Concept Indexing
A Fast Dimensionality Reduction Algorithm with Applications to Document Retrieval & Categorization[*]

George Karypis and Eui-Hong (Sam) Han

University of Minnesota, Department of Computer Science / Army HPC Research Center
Minneapolis, MN 55455
Technical Report: #00-016

{karypis, han}@cs.umn.edu

Last updated on March 6, 2000 at 12:28am

Abstract

In recent years, we have seen a tremendous growth in the volume of text documents available on the Internet, digital libraries, news sources, and company-wide intranets. This has led to an increased interest in developing methods that can efficiently categorize and retrieve relevant information. Retrieval techniques based on dimensionality reduction, such as Latent Semantic Indexing (LSI), have been shown to improve the quality of the information being retrieved by capturing the latent meaning of the words present in the documents. Unfortunately, the high computational requirements of LSI and its inability to compute an effective dimensionality reduction in a supervised setting limits its applicability. In this paper we present a fast dimensionality reduction algorithm, called ***concept indexing*** (CI) that is equally effective for unsupervised and supervised dimensionality reduction. CI computes a k-dimensional representation of a collection of documents by first clustering the documents into k groups, and then using the centroid vectors of the clusters to derive the axes of the reduced k-dimensional space. Experimental results show that the dimensionality reduction computed by CI achieves comparable retrieval performance to that obtained using LSI, while requiring an order of magnitude less time. Moreover, when CI is used to compute the dimensionality reduction in a supervised setting, it greatly improves the performance of traditional classification algorithms such as C4.5 and kNN.

1 Introduction

The emergence of the World-Wide-Web has led to an exponential increase in the amount of documents available electronically. At the same time, various digital libraries, news sources, and company-wide intranets provide huge collections of online documents. It has been forecasted that text (with other unstructured data) will become the

[*]This work was supported by NSF CCR-9972519, by Army Research Office contract DA/DAAG55-98-1-0441, by the DOE ASCI program, and by Army High Performance Computing Research Center contract number DAAH04-95-C-0008. Access to computing facilities was provided by AHPCRC, Minnesota Supercomputer Institute. Related papers are available via WWW at URL: **http://www.cs.umn.edu/˜karypis**

predominant data type stored online [61]. These developments have led to an increased interest in methods that allow users to quickly and accurately retrieve and organize these types of information.

Traditionally, information has been retrieved by literally matching terms in documents with those present in a user's query. Unfortunately, methods that are based only on lexical matching can lead to poor retrieval performance due to two effects. First, because most terms have multiple meanings, many unrelated documents may be included in the answer set just because they matched some of the query terms. Second, because the same concept can be described by multiple terms, relevant documents that do not contain any of the query terms will not be retrieved. These problems arise from the fact that the ideas in a document are more related to the concepts described in them than the words used in their description. Thus, effective retrieval methods should match the concept present in the query to the concepts present in the documents. This will allow retrieval of documents that are part of the desired concept even when they do not contain any of the query terms, and will prevent documents belonging to unrelated concepts from being retrieved even if they contain some of the query terms.

This concept-centric nature of documents is also one of the reasons why the problem of document categorization (*i.e.*, assigning a document into a pre-determined class or topic) is particularly challenging. Over the years a variety of document categorization algorithms have been developed [12, 22, 50, 33, 42, 3, 69, 45, 25], both from the machine learning as well as from the Information Retrieval (IR) community. A surprising result of this research has been that naive Bayesian, a relatively simple classification algorithm, performs well [47, 48, 46, 54, 17] for document categorization, even when compared against other algorithms that are capable of learning substantially more complex models. Some of this robust performance can be attributed to the fact that naive Bayesian is able to model the underlying concepts present in the various classes by summarizing the characteristics of each class using a probabilistic framework, and thus it can exploit the concept-centric nature of the documents.

Recently, techniques based on *dimensionality reduction* have been explored for capturing the concepts present in a collection. The main idea behind these techniques is to map each document (and a query or a test document) into a lower dimensional space that explicitly takes into account the dependencies between the terms. The associations present in the lower dimensional representation can then be used to improve the retrieval or categorization performance. The various dimensionality reduction techniques can be classified as either *supervised* or *unsupervised*. Supervised dimensionality reduction refers to the set of techniques that take advantage of class-membership information while computing the lower dimensional space. These techniques are primarily used for document classification and for improving the retrieval performance of pre-categorized document collections. Examples of such techniques include a variety of feature selection schemes [2, 37, 40, 38, 70, 28, 66, 56, 51] that reduce the dimensionality by selecting a subset of the original features, and techniques that create new features by clustering the terms [3]. On the other hand, unsupervised dimensionality reduction refers to the set of techniques that compute a lower dimensional space without using any class-membership information. These techniques are primarily used for improving the retrieval performance, and to a lesser extent for document categorization. Examples of such techniques include Principal Component Analysis (PCA) [30], Latent Semantic Indexing (LSI) [15, 5, 19], Kohonen Self-Organizing Map (SOFM) [39] and Multi-Dimensional Scaling (MDS) [31]. In the context of document data sets, LSI is probably the most widely used of these techniques, and experiments have shown that it significantly improves the retrieval performance [5, 19] for a wide variety of document collections.

In this paper we present a new fast dimensionality reduction algorithm, called ***concept indexing*** (CI) that can be used both for supervised and unsupervised dimensionality reduction. The key idea behind this dimensionality reduction scheme is to express each document as a function of the various concepts present in the collection. This is achieved by first finding groups of similar documents, each group potentially representing a different concept in the collection, and then using these groups to derive the axes of the reduced dimensional space. In the case of supervised dimensionality reduction, CI finds these groups from the pre-existing classes of documents, whereas in the case of unsupervised dimensionality reduction, CI finds these groups by using a document clustering algorithm. These clusters are found using a near linear time clustering algorithm which contributes to CI's low computational requirement.

We experimentally evaluate the quality of the lower dimensional space computed by CI on a wide range of data sets both in an unsupervised and a supervised setting. Our experiments show that for unsupervised dimensionality reduction, CI achieves comparable retrieval performance to that obtained by LSI, while requiring an order of magnitude less time. In the case of supervised dimensionality reduction, our experiments show that the lower dimensional spaces computed by CI significantly improve the performance of traditional classification algorithms such as C4.5 [60] and k-nearest-neighbor [18, 14, 64]. In fact, the average classification accuracy over 21 data sets obtained by the k-nearest-neighbor algorithm on the reduced dimensional space is 5% higher than that achieved by a state-of-the-art implementation of the naive Bayesian algorithm [55].

The reminder of this paper is organized as follows. Section 2 provides a summary of the earlier work on dimensionality reduction. Section 3 describes the vector-space document model used in our algorithm. Section 4 describes the proposed concept indexing dimensionality reduction algorithm. Section 5 describes the clustering algorithm used by concept indexing. Section 6 provides the experimental evaluation of the algorithm. Finally, Section 7 offers some concluding remarks and directions for future research.

2 Previous Work

In this section, we briefly review some of the techniques that have been developed for unsupervised and supervised dimensionality reduction, which have been applied to document datasets.

Unsupervised Dimensionality Reduction There are several techniques for reducing the dimensionality of high-dimensional data in an unsupervised setting. Most of these techniques reduce the dimensionality by combining multiple variables or attributes utilizing the dependencies among the variables. Consequently, these techniques can capture synonyms in the document data sets. Unfortunately, the majority of these techniques tend to have large computational and memory requirements.

A widely used technique for dimensionality reduction is the Principal Component Analysis (PCA) [30]. Given an $n \times m$ document-term matrix, PCA uses the k-leading eigenvectors of the $m \times m$ covariance matrix as the axes of the lower k-dimensional space. These leading eigenvectors correspond to linear combinations of the original variables that account for the largest amount of term variability [30]. One disadvantage of PCA is that it has high memory and computational requirements. It requires $O(m^2)$ memory for the dense covariance matrix, and $\Omega(km^2)$ for finding the k leading eigenvectors [30]. These requirements are unacceptably high for document data sets, as the number of terms (m) is tens of thousands. Latent Semantic Indexing (LSI) [5] is a dimensionality reduction technique extensively used in the information retrieval domain and is similar in nature to PCA. LSI, instead of finding the truncated singular value decomposition of the covariance matrix, finds the truncated singular value decomposition of the original $n \times m$ document-term matrix, and uses these singular eigenvectors as the axes of the lower dimensional space. Since LSI does not require calculation of the covariance matrix, it has smaller memory and CPU requirements when n is less than m [30]. Experiments have shown that LSI substantially improves the retrieval performance on a wide range of data sets [19]. However, the reason for LSI's robust performance is not well understood, and is currently an active area of research [43, 57, 16, 27]. Other techniques include Kohonen Self-Organizing Feature Map (SOFM) [39] and

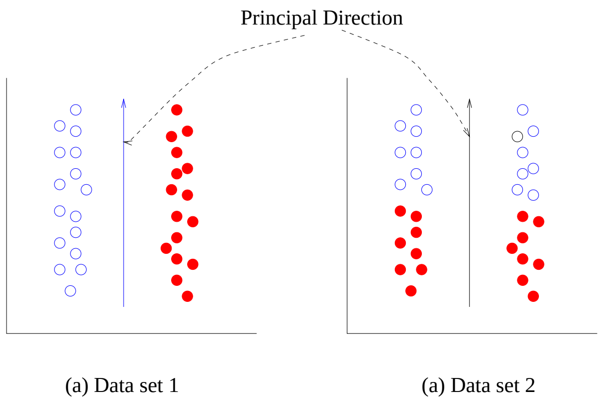

Figure 1: Problem of PCA or LSI in classification data sets.

Multidimensional Scaling (MDS) [31]. SOFM is a scheme based on neural networks that projects high dimensional input data into a feature map of a smaller dimension such that the proximity relationships among input data are preserved. MDS transforms the original data into a smaller dimensional space while trying to preserve the rank ordering of the distances among data points.

Supervised Dimensionality Reduction In principle, all of the techniques developed for unsupervised dimensionality reduction can potentially be used to reduce the dimensionality in a supervised setting as well. However, in doing so they cannot take advantage of the class or category information available in the data set. The limitations of these approaches in a supervised setting are illustrated in the classical example shown in Figure 1. In these data sets, the principle direction computed by LSI or PCA will be the same, as it is the direction that has the most variance. The projection of the first data set onto this principal direction will lead to the worst possible classification, whereas the projection of the second data set will lead to a perfect classification. Another limitation of these techniques in supervised data is that characteristic variables that describe smaller classes tend to be lost as a result of the dimensionality reduction. Hence, the classification accuracy on the smaller classes can be bad in the reduced dimensional space.

In general, supervised dimensionality reduction has been performed by using various feature selection techniques [2, 37, 40, 38, 70, 28, 66, 56, 51]. These techniques can be broadly classified into two groups, commonly referred to as the *filter-* [38] and *wrapper-based* [38, 64] approaches. In the filter-based approaches, the different features are ranked using a variety of criteria, and then only the highest-ranked features are kept. A variety of techniques have been developed for ranking the features (*i.e.*, words in the collection) including document frequency (number of documents in which a word occurs), mutual information [9, 70, 32, 54], and χ^2 statistics [70]. The main disadvantage of the filter-based approaches is that the features are selected independent of the actual classification algorithm that will be used [38]. Consequently, even though the criteria used for ranking measure the effectiveness of each feature in the classification task, these criteria may not be optimal for the classification algorithm used. Another limitation of this approach is that these criteria measure the effectiveness of a feature independent of other features, and hence features that are effective in classification in conjunction with other features will not be selected. In contrast to the filter-based approaches, wrapper-based schemes find a subset of features using a classification algorithm as a black box [38, 51, 36, 41]. In this approach the features are selected based on how well they improve the classification accuracy of the algorithm used. The wrapper-based approaches have been shown to be more effective than the filter-based approaches in many applications [38, 64, 44]. However, the major drawback of these approaches is that their computational requirements are very high [36, 41, 36, 41]. This is particularly true for document data sets in which the features number in the thousands.

Baker and McCallum recently proposed a dimensionality reduction technique based on Distributional Clustering [58] of words [3]. This technique clusters words into groups based on the distribution of class labels associated with each word. Words that have similar class distribution, given a particular word, are grouped into a cluster. Conditional probability of classes, given set of words, are computed by the weighted average of the conditional probability of classes of individual probability of words. By clustering words that have similar class distributions, this technique can potentially identify words that have synonyms. However, since a word can only belong to one cluster, polysemous words will not be identified.

3 Vector-Space Modeling of Documents

In the CI dimensionality reduction algorithm, the documents are represented using the vector-space model [62]. In this model, each document d is considered to be a vector in the term-space. In its simplest form, each document is represented by the *term-frequency* (TF) vector $\vec{d}_{tf} = (tf_1, tf_2, \ldots, tf_n)$, where tf_i is the frequency of the ith term in the document. A widely used refinement to this model is to weight each term based on its *inverse document frequency* (IDF) in the document collection. The motivation behind this weighting is that terms appearing frequently in many documents have limited discrimination power, and for this reason they need to be de-emphasized. This is commonly done [35, 62] by multiplying the frequency of each term i by $\log(N/df_i)$, where N is the total number of documents in the collection, and df_i is the number of documents that contain the ith term (*i.e.*, document frequency). This leads to the *tf-idf* representation of the document, *i.e.*, $\vec{d}_{tfidf} = (tf_1 \log(N/df_1), tf_2 \log(N/df_2), \ldots, tf_n \log(N/df_n))$. Finally, in order to account for documents of different lengths, the length of each document vector is normalized so that it is of unit length, *i.e.*, $\|\vec{d}_{tfidf}\|_2 = 1$. In the rest of the paper, we will assume that the vector representation $\vec{d}$ of each document d has been weighted using *tf-idf* and it has been normalized so that it is of unit length.

In the vector-space model, the similarity between two documents d_i and d_j is commonly measured using the cosine function [62], given by

$$\cos(\vec{d}_i, \vec{d}_j) = \frac{\vec{d}_i \cdot \vec{d}_j}{\|\vec{d}_i\|_2 * \|\vec{d}_j\|_2}, \tag{1}$$

where "$\cdot$" denotes the dot-product of the two vectors. Since the document vectors are of unit length, the above formula is simplified to $\cos(\vec{d}_i, \vec{d}_j) = \vec{d}_i \cdot \vec{d}_j$.

Given a set S of documents and their corresponding vector representations, we define the ***centroid*** vector $\vec{C}$ to be

$$\vec{C} = \frac{1}{|S|} \sum_{d \in S} \vec{d}, \tag{2}$$

which is the vector obtained by averaging the weights of the various terms in the document set S. We will refer to S as the ***supporting set*** for the centroid $\vec{C}$. Analogously to individual documents, the similarity between a document d and a centroid vector $\vec{C}$ is computed using the cosine measure, as follows:

$$\cos(\vec{d}, \vec{C}) = \frac{\vec{d} \cdot \vec{C}}{\|\vec{d}\|_2 * \|\vec{C}\|_2} = \frac{\vec{d} \cdot \vec{C}}{\|\vec{C}\|_2}. \tag{3}$$

Note that even though the document vectors are of length one, the centroid vectors will not necessarily be of unit length.

Intuitively, this document-to-centroid similarity function tries to measure the similarity between a document and the documents belonging to the supporting set of the centroid. A careful analysis of Equation 3 reveals that this similarity captures a number of interesting characteristics. In particular, the similarity between $\vec{d}$ and $\vec{C}$ is the ratio of the dot-product between $\vec{d}$ and $\vec{C}$, divided by the length of $\vec{C}$. If S is the supporting set for $\vec{C}$, then it can be easily shown [11, 24] that

$$\vec{d} \cdot \vec{C} = \frac{1}{|S|} \sum_{x \in S} \cos(\vec{d}, \vec{x}),$$

and that

$$\|\vec{C}\|_2 = \sqrt{\frac{1}{|S|^2} \sum_{d_i \in S} \sum_{d_j \in S} \cos(\vec{d_i}, \vec{d_j})}. \tag{4}$$

Thus, the dot-product is the average similarity between d and all other documents in S, and the length of the centroid vector is the square-root of the average pairwise similarity between the documents in S, including self-similarity. Note that because all the documents have been scaled to be of unit length, $\|\vec{C}\|_2 \leq 1$. Hence, Equation 3 measures the similarity between a document and the centroid of a set S, as the average similarity between the document and all the documents in S, amplified by a function that depends on the average pairwise similarity between the documents in S. If the average pairwise similarity is small, then the amplification is high, whereas if the average pairwise similarity is high, then the amplification is small. One of the important features of this amplification parameter is that it captures the degree of dependency between the terms in S [24]. In general, if S contains documents whose terms are positively dependent (*e.g.*, terms frequently co-occurring together), then the average similarity between the documents in S will tend to be high, leading to a small amplification. On the other hand, as the positive term dependency between documents in S decreases, the average similarity between documents in S tends to also decrease, leading to a larger amplification. Thus, Equation 3 computes the similarity between a document and a centroid, by both taking into account the similarity between the document and the supporting set, as well as the dependencies between the terms in the supporting set.

4 Concept Indexing

The concept indexing algorithm computes a lower dimensional space by finding groups of similar documents and using them to derive the axes of the lower dimensional space. In the rest of this section we describe the details of the CI dimensionality reduction algorithm for both an unsupervised and a supervising setting, and analyze the nature of its lower dimensional representation.

4.1 Unsupervised Dimensionality Reduction

CI computes the reduced dimensional space in the unsupervised setting as follows. If k is the number of desired dimensions, CI first computes a k-way clustering of the documents (using the algorithm described in Section 5), and then uses the centroid vectors of the clusters as the axes of the reduced k-dimensional space. In particular, let D be an $n \times m$ document-term matrix, (where n is the number of documents, and m is the number of distinct terms in the collection) such that the ith row of D stores the vector-space representation of the ith document (*i.e.*, $D[i, *] = \vec{d_i}$). CI uses a clustering algorithm to partition the documents into k disjoint sets, $S_1, S_2, \ldots, S_k$. Then, for each set S_i, it computes the corresponding centroid vector $\vec{C_i}$ (as defined by Equation 2). These centroid vectors are then scaled so that they have unit length. Let $\{\vec{C'}_1, \vec{C'}_2, \ldots, \vec{C'}_k\}$ be these unit length centroid vectors. Each of these vectors form one of the axis of the reduced k-dimensional space, and the k-dimensional representation of each document is obtained by projecting it onto this space. This projection can be written in matrix notation as follows. Let C be the $m \times k$ matrix such that the ith column of C corresponds to $\vec{C'}_i$. Then, the k-dimensional representation of each document $\vec{d}$ is given by $\vec{d}C$, and the k-dimensional representation of the entire collection is given by the matrix $D_k = DC$. Similarly, the k-dimensional representation of a query $\vec{q}$ for a retrieval is given by $\vec{q}C$. Finally, the similarity between two documents in the reduced dimensional space is computed by calculating the cosine between the reduced dimensional vectors.

4.2 Supervised Dimensionality Reduction

In the case of supervised dimensionality reduction, CI uses the pre-existing clusters of documents (*i.e.*, the classes or topics in which the documents belong to) in finding the groups of similar documents. In the simplest case, each one of these groups corresponds to one of the classes in the data set. In this case, the rank of the lower dimensional space will be identical to the number of classes. A lower dimensional space with a rank k that is greater than the number of classes, l, is computed as follows. CI initially computes an l-way clustering by creating a cluster for each one of the document classes, and then uses a clustering algorithm to obtain a k-way clustering by repeatedly partitioning some of these clusters. Note that in the final k-way clustering, each one of these finer clusters will contain documents from only one class. The reverse of this approach can be used to compute a lower dimensional space that has a rank that is smaller than the number of distinct classes, by repeatedly combining some of the initial clusters using an agglomerative clustering algorithm. However, this lower dimensional space tend to lead to poor classification performance as it combines together potentially different concepts, and is not recommended. Note that once these clusters have been identified, then the algorithm proceeds to compute the lower dimensional space in the same fashion as in the unsupervised setting (Section 4.1).

As discussed in Section 1, supervised dimensionality reduction is particularly useful to improve the retrieval performance in a pre-categorized document collection, or to improve the accuracy of document classification algorithms. Experiments presented in Section 6.3 show that the performance of traditional classification algorithms, such as C4.5 [60] and k-nearest-neighbor improves dramatically in the reduced space found by CI.

4.3 Analysis & Discussion

In order to understand this dimensionality reduction scheme, it is necessary to understand two things. First, we need to understand what is encapsulated within the centroid vectors, and second, we need to understand the meaning of the reduced dimensional representation of each document. For the rest of this discussion we will assume that we have a clustering algorithm that returns k reasonably good clusters [11, 45, 7], given a set of documents. By that we mean that each one of the clusters tends to contain similar documents, and documents belonging to different clusters are less similar than those belonging to the same cluster.

Given a set of documents, the centroid vector provides a mechanism to summarize their content. In particular, the prominent dimensions of the vector (*i.e.*, terms with the highest weights), correspond to the terms that are most important within the set. Two examples of such centroid vectors for two different collections of documents are shown in Table 1 (these collections are described in Section 6.1). For each collection we computed a 20-way clustering, and for each of the clusters we computed their unit-length scaled centroid vectors. For each of these vectors, Table 1 shows the ten highest weight terms. The number that precedes each term in this table is the weight of that term in the centroid vector. Also note that the terms shown in this table are not the actual words, but their stems.

A number of observations can be made by looking at the terms present in the various centroids. First, looking at the weight of the various terms, we can see that for each centroid, there are relatively few terms that account for a large fraction of its length. To further illustrate this, we computed the fraction of the centroid length for which these

	re1										
1	0.65 corn	0.20 acre	0.19 bushel	0.18 soybean	0.17 usda	0.17 unknown	0.16 ussr	0.16 tonne	0.13 report	0.12 export	67%
2	0.46 ga	0.24 oil	0.22 cubic	0.21 reserv	0.20 barrel	0.20 feet	0.19 natur	0.15 drill	0.15 mln	0.14 lt	54%
3	0.65 coffee	0.28 quota	0.27 ico	0.17 bag	0.16 export	0.16 brazil	0.14 colombia	0.14 meet	0.13 produc	0.12 ibc	72%
4	0.45 tonne	0.35 palm	0.20 import	0.18 oil	0.15 januari	0.14 rapese	0.14 beef	0.14 februari	0.13 mln	0.13 export	51%
5	0.35 copper	0.30 steel	0.20 ct	0.19 aluminium	0.16 cent	0.15 smelter	0.14 pound	0.14 lb	0.14 price	0.14 alcan	42%
6	0.32 crop	0.24 grain	0.20 wheate	0.19 cotton	0.19 mln	0.19 weather	0.16 china	0.16 rain	0.15 plant	0.15 tonne	40%
7	0.45 bble	0.39 crude	0.31 post	0.26 ct	0.22 dlr	0.21 wti	0.20 raise	0.16 distill	0.16 price	0.15 gasolin	72%
8	0.45 dollar	0.28 bank	0.24 portland	0.23 yen	0.17 load	0.16 juice	0.16 ship	0.14 japan	0.13 orang	0.12 dealer	52%
9	0.73 sugar	0.22 tonne	0.22 white	0.15 trader	0.14 intervent	0.14 ec	0.13 tender	0.12 ecu	0.12 rebat	0.11 cargoe	75%
10	0.59 gold	0.35 ounce	0.33 ton	0.30 mine	0.14 ore	0.12 feet	0.12 silver	0.10 assai	0.09 reserv	0.09 coin	74%
11	0.49 ec	0.34 maize	0.24 tax	0.20 tonne	0.17 european	0.17 licenc	0.17 ecu	0.16 commiss	0.16 barlei	0.14 commun	61%
12	0.30 wheate	0.27 soviet	0.22 farm	0.22 lyng	0.21 bill	0.19 offer	0.18 grain	0.15 agricultur	0.14 eep	0.13 loan	43%
13	0.39 cocoa	0.35 buffer	0.26 deleg	0.24 rubber	0.22 stock	0.22 icco	0.17 pact	0.17 consum	0.14 rule	0.13 council	59%
14	0.32 ship	0.24 gulf	0.22 tanker	0.22 iran	0.21 missil	0.18 vessel	0.15 attack	0.14 iranian	0.13 sea	0.13 line	41%
15	0.43 oil	0.29 tax	0.18 herrington	0.17 explor	0.16 energi	0.15 import	0.12 reagan	0.12 studi	0.12 industri	0.11 petroleum	43%
16	0.28 credit	0.28 wheate	0.25 ccc	0.24 depart	0.22 nil	0.19 sale	0.18 commod	0.18 guarante	0.18 bonu	0.17 mln	49%
17	0.43 ecuador	0.27 bpd	0.27 refineri	0.25 crude	0.25 oil	0.21 pipelin	0.20 venezuela	0.13 mln	0.12 barrel	0.12 energi	59%
18	0.43 wheate	0.42 tonne	0.24 tender	0.24 barlei	0.22 taiwan	0.18 shipment	0.15 soft	0.14 export	0.14 home	0.13 deliveri	64%
19	0.48 strike	0.28 seamen	0.28 union	0.25 port	0.22 worker	0.14 employ	0.13 ship	0.12 pai	0.11 spokesman	0.11 talk	57%
20	0.49 opec	0.31 saudi	0.27 oil	0.25 bpd	0.24 barrel	0.18 mln	0.17 price	0.15 arabia	0.14 crude	0.12 al	65%

	new3										
1	0.25 russian	0.19 russia	0.18 rwanda	0.17 moscow	0.14 soviet	0.14 rebel	0.13 nato	0.13 un	0.13 tass	0.12 militari	26%
2	0.41 vw	0.30 lopez	0.24 iraq	0.23 gm	0.20 matrix	0.19 opel	0.18 inquiri	0.18 churchill	0.16 volkswagen	0.16 scot	56%
3	0.15 econom	0.15 export	0.14 percent	0.12 enterpris	0.12 russian	0.11 reform	0.11 product	0.11 economi	0.10 social	0.10 russia	15%
4	0.26 tunnel	0.19 rail	0.16 argentina	0.15 school	0.14 curriculum	0.14 eurotunnel	0.14 british	0.14 pound	0.14 channel	0.14 labour	27%
5	0.39 hyph	0.29 food	0.22 blank	0.19 label	0.16 fda	0.14 fsi	0.14 speci	0.14 poultri	0.14 cfr	0.12 stag	44%
6	0.71 drug	0.21 patient	0.16 azt	0.14 aid	0.14 fda	0.12 addict	0.10 epo	0.09 treatment	0.08 amgen	0.08 hiv	66%
7	0.46 korea	0.33 north	0.32 nuclear	0.31 iaea	0.28 korean	0.21 dprk	0.18 inspect	0.16 pyongyang	0.15 seoul	0.10 sanction	73%
8	0.52 tax	0.28 bank	0.24 cent	0.23 pound	0.17 incom	0.16 vate	0.15 rate	0.12 taxe	0.11 financ	0.11 imf	57%
9	0.28 japan	0.25 vietnam	0.24 china	0.23 trade	0.22 rice	0.19 japanes	0.17 gat	0.15 tokyo	0.12 vietnames	0.11 import	41%
10	0.59 women	0.47 violenc	0.19 domest	0.15 crime	0.13 speaker	0.12 victim	0.12 abus	0.10 batter	0.10 bill	0.10 sexual	70%
11	0.26 helmslei	0.24 hunter	0.20 tax	0.18 fraud	0.17 evasion	0.16 dominelli	0.15 rose	0.15 sentenc	0.13 guilti	0.13 juri	33%
12	0.38 al	0.24 palestinian	0.23 arab	0.22 israe	0.18 israel	0.17 islam	0.16 lebanon	0.14 kill	0.13 terrorist	0.11 afp	44%
13	0.35 cent	0.24 compani	0.21 dollar	0.18 pound	0.16 pharmaceut	0.16 price	0.14 pulp	0.13 paper	0.12 sale	0.12 market	37%
14	0.43 kong	0.43 hong	0.22 chines	0.21 china	0.20 beij	0.18 journalist	0.16 taiwan	0.15 yang	0.14 mainland	0.13 qiandao	62%
15	0.47 grain	0.34 agricultur	0.23 price	0.19 rural	0.18 product	0.17 percent	0.15 yuan	0.15 farm	0.14 farmer	0.14 peasant	57%
16	0.62 nuclear	0.30 pakistan	0.23 india	0.18 weapon	0.17 ukrain	0.15 plutonium	0.12 treati	0.12 prolifer	0.12 reactor	0.12 japan	67%
17	0.38 nafta	0.33 mexico	0.17 mexican	0.17 speaker	0.16 american	0.16 trade	0.16 gentleman	0.16 job	0.13 rep	0.12 house	44%
18	0.24 polic	0.17 kill	0.16 anc	0.15 murder	0.14 africa	0.11 offic	0.10 death	0.10 journalist	0.10 african	0.10 johannesburg	21%
19	0.47 drug	0.34 traffick	0.25 cocain	0.20 cartel	0.20 colombia	0.17 colombian	0.17 polic	0.13 arrest	0.12 spanish	0.12 narcot	58%
20	0.24 water	0.24 forest	0.22 environment	0.21 river	0.21 project	0.16 pollution	0.16 amazon	0.14 power	0.13 gorge	0.13 energi	36%

Table 1: The ten highest weight terms in the centroids of the clusters of two data sets.

terms are responsible. This is shown in the last column of each table. For example, the highest ten terms for the first centroid of *re1* account for 67% of its length, for the second centroid account for 54% of its length, and so for. Thus, each centroid can be described by a relative small number of *keyword* terms. This is a direct consequence of the fact that the supporting sets for each centroid correspond to clusters of similar documents, and not just random subsets of documents. Second, these terms are quite effective in providing a summary of the topics discussed within the documents, and their weights provide an indication of how central they are in these topics. For example, looking at the centroids for *re1*, we see that the first cluster contains documents that talk about the export of agricultural products to USSR, the second cluster contains energy related documents, the third cluster contains documents related to coffee production, and so on. This feature of centroid vectors has been used successfully in the past to build very accurate summaries [11, 45], and to improve the performance of clustering algorithms [1]. Third, the prevalent terms of the various centroids often contain terms that act as synonyms within the context of the topic they describe. This can easily be seen in some of the clusters for *new3*. For example, the terms *russian* and *russia* are present in the first centroid, the terms *vw* and *volkswagen* are present in the second centroid, and the terms *drug* and *narcot* are present in the nineteenth centroid. Note that these terms may not necessarily be present in a single document; however, such terms will easily appear in the centroid vectors if they are used interchangeably to describe the underlying topic. Fourth, looking at the various terms of the centroid vectors, we can see that the same term often appears in multiple centroids. This can easily happen when the supporting sets of the two centroids are part of the same topic, but it can also happen because many terms have multiple meanings (*polysemy*). For example, this happens in the case of the term *drug* in the sixth and nineteenth cluster of *new3*. The meaning of *drug* in the sixth cluster is that of prescription drugs, whereas the meaning of *drug* in the nineteenth cluster is that of narcotics. This polysemy of terms can also be seen for the term *fda*, that is the abbreviation of the Food & Drug Administration[1] that occurs in the fifth and sixth clusters of *new3*. The meaning of *fda* in the fifth cluster corresponds to the food-regulatory function of FDA (this can be inferred by looking at the other terms in the centroid such as *food*, *label*, *poultri*), whereas the meaning of *fda* in the sixth cluster corresponds to the drug-regulatory function of FDA (this can be inferred by looking at the other terms such as *drug*, *patient*, *azt*, *etc*.). To summarize, the centroid vectors provide a very effective mechanism to represent the *concepts* present in the supporting set of documents, and these vectors capture actual as well as latent associations between the terms that describe the concept.

Given a set of k centroid vectors and a document d, the ith coordinate of the reduced dimensional representation of this document is the similarity between document d and the ith centroid vector as measured by the cosine function (Equation 3). Note that this is consistent with the earlier definition (Section 4.1), in which the ith coordinate was defined as the dot-product between $\vec{d}$, and the unit-length normalized centroid vector $\vec{C'}_i$. Thus, the different dimensions of the document in the reduced space correspond to the degree at which each document matches the concepts that are encapsulated within the centroid vectors. This interpretation of the low dimensional representation of each document is the reason that we call our dimensionality reduction scheme ***concept indexing***. Note that documents that are close in the original space will also tend be close in the reduced space, as they will match the different concepts to the same degree. Moreover, because the centroids capture latent associations between the terms describing a concept, documents that are similar but are using somewhat different terms will be close in the reduced space even though they may not be close in the original space, thus improving the retrieval of relevant information. Similarly, documents that are close in the original space due to polysemous words, will be further apart in the reduced dimensional space; thus, eliminating incorrect retrievals. In fact, as our experiments in Section 6.2 show, CI is able to improve the retrieval performance, compared to that achieved in the original space.

5 Finding the Clusters

Over the years a variety of document clustering algorithms have been developed with varying time-quality trade-offs [11, 45]. Recently, partitional based document clustering algorithms have gained wide-spread acceptance as they provide reasonably good clusters and have a near-linear time complexity [11, 45, 1]. For this reason, the clustering algorithm we used in CI is derived from this general class of partitional algorithms.

Partitional clustering algorithms compute a k-way clustering of a set of documents either directly or via recursive bisection. A direct k-way clustering is computed as follows. Initially, a set of k documents is selected from the collection to act as the *seeds* of the k clusters. Then, for each document, its similarity to these k seeds is computed, and it is assigned to the cluster corresponding to its most similar seed. This forms the initial k-way clustering. This clustering is then repeatedly refined using the following procedure. First, the centroid vector for each cluster is computed, and then each document is assigned to the cluster corresponding to its most similar centroid. This refinement process terminates either after a predetermined small number of iterations, or after an iteration in which no document moved between clusters. A k-way partitioning via recursive bisection is obtained by recursively applying the above algorithm to compute 2-way clusterings (*i.e.*, bisections). Initially, the documents are partitioned into two clusters, then one of these clusters is selected and is further bisected, and so on. This process continues $k-1$ times, leading to k clusters.

A number of different schemes have been developed for selecting the initial set of *seed* documents [11, 20, 45]. A commonly used scheme is to select these seeds at random. In such schemes, a small number of different sets of

[1] For the non-US reader, FDA is responsible for regulating food products and prescription drugs within the US.

random seeds are often selected, a clustering solution is computed using each one of these sets, and the best of these solutions is selected as the final clustering. The quality of such partitional clusterings is evaluated by computing the similarity of each document to the centroid vector of the cluster that it belongs to. The best solution is the one that maximizes the sum of these similarities over the entire set of documents. CI's clustering algorithm uses this random seed approach, and selects the best solution obtained out of five random sets of seeds.

The CI algorithm computes a k-way clustering of the documents using recursive bisection. This approach gives a better control of the relative size of the clusters, as it tends to produce clusters whose sizes are not substantially different. This tends to lead to better dimensionality reductions for the following reason. Recall from Section 4, that CI uses the centroid vectors to represent the concepts present in the collection. Ideally, given a small number of dimensions, we would like to capture concepts that are present in a large number of documents. This is better achieved if the centroid vectors are obtained from larger clusters. We found in our experiments (which are not reported here) that a direct k-way clustering solution may sometimes create some very small clusters, as it tends to be more sensitive to outliers.

One of the key steps in any recursive bisection clustering algorithm is the scheme used to select which cluster to partition next. That is, given an l-way clustering solution, the algorithm must select one of these l clusters to bisect further, so that it will obtain the $(l+1)$-way clustering solution. A simple scheme will be to select the cluster that contains the largest number of documents. Unfortunately, even though this scheme tends to produce clusters whose size is not substantially different, in certain cases concepts may be over-represented in the final clustering. This will happen in cases in which the actual number of documents supporting the various concepts are of substantially different size. In such scenarios, bisecting the largest cluster can easily lead to a solution in which the *large* concepts are captured by multiple clusters, but the *smaller* concepts are completely lost. Ideally, we would like to bisect a cluster that contains a large number of dissimilar documents, as this will allow us to both capture different concepts, and at the same time ensure that these concepts are present in a large number of documents.

CI achieves this goal as follows. Recall from Section 3, that given a cluster S_i and its centroid vector $\vec{C_i}$, the square of the length of this vector (*i.e.*, $\|C_i\|_2{}^2$) measures the average pairwise similarity between the documents in S_i. Thus, we can look at $1 - \|C_i\|_2{}^2$ as a measure of the average pairwise dissimilarity. Furthermore the aggregate pairwise dissimilarity between the documents in the cluster is equal to

$$\text{Aggregate Dissimilarity} = |S_i|^2(1 - \|C_i\|_2{}^2). \tag{5}$$

CI uses this quantity in selecting the next cluster to bisect. In particular, CI bisects the cluster that has the highest aggregate dissimilarity over all the clusters.

The complexity of this clustering algorithm is $O(n \log k)$, where n is the number of documents and k is the number of clusters. Furthermore, for large document data sets such as WWW documents indexed by search engines, clustering algorithms [71, 8, 21] utilizing sampling, out-of-core techniques, and incremental clustering can be used to find clusters efficiently.

6 Experimental Results

In this section we experimentally evaluate the quality of the dimensionality reduction performed by CI. Two different sets of experiments are presented. The first set focuses on evaluating the document retrieval performance achieved by CI when used to compute the dimensionality reduction in an unsupervised setting, and its performance is compared against LSI. The second set of experiments focuses on evaluating the quality of the dimensionality reduction computed by CI in a supervised setting, both in terms of the document retrieval performance as well as in terms of the classification improvements achieved by traditional classification algorithms when operating in the reduced dimensional space. In all the experiments using LSI, we used the same unit length *tf-idf* document representation used by CI.

6.1 Document Collections

Data	Source	# of doc	# of class	min class size	max class size	avg class size	# of words
west1	West Group	500	10	39	73	50.0	977
west2	West Group	300	10	18	45	30.0	1078
west3	West Group	245	10	17	34	24.5	1035
oh0	OHSUMED-233445	1003	10	51	194	100.3	3182
oh5	OHSUMED-233445	918	10	59	149	91.8	3012
oh10	OHSUMED-233445	1050	10	52	165	105.0	3238
oh15	OHSUMED-233445	913	10	53	157	91.3	3100
ohscal	OHSUMED-233445	11162	10	709	1621	1116.2	11465
re0	Reuters-21578	1504	13	11	608	115.7	2886
re1	Reuters-21578	1657	25	10	371	66.3	3758
tr11	TREC	414	9	6	132	46.0	6429
tr12	TREC	313	8	9	93	39.1	5804
tr21	TREC	336	6	4	231	56.0	7902
tr31	TREC	927	7	2	352	132.4	10128
tr41	TREC	878	10	9	243	87.8	7454
tr45	TREC	690	10	14	160	69.0	8261
la1	TREC	3204	6	273	943	534.0	31472
la2	TREC	3075	6	248	905	512.5	31472
fbis	TREC	2463	17	38	506	144.9	2000
new3	TREC	9558	44	104	696	217.2	83487
wap	WebACE	1560	20	5	341	78.0	8460

Table 2: Summary of data sets used.

The characteristics of the various document collections used in our experiments are summarized in Table 2. The first three data sets are from the statutory collections of the legal document publishing division of West Group described in [10]. Data sets *tr11, tr12, tr21, tr31, tr41, tr45*, and *new3* are derived from TREC-5 [63], TREC-6 [63], and TREC-7 [63] collections. Data set *fbis* is from the Foreign Broadcast Information Service data of TREC-5 [63]. Data sets *la1*, and *la2* are from the Los Angeles Times data of TREC-5 [63]. The classes of the various *trXX*, *new3*, and *fbis* data sets were generated from the relevance judgment provided in these collections. The class labels of *la1* and *la2* were generated according to the name of the newspaper sections that these articles appeared, such as "Entertainment", "Financial", "Foreign", "Metro", "National", and "Sports". Data sets *re0* and *re1* are from Reuters-21578 text categorization test collection Distribution 1.0 [49]. We divided the labels into 2 sets and constructed data sets accordingly. For each data set, we selected documents that have a single label. Data sets *oh0, oh5, oh10, oh15*, and *ohscal* are from the OHSUMED collection [26] subset of MEDLINE database, which contains 233,445 documents indexed using 14,321 unique categories. We took different subsets of categories to construct these data sets. Data set *wap* is from the WebACE project (WAP) [56, 23, 6, 7]. Each document corresponds to a web page listed in the subject hierarchy of Yahoo! [67]. For all data sets, we used a stop-list to remove common words, and the words were stemmed using Porter's suffix-stripping algorithm [59].

6.2 Unsupervised Dimensionality Reduction

One of the goals of dimensionality reduction techniques such as CI and LSI is to project the documents of a collection onto a low dimensional space so that similar documents (*i.e.*, documents that are part of the same topic) come closer together, relative to documents belonging to different topics. This transformation, if successful, can lead to substantial improvements in the accuracy achieved by regular queries. The query performance is often measured by looking at the number of relevant documents present in the top-ranked returned documents. Ideally, a query should return most of the relevant documents (*recall*), and the majority of the documents returned should be relevant (*precision*). Unfortunately, a number of the larger collections in our experimental testbed did not have pre-defined queries associated with them, so we were not able to perform this type of evaluation. For this reason our evaluation was performed in terms of how effective the reduced dimensional space was in bringing closer together documents that belong to the same class.

To evaluate the extent to which a dimensionality reduction scheme is able to bring closer together similar documents, we performed the following experiment for each one of the data sets shown in Table 2. Let D be one of these datasets. For each document $d \in D$, we computed the k-nearest-neighbor sets both in the original as well as in the reduced dimensional space. Let K_d^o and K_d^r be these sets in the original and reduced space, respectively. Then, for each of these sets, we counted the number of documents that belong to the same class as d, and let n_d^o and n_d^r be these counts. Let $N_o = \sum_{d \in D} n_d^o$, and $N_r = \sum_{d \in D} n_d^r$, be the cumulative counts over all the documents in the data set. Given these two counts, then the performance of a dimensionality reduction scheme was evaluated by comparing N_r against N_o. In particular, if the ratio N_r / N_o is greater than one, then the reduced space was successful in bringing a larger number of similar documents closer together than they were in the original space, whereas if the ratio is less than one, then the reduced space is worse. We will refer to this ratio as the ***retrieval improvement*** (RI) achieved by the dimensionality reduction scheme.

An alternate way of interpreting this experiment is that for each document d, we perform a query using d as the query itself. In this context, the sets K_d^o and K_d^r are nothing more than the result of this query, the numbers n_d^o and n_d^r are a measure of the recall, and the numbers N_o and N_r are a measure of the cumulative recall achieved by performing as many queries as the total number of documents. Thus, retrieval performance increases as N_r increases, because both the recall, and because we compute the recall on a fixed size neighborhood, the precision also increases.

Table 3 shows the values for the RI measure obtained by both CI and LSI on the eight largest data sets in our testbed. The RI measure was computed using the 20-nearest-neighbors[2]. The first columns of these tables show the number of dimensions of the reduced space. For *re0, re1, la1, la2, fbis, wap,* and *ohscal* we used 10, 20, 30, 40, and 50 dimensions, whereas for *new3*, we used 25, 50, 75, 100, and 125. This is because, for the first seven data sets, the retrieval performance peaks at a smaller number of dimensions than does for *new3*.

Looking at these results we can see that the retrieval improvements achieved by CI are comparable to those achieved by LSI. Both schemes were able to achieve similar values for the RI measure, and both schemes compute spaces in which similar documents are closer together (the RI measures are greater than one in most of the experiments). CI does somewhat better for *la1, fbis,* and *ohscal*, and LSI does somewhat better for *re1, wap,* and *new3*; however these differences are quite small. This can also be seen by comparing the last row of the table, which shows the average value of RI that is achieved over the five different lower dimensional spaces.

The results presented in Table 3 provide a global overview of the retrieval performance achieved by CI over an entire collection of documents. To see how well it does in bringing closer together documents of the different classes, we computed the RI measure on a per class basis. These results are shown in Table 4 for both CI and LSI. Due to space considerations, we only present the per-class comparisons for a single number of dimensions. In particular, for *new3*, Table 4 shows the per-class RI measures obtained by reducing the number of dimensions to 125, and for the other data

[2]We also computed the RI measures using 10-, 30-, and 40-nearest-neighbors. The relative performance between CI and LSI remained the same, so we did not include these results.

	re0		re1		la1		la2		fbis		wap		ohscal	
Ndims	CI	LSI	CI	LSI	CI	LSI	CI	LSI	CI	LSI	CI	LSI	CI	LSI
10	1.03	1.06	0.92	0.97	1.14	1.13	1.13	1.12	1.00	1.00	1.00	1.03	1.24	1.21
20	1.06	1.08	1.02	1.03	1.15	1.14	1.13	1.13	1.04	1.05	1.09	1.11	1.30	1.32
30	1.08	1.07	1.04	1.06	1.15	1.12	1.12	1.13	1.08	1.06	1.10	1.11	1.32	1.32
40	1.09	1.06	1.07	1.06	1.15	1.12	1.13	1.12	1.09	1.05	1.10	1.12	1.33	1.30
50	1.09	1.06	1.07	1.08	1.14	1.12	1.13	1.11	1.09	1.05	1.09	1.11	1.33	1.29
Average	1.07	1.066	1.024	1.04	1.146	1.126	1.128	1.122	1.06	1.042	1.076	1.096	1.304	1.288

	new3	
Ndims	CI	LSI
25	0.98	1.03
50	1.06	1.08
75	1.07	1.09
100	1.09	1.09
125	1.09	1.09
Average	1.058	1.076

Table 3: The values of the RI measure achieved by CI and LSI.

sets, Table 4 shows the per-class RI measures obtained by reducing the number of dimensions to 50. Also note that for each dataset, the column labeled "Size" shows the number of documents in each class. The various classes are sorted in decreasing class-size order.

A number of interesting observations can be made from the results shown in this table. First, the overall performance of CI is quite similar to LSI. Both schemes are able to improve the retrieval performance for some classes, and somewhat decrease it for others. Second, the size of the different classes does affect the retrieval performance. Both schemes tend to improve the retrieval of larger classes at a higher degree than they do for the smaller classes. Third, from these results we can see that CI compared to LSI, in general, does somewhat better for larger classes and somewhat worse for smaller classes. We believe this is a direct result of the way the clustering algorithm used by CI is biased towards creating large clusters (Section 5). A clustering solution that better balances the tradeoffs between the size and the variance of the clusters can potentially lead to better results even for the smaller classes. This is an area that we are currently investigating.

Summarizing the results, we can see that the dimensionality reductions computed by CI achieve comparable retrieval performance to that obtained using LSI. However, the amount of time required by CI to find the axes of the reduced dimensionality space is significantly smaller than that required by LSI. CI finds these axes by just using a fast clustering algorithm, whereas LSI needs to compute the singular-value-decomposition. The run-time comparison of CI and LSI is shown in Table 5. We used the single-vector Lanczos method ($las2$) of SVDPACK [4] for LSI. SVDPACK is a widely used package for computing the singular-value-decomposition of sparse matrices and $las2$ is the fastest implementation of SVD among the algorithms available in SVDPACK. From the results shown in this table we can see that CI is consistently eight to ten times faster than LSI.

6.3 Supervised Dimensionality Reduction

One of the main features of CI is that it can quickly compute the axes of the reduced dimensional space by taking into account a priori knowledge about the classes that the various documents belong to. As discussed in Section 4, this supervised dimensionality reduction is particularly useful to improve the retrieval performance of a pre-categorized collection of documents. To illustrate this, we used the same set of data sets as in the previous section, but this time we used the centroid of the various classes as the axes of the reduced dimensionality space. The RI measures for the different classes in each one of these data sets are shown in Table 6. Note that the number of dimension in the reduced

re0		
Size	CI	LSI
608	1.06	1.01
319	1.15	1.11
219	1.19	1.12
80	1.53	1.30
60	1.04	0.99
42	0.97	1.14
39	0.98	1.14
38	1.06	0.82
37	0.89	1.16
20	0.95	1.06
16	0.75	1.00
15	0.86	0.76
11	0.68	0.73

re1		
Size	CI	LSI
371	1.08	1.05
330	1.11	1.06
137	1.21	1.24
106	1.19	1.13
99	1.06	1.04
87	1.07	1.04
60	1.15	1.14
50	0.79	0.90
48	0.94	0.99
42	0.82	1.01
37	0.92	1.22
32	1.04	1.19
31	1.13	1.23
31	1.12	1.26
27	1.15	1.30
20	0.99	1.06
20	1.24	1.27
19	0.93	0.93
19	0.61	0.80
18	0.61	0.97
18	0.73	1.09
17	0.69	0.83
15	1.08	0.98
13	0.82	0.80
10	0.50	0.43

fbis		
Size	CI	LSI
506	1.05	1.03
387	1.00	0.99
358	1.17	1.14
190	1.03	0.99
139	1.02	1.04
125	1.22	1.15
121	1.03	1.09
119	0.97	0.99
94	1.28	1.20
92	1.27	1.09
65	0.93	1.04
48	1.39	1.29
46	0.97	1.14
46	1.08	1.06
46	0.99	0.97
43	0.87	0.91
38	1.17	0.94

wap		
Size	CI	LSI
341	1.06	1.04
196	1.31	1.32
168	0.97	0.94
130	0.99	1.03
97	1.13	1.09
91	1.16	1.29
91	1.51	1.74
76	1.08	1.14
65	1.02	0.99
54	1.01	1.09
44	1.55	1.34
40	0.84	0.88
37	1.43	1.27
35	1.69	1.52
33	1.03	1.10
18	0.49	0.52
15	0.75	0.76
13	0.53	0.87
11	1.07	1.02
5	0.78	0.78

new3		
Size	CI	LSI
696	1.10	1.05
568	1.01	0.98
493	1.35	1.24
369	1.10	1.11
330	1.02	1.03
328	1.05	1.08
326	1.11	1.09
306	1.05	1.05
281	1.09	1.05
278	1.06	1.06
276	1.06	1.03
270	1.17	1.14
253	1.25	1.29
243	1.05	1.04
238	1.05	1.08
218	1.07	1.11
211	1.02	1.02
198	1.26	1.38
196	1.15	1.14
187	1.11	1.16
181	1.22	1.23
179	1.07	1.02
174	0.94	0.99
171	1.44	1.35
171	0.95	1.00
161	1.09	1.11
159	1.22	1.19
153	1.06	1.02
141	1.13	1.16
139	1.06	1.10
139	1.12	1.11
136	1.01	1.08
130	1.23	1.22
126	1.17	1.08
124	1.03	1.03
123	1.00	1.16
120	0.89	0.97
116	0.81	0.92
115	0.94	1.03
110	1.13	1.08
110	1.02	1.07
106	1.00	1.02
105	1.12	1.16
104	1.36	1.17

la1		
Size	CI	LSI
943	1.16	1.12
738	1.09	1.07
555	1.16	1.11
354	1.26	1.25
341	1.14	1.14
273	1.08	1.08

la2		
Size	CI	LSI
905	1.17	1.13
759	1.07	1.06
487	1.16	1.13
375	1.14	1.15
301	1.09	1.14
248	1.00	1.09

ohscal		
Size	CI	LSI
1621	1.28	1.24
1450	1.37	1.37
1297	1.21	1.19
1260	1.28	1.29
1159	1.41	1.41
1037	1.34	1.39
1001	1.57	1.53
864	1.34	1.33
764	1.42	1.35
709	1.16	1.28

Table 4: The per-class RI measures for various data sets.

space for each data set is different, and is equal to the number of classes in the data set.

As we can see from this table, the supervised dimensionality reduction computed by CI dramatically improves the retrieval performance for all the different classes in each data set. Moreover, the retrieval performance of the smaller classes tends to improve the most. This is because in unsupervised dimensionality reduction, these smaller classes are not sufficiently represented (as the experiments shown in Table 4 indicate), whereas in supervised dimensionality reduction, all classes are equally represented, regardless of their size.

The supervised dimensionality reduction performed by CI can also be used to improve the performance of traditional classification algorithms. To illustrate this, we performed an experiment in which we used two traditional classification algorithms, C4.5 and *k*-nearest-neighbor, both on the original space, as well as on the reduced dimensional space. C4.5 [60] is a widely used decision tree-based classification algorithm that has been shown to produce good classification results, primarily on low dimensional data sets. The *k*-nearest-neighbor (*k*NN) classification algorithm is a well known instance-based classification algorithm that has been applied to text categorization since the early days of research [53, 29, 68].

	re0	*re1*	*la1*	*la2*	*fbis*	*wap*	*ohscal*	*new3*
CI	0.56	0.72	5.01	4.59	3.17	1.97	7.01	29.85
LSI	6.58	7.00	44.20	39.80	20.10	18.10	65.10	275.00

Table 5: Run-time comparison (in seconds) of LSI and CI. These times correspond to the amount of time required to compute 50 dimensions for all data sets except *new3* for which 125 dimensions were computed. All experiments were performed on a Linux workstation equipped with an Intel Pentium II running at 500Mhz.

For each set of documents, the reduced dimensionality experiments were performed as follows. First, the entire set of documents was split into a training and test set. Next, the training set was used to find the axes of the reduced dimensional space by constructing an axis for each one of the classes[3]. Then, both the training and the test set were projected into this reduced dimensional space. Finally, in the case of C4.5, the projected training and test set were used to learn the decision tree and evaluate its accuracy, whereas in the case of kNN, the neighborhood computations were performed on the projected training and test. In our experiments, we used a value of $k = 10$ for kNN, both for the original as well as for the reduced dimensional space.

The classification accuracy of the various experiments are shown in Table 7. These results correspond to the average classification accuracies of 10 experiments, where in each experiment a randomly selected 80% fraction of the documents was used for training and the remaining 20% was used for testing. The first two columns of this table, show the classification accuracy obtained by C4.5 and kNN when used on the original data sets. The next two columns show the classification accuracy results obtained by the same algorithms when used on the reduced dimensional space computed by CI. The next four columns show the classification accuracy obtained by these algorithms when used on the reduce dimensional space computed by LSI. For each algorithm, we present two sets of results, obtained on a 25- and on a 50-dimensional space. Note that these lower dimensional spaces were computed without taking into account any class information, as LSI cannot perform dimensionality reduction in a supervised setting. Finally, the last column shows the results obtained by the naive Bayesian (NB) classification algorithm in the original space. In our experiments, we used the NB implementation provided by the Rainbow [55] software library. The NB results are presented here to provide a reference point for the classification accuracies. Note that we did not use the NB algorithm in the reduced dimensional space, as NB cannot effectively handle continuous attributes [34]. Also, for each of these data sets, we highlighted the scheme that achieved the highest classification accuracy, by using a boldface font.

Looking at the results, we can see that both C4.5 and kNN, benefit greatly by the supervised dimensionality reduction computed by CI. For both schemes, the classification accuracy achieved in the reduced dimensional space is greater than the corresponding accuracy in the original space for all 21 data sets. In particular, over the entire 21 data sets, CI improves the average accuracy of C4.5 and kNN by 7%, and 6%, respectively. Comparing these results against those obtained by naive Bayesian, we can see that kNN, when applied on the reduced dimensional space, substantially outperforms naive Bayesian, which was not the case when comparing the performance of kNN in the original space. In particular, over the entire 21 data sets, the accuracy of kNN in the reduced space is 5% greater than that of naive Bayesian. Looking at the various classification results obtained by C4.5 and kNN on the lower dimensional spaces computed by LSI, we can see that the performance is mixed. In particular, comparing the best performance achieved in either one of the lower dimensional spaces, over that achieved in the original space, we can see that LSI improves the results obtained by C4.5 in only four data sets, and by kNN in ten data sets. However, CI, by computing a lower

[3]We also performed experiments in which the number of dimensions in the reduced space was two and three times greater than the number of classes. The overall performance of the algorithms did not change, and due to space limitations we did not include these results.

re0

Size	CI-S
608	1.12
319	1.31
219	1.28
80	1.89
60	1.26
42	2.17
39	1.30
38	1.38
37	1.66
20	1.54
16	1.60
15	1.32
11	1.64

re1

Size	CI-S
371	1.25
330	1.18
137	1.51
106	1.23
99	1.11
87	1.11
60	1.44
50	1.94
48	1.05
42	2.13
37	1.59
32	1.33
31	1.67
31	1.72
27	1.84
20	2.01
20	1.41
19	1.81
19	2.18
18	1.69
18	3.67
17	1.49
15	3.75
13	1.40
10	2.27

wap

Size	CI-S
341	1.05
196	1.72
168	1.31
130	1.42
97	1.17
91	1.75
91	1.94
76	1.37
65	1.22
54	1.71
44	3.81
40	1.14
37	2.36
35	2.98
33	2.83
18	3.63
15	3.49
13	2.57
11	2.66
5	2.78

fbis

Size	CI-S
506	1.07
387	1.02
358	1.31
190	1.07
139	1.17
125	1.32
121	1.17
119	1.03
94	1.33
92	1.44
65	1.40
48	1.80
46	1.80
46	1.09
46	1.73
43	2.26
38	2.68

new3

Size	CI-S
696	1.13
568	1.03
493	1.87
369	1.31
330	1.09
328	1.49
326	1.24
306	1.08
281	1.18
278	1.16
276	1.07
270	1.23
253	1.63
243	1.07
238	1.35
218	1.24
211	1.17
198	1.85
196	1.20
187	1.34
181	1.39
179	1.14
174	1.84
171	1.92
171	1.09
161	1.19
159	1.41
153	1.25
141	1.69
139	1.25
139	1.27
136	1.19
130	1.29
126	1.66
124	1.06
123	1.23
120	1.03
116	1.53
115	1.18
110	1.18
110	1.11
106	1.04
105	1.28
104	2.54

la1

Size	CI-S
943	1.33
738	1.11
555	1.21
354	1.34
341	1.41
273	2.22

la2

Size	CI-S
905	1.31
759	1.10
487	1.25
375	1.20
301	1.48
248	1.75

ohscal

Size	CI-S
1621	1.38
1450	1.56
1297	1.37
1260	1.46
1159	1.63
1037	1.81
1001	1.85
864	1.47
764	1.78
709	1.51

Table 6: The per-class RI measures for various data sets for supervised dimensionality reduction.

dimensional space in a supervised setting, significantly and consistently outperforms the classification results obtained on the lower dimensional spaces obtained by LSI.

We have not included the results of C4.5 and *k*NN using feature selection techniques due to the inconsistent performance of such schemes in these data sets. In particular, the right number of dimensions for different data sets varies considerably. For detailed experiments showing the characteristics of feature selection schemes in text categorization, readers are advised to see [70, 25].

7 Conclusion and Directions of Future Work

In this paper we presented a new fast dimensionality reduction technique called concept indexing that can be used equally well for reducing the dimensions in a supervised and in an unsupervised setting. CI reduces the dimensionality of a document collection according to the concepts present in the collection and expresses each document as a function of the various concepts. Our analysis has shown that the lower-dimensional representation computed by CI is capable of capturing both the actual as well as the latent information available in the document collections. In particular,

					LSI Reduced Space				
	Original Space		CI Reduced Space		C4.5		*k*NN		
	C4.5	*k*NN	C4.5	*k*NN	25 Dims	50 Dims	25 Dims	50 Dims	NB
west1	85.5%	82.9%	86.2%	**86.7%**	73.7%	74.5%	83.0%	81.4%	**86.7%**
west2	75.3%	77.2%	75.3%	**78.7%**	63.8%	59.2%	75.5%	73.8%	76.5%
west3	73.5%	76.1%	74.5%	**80.6%**	57.8%	55.3%	75.5%	77.3%	75.1%
oh0	82.8%	84.4%	87.3%	**89.8%**	74.5%	72.8%	83.9%	81.9%	89.1%
oh5	79.6%	85.6%	88.4%	**92.0%**	76.5%	76.7%	87.0%	86.8%	87.1%
oh10	73.1%	77.5%	79.6%	**82.6%**	70.9%	65.5%	79.4%	77.7%	81.2%
oh15	75.2%	81.7%	84.6%	**86.4%**	67.5%	64.9%	81.3%	80.7%	84.0%
re0	75.8%	77.9%	82.3%	**85.0%**	69.1%	64.4%	79.5%	76.3%	81.1%
re1	77.9%	78.9%	80.0%	**81.6%**	59.8%	60.6%	71.2%	75.4%	80.5%
tr11	78.2%	85.3%	87.0%	**88.9%**	79.3%	80.5%	81.3%	83.0%	85.3%
tr12	79.2%	85.7%	88.4%	**89.0%**	76.2%	72.5%	80.8%	82.7%	79.8%
tr21	81.3%	89.1%	**90.3%**	90.0%	74.6%	73.1%	87.6%	88.5%	59.6%
tr31	93.3%	93.9%	94.7%	**96.9%**	90.2%	87.5%	93.0%	92.3%	94.1%
tr41	89.6%	93.5%	95.3%	**95.9%**	89.9%	87.3%	93.4%	92.4%	94.5%
tr45	91.3%	91.1%	92.9%	**93.6%**	80.3%	80.9%	91.1%	92.1%	84.7%
la1	75.2%	82.7%	85.7%	**87.6%**	76.1%	74.2%	83.4%	82.1%	**87.6%**
la2	77.3%	84.1%	87.2%	88.6%	78.2%	76.1%	85.9%	84.7%	**89.9%**
fbis	73.6%	78.0%	81.3%	**84.1%**	59.7%	56.0%	76.4%	76.3%	77.9%
wap	68.1%	75.1%	77.5%	**82.9%**	62.3%	60.2%	74.3%	76.1%	80.6%
ohscal	71.5%	62.5%	73.5%	**77.8%**	59.4%	57.5%	70.9%	69.6%	74.6%
new3	72.7%	67.9%	73.1%	**77.2%**	41.1%	43.5%	53.9%	63.1%	74.4%

Table 7: The classification accuracy of the original and reduced dimensional data sets.

CI captures concepts with respect to word synonymy and polysemy. Our experimental evaluation has shown that in an unsupervised setting, CI performs equally well to LSI while requiring an order of magnitude less time, and in a supervised setting it dramatically improves the performance of various classification algorithms.

The performance of CI can be improved in a variety of ways. First, CI when used in an unsupervised setting, can take advantage of better document clustering algorithms, leading to better lower dimensional spaces as well as faster performance. One area that we are currently investigating is to develop robust clustering algorithms that compute a k-way clustering directly and not via recursive bisection. Such techniques hold the promise of improving both the quality of the lower dimensional representation, especially for small classes, as well as further reducing the already low computational requirements of CI. Second, the supervised dimensionality reductions computed by CI can be further improved by using techniques that adjust the importance of the different features in a supervised setting. A variety of such techniques have been developed in the context of k-nearest-neighbor classification [13, 65, 64, 37, 40, 52, 25], all of which can be used to scale the various dimensions prior to the dimensionality reduction for computing centroid vectors and to scale the reduced dimensions for the final classification.

References

[1] Charu C. Aggarwal, Stephen C. Gates, and Philip S. Yu. On the merits of building categorization systems by supervised clustering. In *Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 352–356, 1999.

[2] H. Almuallim and T.G. Dietterich. Learning with many irrelevant features. In *Proc. of the Ninth International Conference on Machine Learning*, pages 547–552, 1991.

[3] L. Baker and A. McCallum. Distributional clustering of words for text classification. In *SIGIR-98*, 1998.

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

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

[6] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the world wide web using WebACE. *AI Review (accepted for publication)*, 1999.

[7] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27(3):329–341, 1999.

[8] P.S. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining*, 1998.

[9] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons, 1991.

[10] T. Curran and P. Thompson. Automatic categorization of statute documents. In *Proc. of the 8th ASIS SIG/CR Classification Research Workshop*, Tucson, Arizona, 1997.

[11] D.R. Cutting, J.O. Pedersen, D.R. Karger, and J.W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the ACM SIGIR*, pages 318–329, Copenhagen, 1992.

[12] D.J. Spiegelhalter D. Michie and C.C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[13] W. Daelemans, S. Gills, and G. Durieux. Learnability and markedness in data-driven acquisition of stress. Technical Report TR 43, Institute for Language Technology and Artificial Intelligence, Tilburg University, Netherlands, 1993.

[14] B.V. Dasarathy. *Nearest neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press, 1991.

[15] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Hashman. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, 41(6), 1990.

[16] C.H.Q. Ding. A similarity-based probability model for latent semantic indexing. In *SIGIR-99*, 1999.

[17] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[18] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[19] S.T. Dumais. Using LSI for information filtering: TREC-3 experiments. In *Proc. of the Third Text REtrieval Conference (TREC-3), National Institute of Standards and Technology*, 1995.

[20] U. Fayyad, C. Reina, and P.S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining*, 1998.

[21] V. Ganti, R. Ramakrishnan, J. Gehrke, A. Powell, and J. French. Clustering large datasets in arbitrary metric spaces. In *Proc. of the 15th Int'l Conf. on Data Eng.*, 1999.

[22] D. E. Goldberg. *Genetic Algorithms in Search, Optimizations and Machine Learning*. Morgan-Kaufman, 1989.

[23] E.H. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. WebACE: A web agent for document categorization and exploartion. In *Proc. of the 2nd International Conference on Autonomous Agents*, May 1998.

[24] E.H. Han and G. Karypis. Centroid-based document classification algorithms: Analysis & experimental results. Technical Report TR-00-XXX, Department of Computer Science, University of Minnesota, Minneapolis, 2000. Also available on WWW at URL http://www.cs.umn.edu/˜karypis.

[25] Eui-Hong Han. *Text Categorization Using Weight Adjusted* k*-Nearest Neighbor Classification*. PhD thesis, University of Minnesota, October 1999.

[26] W. Hersh, C. Buckley, T.J. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *SIGIR-94*, pages 192–201, 1994.

[27] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR-99*, 1999.

[28] S.J. Hong. Use of contextual information for feature ranking and discretization. *IEEE Transactions on Knowledge and Data Eng.*, 9(5):718–730, September/October 1997.

[29] Makato Iwayama and Takenobu Tokunaga. Cluster-based text categorization: a comparison of category search strategies. In *SIGIR-95*, pages 273–281, 1995.

[30] J. E. Jackson. *A User's Guide To Principal Components*. John Wiley & Sons, 1991.

[31] A.K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[32] T. Joachims. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *Proc. of the Fourteenth International Conference on Machine Learning*, 1997.

[33] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of the European Conference on Machine Learning*, 1998.

[34] G.H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proc. of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 338–345, 1995.

[35] K.S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 29(4):11–21, 1973.

[36] L. N. Kanal and Vipin Kumar, editors. *Search in Artificial Intelligence*. Springer-Verlag, New York, NY, 1988.

[37] K. Kira and L.A. Rendell. A practical approach to feature selection. In *Proc. of the 10th International Conference on Machine Learning*, 1992.

[38] R. Kohavi and D. Sommerfield. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *Proc. of the First Int'l Conference on Knowledge Discovery and Data Mining*, pages 192–197, Montreal, Quebec, 1995.

[39] T. Kohonen. *Self-Organization and Associated Memory*. Springer-Verlag, 1988.

[40] I. Kononenko. Estimating attributes: Analysis and extensions of relief. In *Proc. of the 1994 European Conference on Machine Learning*, 1994.

[41] R. E. Korf. Search. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 994–998. John Wiley & Sons, Inc., 1990.

[42] Wai Lam and Chao Yang Ho. Using a generalized instance set for automatic text categorization. In *SIGIR-98*, 1998.

[43] T. K. Landauer and S.T. Dumais. A solution to plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

[44] P. Langley and S. Sage. Induction of selective bayesian classifiers. In *Proc. of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406, Seattle, WA, 1994.

[45] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proc. of the Fifth ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1999.

[46] D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Tenth European Conference on Machine Learning*, 1998.

[47] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *SIGIR-94*, 1994.

[48] D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In *Proc. of the Third Annual Symposium on Document Analysis and Information Retrieval*, 1994.

[49] D. D. Lewis. Reuters-21578 text categorization test collection distribution 1.0. *http://www.research.att.com/$\sim$lewis*, 1999.

[50] David D. Lewis, Robert E. Shapire, James P. Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19 th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–306, 1996.

[51] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, 1998.

[52] D.G. Lowe. Similarity metric learning for a variable-kernel classifier. *Neural Computation*, pages 72–85, January 1995.

[53] B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory based reasoning. In *SIGIR-92*, pages 59–64, 1992.

[54] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[55] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[56] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher. Web page categorization and feature selection using association rule and principal component clustering. In *7th Workshop on Information Technologies and Systems*, Dec. 1997.

[57] C. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent sematic indexing: A probabilistic analysis. In *Proc. of Symposium on Principles of Database Systems*, 1998.

[58] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of English words. In *Proc. of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, 1993.

[59] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[60] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[61] Forrester Research. Coping with complex data. The Forrester Report, April 1995.

[62] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[63] TREC. Text REtrieval conference. *http://trec.nist.gov*.

[64] D. Wettschereck, D.W. Aha, and T. Mohri. A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. *AI Review*, 11, 1997.

[65] D. Wettschereck and T.G. Dietterich. An experimental comparison of the nearest neighbor and nearest hyperrectangle algorithms. *Machine Learning*, 19:5–28, 1995.

[66] Marilyn Wulfekuhler and Bill Punch. Finding salient features for personal web page categories. In *6th WWW Conference*, Santa Clara, CA, 1997.

[67] Yahoo! Yahoo! http://www.yahoo.com.

[68] Y. Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *SIGIR-94*, 1994.

[69] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR-99*, 1999.

[70] Y. Yang and J. Pederson. A comparative study on feature selection in text categorization. In *Proc. of the Fourteenth International Conference on Machine Learning*, 1997.

[71] T. Zhang, R. Ramakrishnan, and M. Linvy. Birch: an efficient data clustering method for large databases. In *Proc. of 1996 ACM-SIGMOD Int. Conf. on Management of Data*, Montreal, Quebec, 1996.