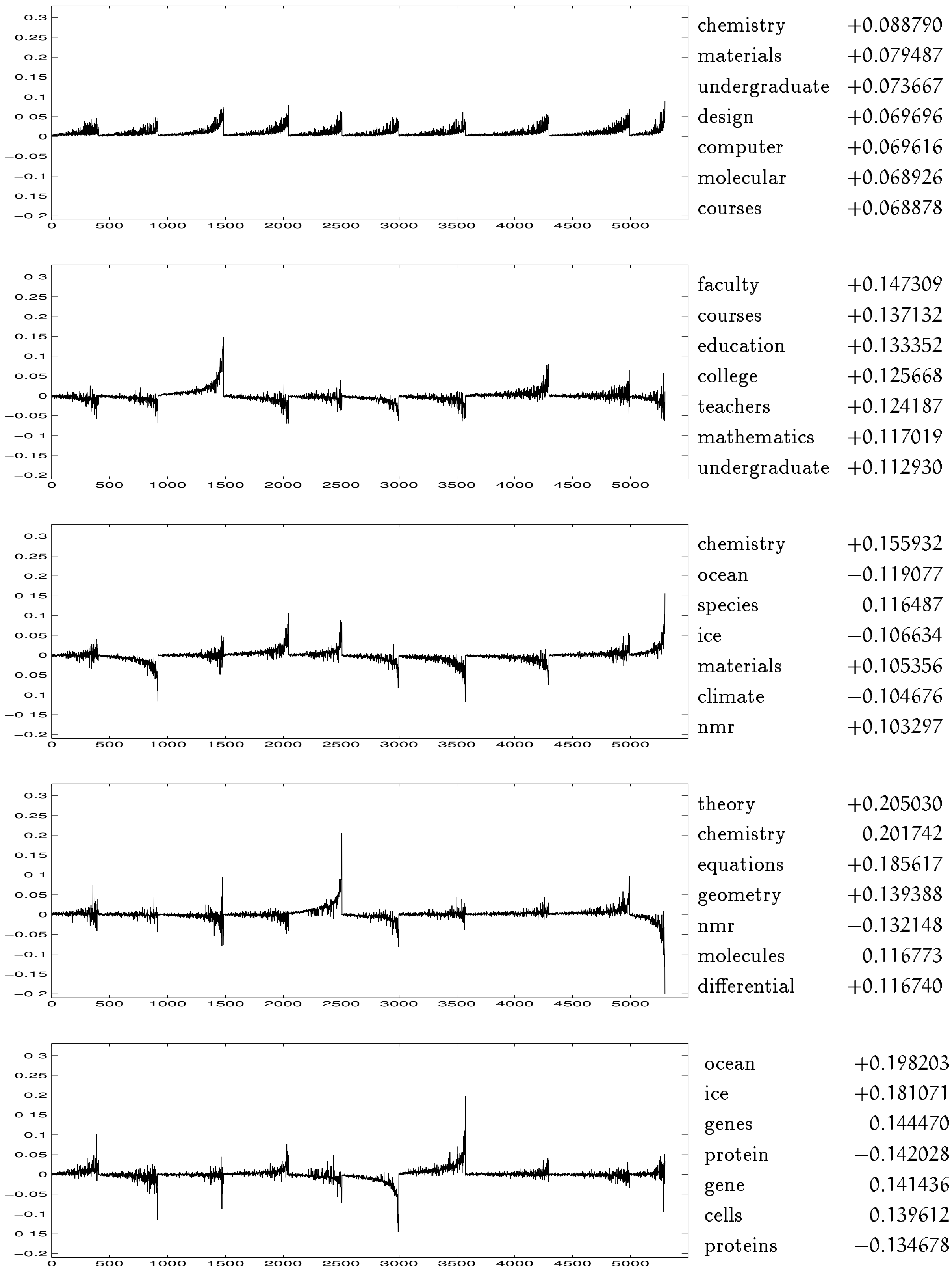
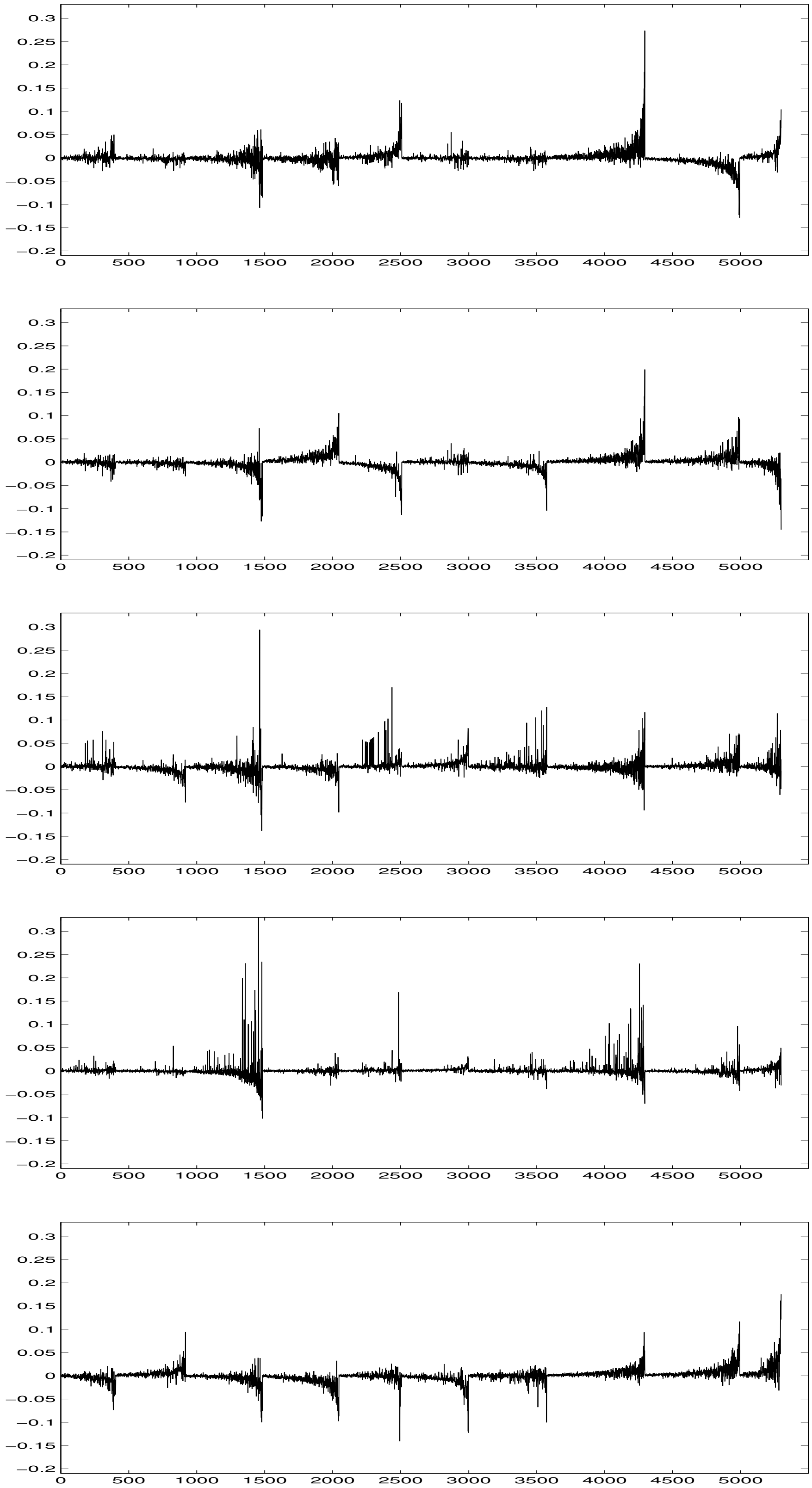


Figure 33: The singular vectors 1-5 for the NSF word-by-document matrix. For each singular vector, the top seven words with the corresponding weights are shown on the right.



conference	+0.273371
held	+0.184812
workshop	+0.151084
international	+0.150507
meeting	+0.148724
design	−0.128530
scientists	+0.124317

conference	+0.199307
workshop	+0.147236
chemistry	−0.145337
held	+0.141282
mathematics	−0.127382
international	+0.125168
meeting	+0.120061

equipment	+0.293889
shipboard	+0.170267
teachers	−0.138021
ocean	+0.127893
vessel	+0.120406
conference	+0.116393
instrumentation	+0.114066

internet	+0.445364
college	+0.234226
connection	+0.231291
connections	+0.230781
connect	+0.199320
access	+0.173787
partial	+0.168797

chemistry	+0.175281
nmr	+0.161630
physics	−0.140994
reactions	+0.126851
cells	−0.122399
organic	+0.122119
molecules	+0.121044

Figure 34: The singular vectors 6-10 for the NSF word-by-document matrix. For each singular vector, the top seven words with the corresponding weights are shown on the right.

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. A further evidence of self-similarity is provided by our plots of the concept vectors (Figures 13, 31, and 32) 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 clustering can be used for the more basic task of matrix approximation. In particular, we proposed concept decompositions that compute the least-squares projection of each document vector onto the linear subspace spanned by the concept vectors. We empirically established the surprising fact that the approximation power of concept decompositions is comparable to that of truncated SVDs (Figures 11 and 12). Furthermore, we empirically demonstrate that the subspaces spanned by concept vectors are quite close to the subspaces spaned by the singular vectors (Figures 19–30). 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, by definition, global (Figures 13, 14, 31, 32, and 33, 34). 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 13, 31, and 32. Furthermore, unlike singular vectors, the concept vectors are sparse (Figures 15 and 16), 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, while the singular vectors to Fourier series.

References

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

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

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

[Broder et al., 1997] Broder, A. Z., Glassman, S. C., Manasse, M. S., and Zweig, G. (1997). Syntactic clustering of the web. Technical Report 1997-015, Digital Systems Research Center. http://ftp.digital.com/pub/DEC/SRC/technical-notes/SRC-1997-015-html/.

[Caid and Oing, 1997] Caid, W. R. and Oing, P. (1997). System and method of context vector generation and retrieval. US Patent No. 5619709.

[Cutting et al., 1992] Cutting, D. R., Karger, D. R., Pedersen, J. O., and Tukey, J. W. (1992). Scatter/gather: A cluster-based approach to browsing large document collections. In *ACM SIGIR*.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

[Dhillon and Modha, 1999] Dhillon, I. S. and Modha, D. S. (1999). A parallel data-clustering algorithm for distributed memory multiprocessors. In *Proceedings of the Large-Scale Parallel KDD Systems Workshop, Fifth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, San Diego, CA*. Also appears as Technical Report RJ 10134, IBM Almaden Research Center, 1998.

[Dhillon et al., 1998] Dhillon, I. S., Modha, D. S., and Spangler, W. S. (1998). Visualizing class structure of multidimensional data. In Weisberg, S., editor, *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics, Minneapolis, MN.*

[Duda and Hart, 1973] Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis.* Wiley.

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

[Gallant, 1994] Gallant, S. I. (1994). Methods for generating or revising context vectors for a plurality of word stems. US Patent No. 5325298.

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

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

[Hartigan, 1975] Hartigan, J. A. (1975). *Clustering Algorithms.* Wiley.

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

[Hofmann, 1999] Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval.* ACM Press.

[Isbell and Viola, 1998] Isbell, C. L. and Viola, P. (1998). Restructuring sparse high dimensional data for effective retrieval. In *Advances in Neural Information Processing*, volume 11.

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

[Kolda, 1997] Kolda, T. G. (1997). *Limited-Memory Matrix Methods with Applications.* PhD thesis, The Applied Mathematics Program, University of Maryland, College Park, Mayland.

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

[Mandelbrot, 1988] Mandelbrot, B. B. (1988). *Fractal Geometry of Nature.* W. H. Freeman & Company.

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

[Papadimitriou et al., 1998] Papadimitriou, C. H., Raghavan, P., Tamaki, H., and Vempala, S. (1998). Latent semantic indexing: A probabilistic analysis. In *Proceedings of the Seventeenth ACM-SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Seattle, Washington*, pages 159–168.

[Pollard, 1982] Pollard, D. (1982). Quantization and the method of k-means. *IEEE Trans. Inform. Theory*, 28:199–205.

[Rasmussen, 1992] Rasmussen, E. (1992). Clustering algorithms. In Frakes, W. B. and Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms*, pages 419–442. Prentice Hall, Englewood Cliffs, New Jersey.

[Rissanen et al., 1992] Rissanen, J., Speed, T., and Yu, B. (1992). Density estimation by stochastic complexity. *IEEE Trans. Inform. Theory*, 38:315–323.

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

[Sahami et al., 1999] Sahami, M., Yusufali, S., and Baldonado, M. (1999). SONIA: A service for organizing networked information autonomously. In *Digital Libraries 98: Proceedings of the Third ACM Conference on Digital Libraries, New York, NY*, pages 200–209. ACM.

[Salton and Buckley, 1988] Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 4(5):513:523.

[Salton and McGill, 1983] Salton, G. and McGill, M. J. (1983). *Introduction to Modern Retrieval.* McGraw-Hill Book Company.

[Schütze and Silverstein, 1997] Schütze, H. and Silverstein, C. (1997). Projections for efficient document clustering. In *ACM SIGIR*.

[Silverstein and Pedersen, 1997] Silverstein, C. and Pedersen, J. O. (1997). Almost-constant-time clustering of arbitrary corpus subsets. In *ACM SIGIR*.

[Singhal et al., 1996] Singhal, A., Buckley, C., Mitra, M., and Salton, G. (1996). Pivoted document length normalization. In *ACM SIGIR*.

[Vaithyanathan and Dom, 1999] Vaithyanathan, S. and Dom, B. (1999). Model selection in unsupervised learning with applications to document clustering. In *International Conference on Machine Learning*.

[Willet, 1988] Willet, P. (1988). Recent trends in hierarchic document clustering: a critical review. *Information Processing & Management*, 24(5):577–597.

[Zamir and Etzioni, 1998] Zamir, O. and Etzioni, O. (1998). Web document clustering: A feasibility demonstration. In *ACM SIGIR*.

[Zipf, 1949] Zipf, G. K. (1949). *Human Behavior and the Principle of Least Effort.* Addison Wesley, Reading, MA.