# Modified LSI Model for Efficient Search by Metric Access Methods

Tomáš Skopal[1] and Pavel Moravec[2]

[1]Charles University in Prague, FMP, Department of Software Engineering,
Malostranské nám. 25, 118 00 Prague, Czech Republic
tomas@skopal.net
[2]Technical University of Ostrava, FEECS, Department of Computer Science,
17. listopadu 15, 708 33 Ostrava, Czech Republic
pavel.moravec@vsb.cz

**Abstract.** Text collections represented in LSI model are hard to search efficiently (i.e. quickly), since there exists no indexing method for the LSI matrices. The inverted file, often used in both boolean and classic vector model, cannot be effectively utilized, because query vectors in LSI model are dense. A possible way for efficient search in LSI matrices could be the usage of metric access methods (MAMs). Instead of cosine measure, the MAMs can utilize the deviation metric for query processing as an equivalent dissimilarity measure. However, the intrinsic dimensionality of collections represented by LSI matrices is often large, which decreases MAMs' performance in searching. In this paper we introduce $\sigma$-LSI, a modification of LSI in which we artificially decrease the intrinsic dimensionality of LSI matrices. This is achieved by an adjustment of singular values produced by SVD. We show that suitable adjustments could dramatically improve the efficiency when searching by MAMs, while the precision/recall values remain preserved or get only slightly worse.

## 1 Introduction

Text collections represented in the classic vector model (CVM) can be efficiently (i.e. quickly) searched using the inverted file. More precisely, the inverted file provides a way for very efficient processing of queries, the vectors of which are sparse (such a query contains only several terms). However, in case of LSI model the query vectors are dense, and the usage of inverted file becomes useless, since processing of any query deteriorates to sequential search over the entire concept-by-document matrix.

In this paper we utilize a method of searching in LSI collections by metric access methods (MAMs). The metric access methods are, however, sensitive to the curse of dimensionality, i.e. they become inefficient for high dimensionalities. Therefore, in this paper we propose $\sigma$-LSI, a modified LSI model in which we artificially reduce the intrinsic dimensionality of the indexed collection. This is achieved by an adjustment of singular values produced by SVD. We show that suitable adjustments could dramatically improve the efficiency when searching

D.E. Losada and J.M. Fernández-Luna (Eds.): ECIR 2005, LNCS 3408, pp. 245–259, 2005.

by MAMs, while the precision/recall values remain preserved or get only slightly worse.

The paper is organized as follows: In the rest of this section we briefly overview CVM, the LSI model, and formulate the problem of searching in LSI model. In Section 3 we show how the classic similarity search in CVM (LSI model respectively) can be turned into metric search. We also mention the principles of metric access methods and the problem of high intrinsic dimensionality. In Section 4 we propose $\sigma$-LSI model allowing a more efficient search by MAMs. The *effectiveness* (the quality) and *efficiency* (the response time) of retrieval in the $\sigma$-LSI model are evaluated in Section 5.

### 1.1 Classic Vector Model

In CVM, a given text collection (containing $n$ documents consisting of $m$ unique terms) is represented by an $m \times n$ *term-by-document matrix* $A$, where each column vector $d_j$ in $A$ represents a single document $D_j$. Thus, the documents are represented as points in $m$-dimensional vector space (the *document-space*). Each dimension of the document-space is associated with a single term, while each coordinate in a *document vector* $d_j$ represents a weight of the respective term in the document. There are many ways how to compute the term weights $A_{ij}$ – a popular weight construction is computed as $tf \cdot idf$ (see e.g. [3]).

The most important part of CVM is the query semantics for searching the matrix $A$ with respect to a query $Q$, and returning only the relevant document vectors (appropriate documents respectively). The query $Q$ is represented by a vector $q$ in the document space the same way as a document $D_j$ is represented by $d_j$. The goal is to return the most similar documents to the query. For this purpose a similarity measure must be defined, assessing a similarity score for each pair of query and document vectors $(q, d_j)$. In many cases, the *cosine measure*

$$\mathrm{SIM}_{cos}(q, d_j) = \frac{\sum_{i=1}^{m} q_i d_{ji}}{\sqrt{\sum_{i=1}^{m} {q_i}^2 \cdot \sum_{i=1}^{m} {d_{ji}}^2}}$$

is widely used. Besides the simple ranking to $q$ (used for *ranked lists*), we also distinguish bounded queries, in particular *range queries* and *k-nearest neighbors (k-NN) queries*. A range query returns documents with similarity to the query

| term \ doc. | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ |
|---|---|---|---|---|---|
| *database* | 0 | 0.48 | 0.05 | 0 | 0.70 |
| *vector* | 0.23 | 0 | 0.23 | 0 | 0 |
| *index* | 0.43 | 0 | 0 | 0 | 0 |
| *image* | 0 | 0 | 0.10 | 0 | 0.54 |
| *compression* | 0 | 0 | 0 | 0 | 0.21 |
| *multimedia* | 0.12 | 0.52 | 0.62 | 0 | 0 |

**Fig. 1.** Term-by-document matrix $A$

higher than a given similarity threshold $t$. A $k$-NN query returns the $k$ most similar documents[1].

## 2 Latent Semantic Indexing

*Latent semantic indexing* (*LSI*) [3, 4] is an algebraic extension of CVM. Its benefits rely on discovering *latent semantics* hidden in the term-by-document matrix $A$. Informally, LSI discovers significant groups of terms (called *concepts*) and represents the documents as linear combinations of the concepts. Moreover, the concepts are ordered according to their significance in the collection, which allows us to consider only the first $k$ concepts important (the remaining ones are interpreted as "noise" and discarded). To name the advantages, LSI helps solve problems with synonymy and homonymy. Furthermore, LSI is often referred to as more successful in recall when compared to CVM [4], which was proved for pure (only one topic per document) and style-free collections [17].

Formally, we decompose the term-by-document matrix $A$ by *singular value decomposition* (*SVD*), calculating singular values and singular vectors of $A$. SVD is especially suitable in its variant for sparse matrices (Lanczos [13]). Several approximate methods for faster SVD calculation were offered recently, such as using random projection of document vectors into suitable subspace before LSI calculation [17] or application of Monte-Carlo method [11].

There are several other methods for latent semantic indexing, such as ULV-decomposition [5], random indexing [16] (and some other approaches achieving similar goals, e.g. language modeling [19]), which we do not discuss in this paper.

**Theorem 1 (Singular Value Decomposition [4]).** *Let $A$ is an $m \times n$ rank-$r$ matrix. Be values $\sigma_1, \ldots, \sigma_r$ calculated from eigenvalues of matrix $AA^T$ as $\sigma_i = \sqrt{\lambda_i}$. Then there exist column-orthonormal matrices $U = (u_1, \ldots, u_r)$ and $V = (v_1, \ldots, v_r)$, where $U^T U = I_m$ a $V^T V = I_n$, and a diagonal matrix $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$, where $\sigma_i > 0, \sigma_i \geq \sigma_{i+1}$. The decomposition*

$$A = U \Sigma V^T$$

*is called* singular decomposition *of matrix $A$ and the numbers $\sigma_1, \ldots, \sigma_r$ are* singular values *of the matrix $A$. Columns of $U$ (or $V$) are called* left (or right) singular vectors *of matrix $A$.*

Now we have a decomposition of the original term-by-document matrix $A$. The left and right singular vectors (i.e. $U$ and $V$ matrices) are not sparse. We get $r$ nonzero singular numbers, where $r$ is the rank of the original matrix $A$. Because the singular values usually fall quickly, we can take only $k$ greatest singular values with the corresponding singular vector coordinates and create a *$k$-reduced singular decomposition* of $A$.

[1] In the next section we independently use $k$ for another parameter (rank-$k$ SVD), but in either case the respective meaning of $k$ is obvious from the actual context.

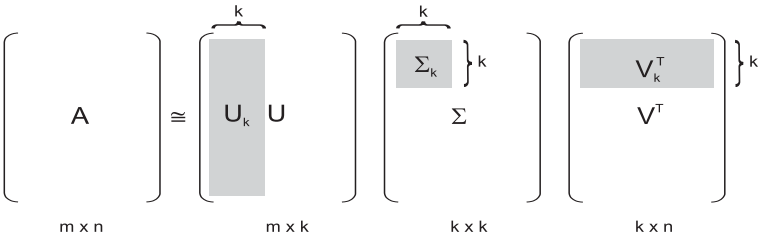


**Fig. 2.** $k$-reduced singular value decomposition

**Definition 1.** *Let us have* $k$ $(0 < k < r)$ *and singular value decomposition of* $A$

$$A = U\Sigma V^T \approx A_k = (U_k U_0) \begin{pmatrix} \Sigma_k & 0 \\ 0 & \Sigma_0 \end{pmatrix} \begin{pmatrix} V_k^T \\ V_0^T \end{pmatrix}$$

*We call* $A_k = U_k \Sigma_k V_k^T$ *a k-reduced singular value decomposition* (*rank-k SVD*).

Instead of the $A_k$ matrix, a *concept-by-document matrix* $D_k = \Sigma_k V_k^T$ is used in LSI as the representation of document collection. The document vectors (columns in $D_k$) are now represented as points in $k$-dimensional space (the *pseudodocument-space*). For an illustration of rank-$k$ SVD see Figure 2.

The value of $k$ was experimentally determined as several tens or hundreds (e.g. 50–250), however, the optimal[2] value of $k$ is hard to choose; it is dependent on the number of topics in collection. Rank-$k$ SVD is the best rank-$k$ approximation of the original matrix $A$, regarding to Frobenius norm (see e.g. [12]). This means, that any other decomposition will increase the sum of squares of matrix $A - A_k$. However, this does not tell us that we could not obtain better precision and recall values with a different approximation.

To execute a query Q in the pseudodocument-space, we create a reduced query vector $q_k = U_k^T q$ (another approach is to simply use a matrix $D'_k = V_k^T$ instead of $D_k$, and $q'_k = \Sigma_k^{-1} U_k^T q$). Instead of $A$ against $q$, the matrix $D_k$ against $q_k$ (or $q'_k$) is evaluated using the cosine measure. The crucial property is that, due to the projection by dense matrix $U_k^T$, $q_k$ is dense as well (even if $q$ is sparse).

### 2.1 LSI Model and Inverted Files

In CVM, searching the term-by-document matrix $A$ according to a query $Q$ can be provided using *inverted file* [15, 18, 1], which can be viewed as the matrix $A$ stored by rows. For a given matrix $A$ the inverted file consists of $m$ lists, each list is associated with a single term. Each list stores entries, which are pairs consisting of a document id and weight of the term in corresponding document (obviously, entries with zero weights are not stored). When a query is processed, only the lists representing terms from the query are sequentially searched.

[2] optimal in sense of best achieved precision/recall values.

The inverted file is very efficient for processing of sparse query vectors (few-term queries respectively), because only several lists have to be processed. Unfortunately, in case of LSI the pseudo-query vector is dense and usage of inverted file for indexing $D_k$ would deteriorate to sequential search over the entire file and thus, over the entire matrix $D_k$.

## 3 Metric Indexing

Recently, there has been introduced an approach to searching in LSI model, based on *metric indexing* [20]. Instead of inverted file, the *M-tree* [9] was used for indexing the matrix $D_k$. Before we discuss benefits of the metric approach, we must turn the cosine measure (similarity) into metric (distance).

### 3.1 Turning Vector Model into Metric Model

The cosine measure $\mathrm{SIM}_{cos}(d_i, d_j)$ itself is not a metric, since it does not satisfy three metric properties (reflexivity, positivity and triangular inequality). Even $1-\mathrm{SIM}_{cos}(d_i, d_j)$ is not a metric, since it does not satisfy the triangular inequality. As an appropriate metric, we use the *deviation metric* (or angular distance) $d_{dev}(d_i, d_j)$, defined as

$$d_{dev}(d_i, d_j) = arccos(\mathrm{SIM}_{cos}(d_i, d_j))$$

Instead of cosine, the deviation metric measures directly the angle between two vectors[3]. Since *arccos* is strictly decreasing on $\langle -1, 1\rangle$, the deviation metric preserves the semantic meaning of cosine measure. There is only a difference in terminology – cosine measure is *similarity function* (similar documents have a high score), while the deviation metric is *dissimilarity function* (similar documents have a lower score, i.e. they are close). Hence, the $k$-dimensional pseudodocument-space $\mathbb{R}^k$ together with the deviation metric $d_{dev}$ can be regarded as a metric space $\mathcal{M} = (\mathbb{R}^k, d_{dev})$.

The queries in metric model are evaluated in similar way as in CVM; the difference is that range queries select objects within a *query radius* $r_Q$ (which equals to *arccos* of the desired similarity threshold $t$), while $k$-NN queries select the $k$ closest objects.

### 3.2 Metric Access Methods

The *metric access methods* [8] organize (or index) a given metric dataset $\mathbb{S} \subset \mathcal{M}$ in a way that metric queries (e.g. range or $k$-NN queries) can be processed efficiently – without a need of processing the entire dataset $\mathbb{S}$. The main principle behind all MAMs is the triangular inequality property satisfied by every metric. Due to the triangular inequality, MAMs can organize the objects in equivalence

[3] Actually, we can view the deviation metric $d_{dev}$ as a kind of Euclidean ($L_2$) distance, defined just on the surface of unitary hyper-sphere.

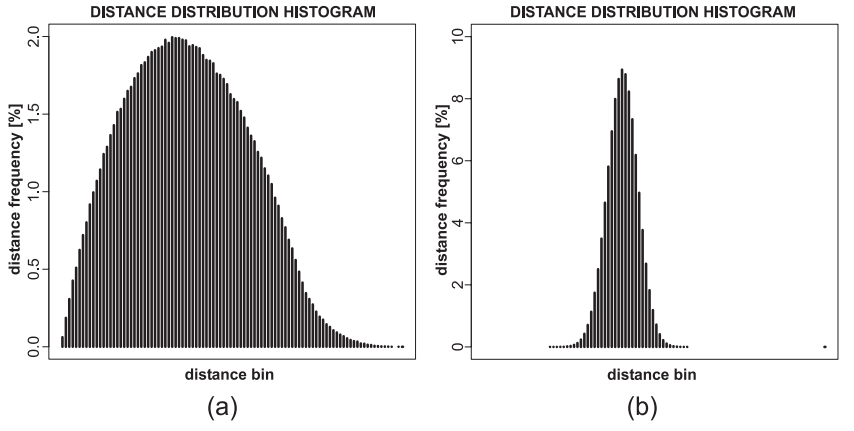


**Fig. 3.** DDHs indicating (a) low (b) high intrinsic dimensionality

classes (the classes are some regions in the metric space). When a query is processed, many irrelevant equivalence classes are filtered (those with metric regions not overlapping the query region), and so the searching becomes more efficient. Another advantage is that MAMs use solely the metric function for indexing, no information about the indexed objects representation is necessary. This feature allows to index/search non-vectorial datasets, too.

There has been developed a plenty of MAMs, varying in applicability to different problems. Besides others, we name *M-tree* [9], *vp-tree* [22], *LAESA* [14], *D-index* [10], etc.

### 3.3 Intrinsic Dimensionality

The metric indexing itself (as was presented in [20]) could be quite beneficial for searching in the LSI model. However, searching in a collection of high-dimensional document vectors is negatively affected by a phenomenon called the *curse of dimensionality* [6, 7]. For MAMs the curse of dimensionality causes almost all equivalence classes to be overlapped by nearly every "reasonable" query region, so that searching deteriorates to sequential scan over all the classes.

In the context of metric indexing, the curse of dimensionality can be generalized for general metric spaces. The major condition determining the efficiency limits of any metric access method is the *intrinsic dimensionality* of the indexed dataset, defined as (proposed in [7]):

$$\rho(\mathbb{S}, d) = \frac{\mu^2}{2\sigma^2}$$

where $\mu$ and $\sigma^2$ are the mean and the variance of the dataset's *distance distribution* (according to a metric $d$). In other words, the intrinsic dimensionality is low if there exist tight clusters of objects. Conversely, if all pairs of the indexed objects are almost equally distant, the intrinsic dimensionality is high

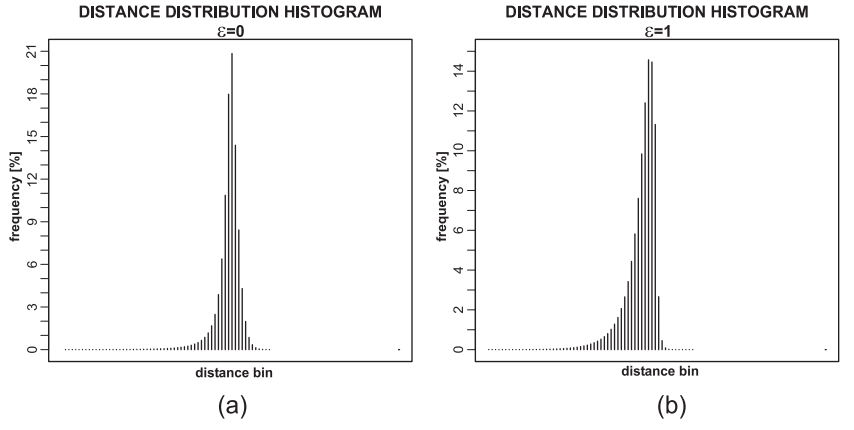


**Fig. 4.** (a) DDH for $D'_k$ (b) DDH for $D_k$

(i.e. the mean is high and/or the variance is low), which means the dataset is poorly intrinsically structured. In Figure 3 see an example of distance distribution histograms (DDHs) indicating lower ($\rho \approx 2$) and higher ($\rho \approx 30$) intrinsic dimensionalities.

In case of vector datasets, the intrinsic dimensionality can reach up to (or even beyond) the value of the classic (embedding) dimensionality. For example, for uniformly distributed $n$-dimensional vectors (i.e. not clustered) $\rho \approx n$.

So far, for datasets of high intrinsic dimensionality there still does not exist an efficient MAM for exact[4] metric search.

## 4 The $\sigma$-LSI Model

In case of LSI, we are concerned by intrinsic dimensionality of the pseudodocument vectors (columns in $D_k$), with respect to the deviation metric $d_{dev}$. The smaller $\rho$, the greater search efficiency can be achieved for the MAMs.

In this section we propose the $\sigma$-LSI model, a modification of LSI in which we are able to artificially decrease the intrinsic dimensionality of $D_k$.

### 4.1 Motivation

In order to understand the intrinsic dimensionality of $D_k$, we first consider the simpler approach of LSI, where the pseudodocument matrix is just $D'_k = V_k^T$ (instead of $D_k = \Sigma_k V_k^T$). This is equivalent to $D'_k = \Sigma_k^0 V_k^T$, where $\Sigma_k^0$ is unitary matrix (the singular values $\sigma_i$ are powered by 0). To illustrate the situation on an example, we use a term-by-document matrix $A$ (closely described in Section 5) decomposed using rank-$k$ SVD, $k = 100$.

[4] Nevertheless, efficient searching in high-dimensional datasets can be realized by approximate or probabilistic MAMs, but such methods often suffer from lower precision/recall values [23, 7].

In Figure 4a see the DDH for columns in $D'_k$ with respect to $d_{dev}$. The intrinsic dimensionality is $\rho = 98.1$, so we can claim that in this case $k \approx \rho$. This interesting observation arises from the fact that rows in $V_k^T$ are orthonormal and columns in $V_k^T$ (the pseudodocument vectors) are (almost) uniformly distributed.

Second, we consider the pseudodocument matrix $D_k = \Sigma_k V_k^T$ (the classic LSI). In Figure 4b see the DDH for columns in $D_k$ with respect to $d_{dev}$, the intrinsic dimensionality is now $\rho = 52.6$. Obviously, the difference between $\rho(D'_k, d_{dev})$ and $\rho(D_k, d_{dev})$ is in the multiplication of $V_k^T$ by $\Sigma_k$. Since the singular values $\sigma_i$ fall with increasing $i$, the uniformly distributed columns of $V_k^T$ (i.e. $D'_k$) turn into non-uniformly distributed columns of $\Sigma_k V_k^T$ (i.e. $D_k$). Furthermore, multiplication with greater $\sigma_i$ makes the $i$-th dimension ($i$-th concept resp.) more significant and vice versa. In consequence, only the most significant dimensions can affect the spatial distribution of pseudodocument vectors; the small values in insignificant dimensions can "shift" the vectors only fractionally. Hence, the quicker falling of $\sigma_i$, the smaller number of significant dimensions and, in turn, the smaller intrinsic dimensionality of $D_k$.

### 4.2 Singular Values Modification

To decrease the intrinsic dimensionality of $D_k$, we can adjust the singular values $\sigma_i$ such that they fall more quickly (with increasing $i$). This can be achieved by a suitable modifying function $f$.

$$\Sigma_k = diag(\sigma_1, \ldots, \sigma_k) \qquad \Longrightarrow \qquad \Sigma_k^f = diag(f(\sigma_1), \ldots, f(\sigma_k))$$

The function $f$ must be increasing in order to preserve the ordering of singular values (they are ordered by values). Moreover, $f$ must be convex, because we need to make the falling of $\sigma_i$ faster (concave functions do the opposite).

Finally, we apply the modified values in $\Sigma_k^f$ instead of the original $\Sigma_k$, i.e. we use $D_k^f = \Sigma_k^f V_k^T$ instead of $D_k$ and $q_k^f = \Sigma_k^f \Sigma_k^{-1} U_k^T q$ instead of $q_k$.

In the following we have chosen functions $f(x) = x^\varepsilon$ ($\varepsilon \geq 1$), so we will denote $\Sigma_k^f$ as $\Sigma_k^\varepsilon$, $D_k^f$ as $D_k^\varepsilon$, and $q_k^f$ as $q_k^\varepsilon = \Sigma_k^{\varepsilon-1} U_k^T q$. Note the notation is consistent with the simple LSI (i.e. usage of $\Sigma_k^0$). In Figure 5 see a normed visualization of the singular values modified by several functions $f(x) = x^\varepsilon$. The greater $\varepsilon$, the more quick falling of $\sigma_i^\varepsilon$.

From the semantic point of view, a convex modification of singular values means that we even more emphasize the significant concepts and even more inhibit the less significant ones. It seems that we perform a kind of an additional dimensionality reduction.

On the other side, any modification of singular values surely must increase the approximation error mentioned in Section 2. However, this kind of error is algebraical; the human-dependent effectiveness measures (e.g. the precision and the recall) are something else. We present an experimental evaluation of the $\sigma$-LSI model effectiveness in Section 5.1.

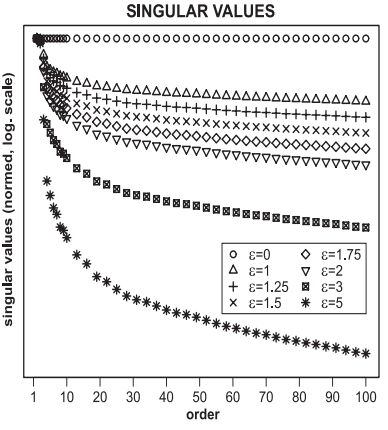


**Fig. 5.** Visualization of modified singular numbers $\sigma_i^\varepsilon$ (for different $\varepsilon$)

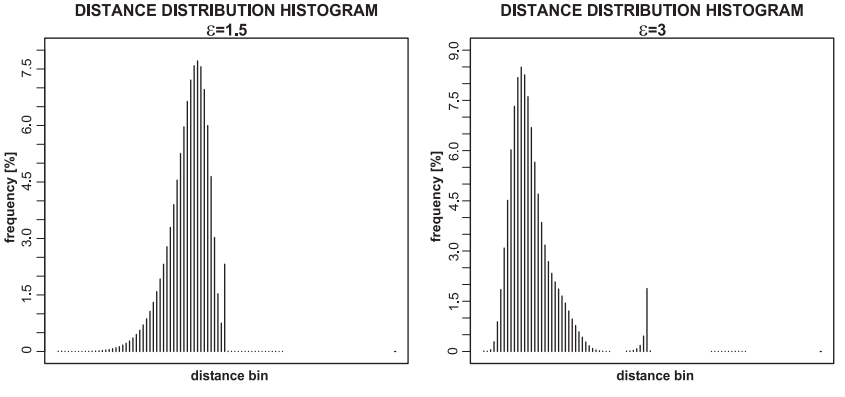


**Fig. 6.** DDHs for $D_k^{1.5}$ and $D_k^3$DDHs for $D_k^{1.5}$ and $D_k^3$

### 4.3 Intrinsic Dimensionality Reduction

In Figure 6 see distance distribution histograms for $D_k^\varepsilon$, $\varepsilon = 1.5$ and $\varepsilon = 3$. The intrinsic dimensionality for $D_k^{1.5}$ (or $D_k^3$) is $\rho = 21.22$ ($\rho = 1.72$ respectively).

In Figure 7 the intrinsic dimensionality $\rho$ of $D_k^\varepsilon$ is presented in dependence on $\varepsilon$. As we have assumed, $\rho$ is decreasing with growing $\varepsilon$, which should be reflected by a more efficient searching by MAMs. The search efficiency achieved by the M-tree is presented in Section 5.2.

## 5 Experimental Query Evaluation

For testing of our approach, we used a subset of TREC collection [21], consisting of 30,000 Los Angeles Times articles (years 1989 and 1990), from which 16,889 articles were assessed in TREC-8 ad-hoc queries (see below). The remaining arti-

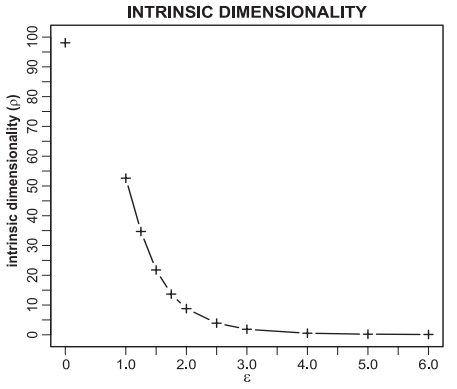


**Fig. 7.** Dependence of $\rho(D_k^\varepsilon, d_{dev})$ on $\varepsilon$

cles were added chronologically (from January to April 1989) and should provide finer LSI concepts. We indexed this collection, removing well-known stop-words and terms appearing in more than 25% of documents, thus obtaining 49,689 terms. Rank-100 SVD of the term-by-document matrix $A$ was then calculated.

### 5.1 Effectiveness

For the evaluation of $\sigma$-LSI model, we need some qualitative measures for evaluating query results. We used *precision* ($P$) and *recall* ($R$), which are calculated from set *Rel* of objects relevant to the query (usually determined by manual annotation of the collection, giving us subjective human assessment of documents' relevance) and a set *Ret* of retrieved objects. Based on these sets, we define precision and recall as:

$$P = \frac{|Rel \cap Ret|}{|Ret|}, \quad R = \frac{|Rel \cap Ret|}{|Rel|}$$

For the overall comparison of precision and recall across different methods, we can use rank lists and evaluate precision on 11 standard recall levels (0.0, 0.1, 0.2, ..., 0.9, 1.0). Since the queries may have different number of relevant documents, we can use interpolated values for each query. For complete description of this method, see e.g. [2].

Unfortunately, it was observed that with the increase of recall, the precision usually decreases. This means that when it is necessary to retrieve more relevant objects, a higher percentage of irrelevant will be probably retrieved, too. To obtain a single ratio for evaluation of the retrieval performance, we can employ a measure called $F$-score – harmonic mean of recall and precision. Determination of the maximum value for $F$ can be interpreted as an attempt to find the best possible compromise between recall and precision.

The universal version of $F$-score employs a coefficient $\beta$, by which can be the precision-recall ratio tuned. We will use the basic form of $F$ score with $\beta = 1$:

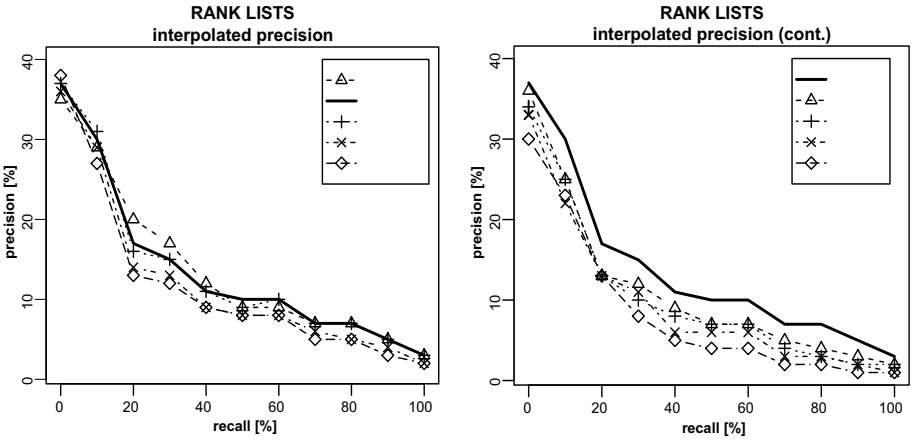


**Fig. 8.** Precision for 11 standard recall levels calculated from rank lists

$$F_\beta = \frac{(1+\beta^2)\cdot P \cdot R}{\beta^2 P + R}, \quad F = F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

To measure the effectiveness of $\sigma$-LSI, we must know the values of precision and recall for both the original method (LSI) and the modification ($\sigma$-LSI). Since we use a subset of TREC collection, we have a baseline for the effectiveness measurement via a set of predefined topics and assessed documents, called TREC Queries. TREC topics (written in SGML) contain at least the following tags:

```
<top>
   <num> Number: 401
   <title> foreign minorities, Germany
   <desc> Description:
      What language and cultural differences impede the
      integration of foreign minorities in Germany?
   <narr> Narrative:
      A relevant document will focus on ...
</top>
```

For every topic, there is a set of relevance assessments for selected documents, which indicates, whether the particular assessed document was relevant or irrelevant. The remaining unassessed documents were *assumed* irrelevant.

We used TREC-8 Ad-hoc topics 401-450 with their relevance assessments for Los Angeles Times subcollection for our task. Term weights in query vectors were calculated from term frequency (*tf*) component, the query vectors were then projected to pseudodocument space for given $\varepsilon$. The values of $\varepsilon$ have been chosen from $\{0\} \cup < 1, 9 >^5$. The cosine measure $\mathrm{SIM}_{cos}$ (deviation metric $d_{dev}$ respectively) values were calculated for both $k$-NN queries and rank lists for each TREC Query in the pseudodocument spaces.

[5] For $\varepsilon = 1$, we obtain classic LSI model with $D_k = \Sigma_k V_k^T$, which we used as a baseline; for $\varepsilon = 0$ we get simple LSI with $D'_k = V_k^T$.

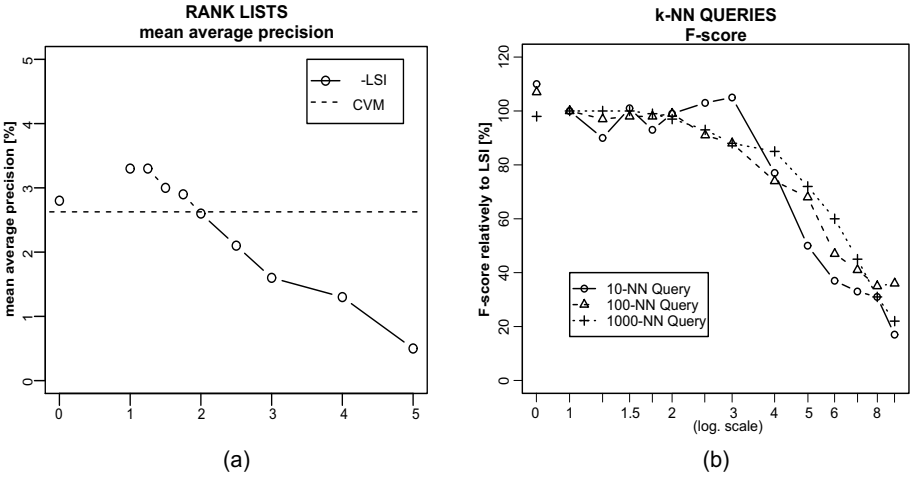


**Fig. 9.** (a) Mean average precision of $\sigma$-LSI for all relevant documents for different values of $\varepsilon$ with CVM baseline (b) $F$-score of $k$-NN queries for different values of $\varepsilon$

Firstly, we used rank lists and measured interpolated average precision of the above mentioned TREC Queries for 11 standard recall levels. The comparison for different values of $\varepsilon$ and original LSI ($\varepsilon = 1$) is addressed in Figure 8. The precision-recall curves for reasonably small values of $\varepsilon$ are very similar to classic LSI, thus the method yields similar results even with much smaller intrinsic dimensionality, which is suitable for MAMs.

Additionally, we calculated the mean average precision for all relevant documents in rank lists. The results for $\sigma$-LSI are shown in Figure 9a together with the mean average precision of corresponding CVM representation.

Secondly, we executed TREC Queries as $k$-NN queries for several values of $k$, ranging from 10 to 1000 and compared the $F$-score for different values of $\varepsilon$. Some of the results are shown in Figure 9b. We can observe, for the values of $\varepsilon < 3$ the precision and F-score seem to be well-preserved.

### 5.2 Efficiency

The motivation and main reason for introduction of the $\sigma$-LSI model is an improvement of query evaluation efficiency, when using MAMs. Among the many metric access methods, we have chosen the M-tree [9] as a "database-friendly" MAM (M-tree is a balanced, paged and dynamic structure), which we employed to index several $D_k^\varepsilon$ matrices. The matrices were stored externally (the M-tree index contained just pointers to the respective vectors in $D_k^\varepsilon$) and size of each matrix was about 12 MB. The size of each M-tree index was quite small, about 600 kB.

As search costs of $k$-NN queries, we measured the I/O costs (disk accesses) and also the realtimes. Each $k$-NN query was executed 1000 times, every time for a (new) randomly selected vector from $D_k^\varepsilon$ (i.e. as query vectors we have reused

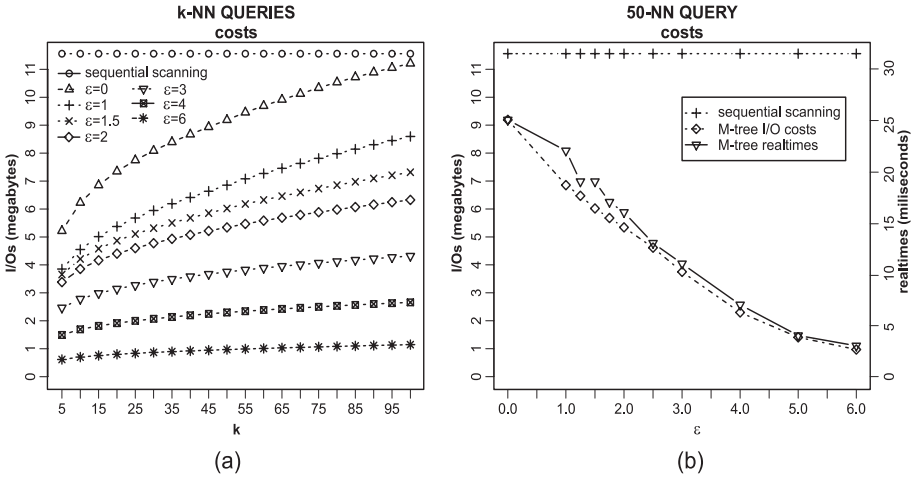


**Fig. 10.** (a) $k$-NN queries costs (b) 50-NN query costs, depending on $\varepsilon$

the pseudodocument vectors). The results were averaged. To have an efficiency baseline, we also present results for searching by simple sequential scanning of the entire matrix $D_k^\varepsilon$.

In Figure 10a see the costs of $k$-NN queries evaluation for several values of $\varepsilon$. With growing $\varepsilon$ the query evaluation is more efficient, up to 8 times for $\varepsilon = 6$ and $k = 100$, when related to $\varepsilon = 1$ (the classic LSI). Even in case when $\varepsilon = 3$ (for which the $F$-score is still well-preserved) the efficiency is improved more than twice, when compared to $\varepsilon = 1$.

The dependence of efficiency on $\varepsilon$ is presented in Figure 10b. For 50-NN queries, both I/O costs and realtimes decrease with growing $\varepsilon$. However, had we compared Figures 10b and 7, the intrinsic dimensionality drops much faster than the costs needed for processing a 50-NN query by the M-tree. This observation indicates that an "ideal" MAM should perform even better than the M-tree.

## 6 Conclusions

In this paper we have proposed $\sigma$-LSI – a novel modification of LSI model for efficient searching in document collections by metric access methods. To battle high intrinsic dimensionality, a convex modification of singular values $\sigma_i$ by calculating $\sigma_i^\varepsilon$, $\varepsilon \geq 1$ was proposed. We have shown that for reasonable values of $\varepsilon$ the intrinsic dimensionality drops quickly, while the similarity of documents is still well-preserved. In fact, we have observed that our collection seemed to yield almost the same results for $\varepsilon \leq 2.5$, while the search efficiency was doubled.

In future, we would like to apply other convex functions on singular values, testing whether they yield better global results for precision, recall and intrinsic dimensionality than the currently proposed approach. We would like test the

approach on a greater collection, too, using some probabilistic methods of LSI calculation, if needed.

Because rank-$k$ SVD is also often used on other types of data, especially images, it would be interesting to evaluate the impact of our method on other metrics (e.g. $L_2$), query results and intrinsic dimensionality in these collections, too.

Additionally, with the techniques of local dimension reduction, approximate LSI, and $\sigma$-LSI modification for better metric indexing, we may be able to build a really viable LSI index.

## Acknowledgement

This research has been partially supported by Czech Science Foundation (GAČR) grants Nr. 201/05/P036 and Nr. 201/03/1318.

## References

1. V. N. Anh, O. de Kretser, and A. Moffat. Vector-space ranking with effective early termination. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 35–42. ACM Press, 2001.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, New York, 1999.
3. M. Berry and M. Browne. *Understanding Search Engines, Mathematical Modeling and Text Retrieval*. Siam, 1999.
4. M. Berry, S. Dumais, and T. Letsche. Computation Methods for Intelligent Information Access. In *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*, 1995.
5. M. W. Berry and R. D. Fierro. Low-Rank Orthogonal Decomposition for Information Retrieval Applications. *Numerical Algebra with Applications*, 1(1):1–27, 1996.
6. C. Böhm, S. Berchtold, and D. Keim. Searching in High-Dimensional Spaces – Index Structures for Improving the Performance of Multimedia Databases. *ACM Computing Surveys*, 33(3):322–373, 2001.
7. E. Chávez and G. Navarro. A probabilistic spell for the curse of dimensionality. In *Proc. 3rd Workshop on Algorithm Engineering and Experiments (ALENEX'01), LNCS 2153*. Springer-Verlag, 2001.
8. E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Compututing Surveys*, 33(3):273–321, 2001.
9. P. Ciaccia, M. Patella, and P. Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd Athens Intern. Conf. on VLDB*, pages 426–435. Morgan Kaufmann, 1997.
10. V. Dohnal, C. Gennaro, P. Savino, and P. Zezula. D-index: Distance searching index for metric data sets. *Multimedia Tools Applications*, 21(1):9–33, 2003.
11. A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo Algorithms for Finding Low Rank Approximations. In *Proceedings of 1998 FOCS*, pages 370–378, 1998.
12. G. H. Golub and C. F. V. Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, 1996.

13. R. M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. Technical report, University of Aarhus, 1998.
14. M. L. Micó, J. Oncina, and E. Vidal. An algorithm for finding nearest neighbour in constant average time with a linear space complexity. In *International Conference on Pattern Recognition*, pages 557–560, 1992.
15. A. Moffat and J. Zobel. Fast ranking in limited space. In *Proceedings of the Tenth International Conference on Data Engineering*, pages 428–437. IEEE Computer Society, 1994.
16. J. K. P. Kanerva and A. Holst. Random Indexing of Text Samples for Latent Semantic Analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, page 1036, 2000.
17. C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proocedings of the ACM Conference on Principles of Database Systems (PODS)*, pages 159–168, 1998.
18. M. Persin. Document filtering for fast ranking. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 339–348. Springer-Verlag New York, Inc., 1994.
19. J. Ponte and W. Croft. A language modelling approach to IR. In *Proceedings of the 21 st ACM SIGIR Conference*, pages 275–281, 1998.
20. T. Skopal, P. Moravec, J. Pokorný, and V. Snášel. Metric Indexing for the Vector Model in Text Retrieval. In *Proceedings of the 11th Symposium on String Processing and Information Retrieval (SPIRE), Padova, Italy, LNCS 3246, Springer-Verlag*, pages 183–195, 2004.
21. E. M. Voorhees and D. Harman. Overview of the sixth text REtrieval conference (TREC-6). *Information Processing and Management*, 36(1):3–35, 2000.
22. P. N. Yanilos. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. In *Proceedings of Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms - SODA*, pages 311–321, 1993.
23. P. Zezula, P. Savino, G. Amato, and F. Rabitti. Approximate Similarity Retrieval with M-Trees. *VLDB Journal*, 7(4):275–293, 1998.