

# Testing Dimension Reduction Methods for Text Retrieval

Pavel Moravec

Department of Computer Science, VŠB – Technical University of Ostrava  
17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic  
pavel.moravec@vsb.cz

**Abstract.** In this paper, we compare performance of several dimension reduction techniques, namely LSI, random projections and FastMap. The qualitative comparison is based on rank lists and evaluated on a subset of TREC 5 collection and corresponding TREC 8 ad-hoc queries. Moreover, projection times and intrinsic dimensionality were measured to present a common baseline for methods' usability.

**Key words:** vector model, LSI, information retrieval, random projection, FastMap, ranked lists, TREC, intrinsic dimensionality, curse of dimensionality

## 1 Introduction

The *information retrieval* [14, 2] deals among other things with storage and retrieval of multimedia data that can be usually represented as vectors in multidimensional space. This is especially suitable for *text retrieval*, where we store a *collection* (or *corpus*) of texts. There are several models used in text retrieval, from which we will use the *vector model* [12] providing qualitatively better results than the *Boolean model* [14], which combines word matching with Boolean operators.

In the vector model, we have to solve several problems. The ones addressed in this paper are problems with the ability to index given collection, search efficiency and result set quality.

*Latent semantic indexing (LSI)* adds an important step to the indexing process. In addition to recording which terms a document contains, the method examines the document collection as a whole, to see which other documents contain some of those same terms. LSI considers documents that have many terms in common to be semantically close, and ones with few words in common to be semantically distant. However it is not suitable for huge collections and is computationally expensive, so other methods of dimension reduction were proposed. We test two of them – *Random projection*, which projects document vectors into a subspace using a randomly generated matrix, and *FastMap*, a pivot-based method based loosely on *Multi-Dimensional Scaling*. Since both of them were created for Euclidean spaces, they may not supply good results for a

different distance functions. In our case, we need to evaluate, how these methods behave when using *cosine measure*, common in text retrieval.

The rest of this paper is organised as follows. In the second section, we describe classic vector model and its problems, which may be addressed by dimension reduction. The third section explains used dimension reduction methods. In the fourth section, we briefly describe qualitative measures used for evaluation of our tests and in the fifth the projection properties. In the sixth section, we supply results of tests on a subset of TREC 5 collection. In conclusions we give ideas for future research.

## 2 Vector model

In vector model, a document  $D_j$  is represented as a vector  $d_j$  of term weights, which record the extent of importance of the term for the document.

To portrait the vector model, we usually use an  $n \times m$  *term-by-document matrix*  $A$ , having  $n$  rows – term vectors  $t_1 \dots t_n$  (where  $n$  is the total number of terms in collection) and  $m$  columns – document vectors  $d_1, \dots, d_m$ , where  $m$  is the size of collection (or *corpus*)  $C$ .

Term weights can be calculated in many different ways:  $w_{ij} \in \{0, 1\}$ ; as a membership grade to a fuzzy set; or as a product of functions of term frequency both in a document and in the whole collection [13] (usually *tf.idf* – count of term occurrences in the document multiplied by a logarithm of the inverse portion of documents containing the term). The normalisation of document vectors is sometimes applied during index generation phase to make the calculation in the retrieval phase faster.

A query  $Q$  is represented as an  $n$ -dimensional vector  $q$  in the same vector space as the document vectors. There are several ways how to search for relevant documents. Generally, we can compute some  $L_n$  metrics to represent the similarity of query and document vectors. However, in text retrieval better results can be obtained by computing similarity, usually using the *cosine measure*:

$$SIM_{cos}(d_j, q) = \frac{d_j \bullet q}{\|d_j\| \cdot \|q\|} = \frac{\sum_{i=1}^n (w_{i,j} \cdot q_i)}{\sqrt{\sum_{i=1}^n w_{i,j}^2 \cdot \sum_{i=1}^n q_i^2}}$$

As one can see, we do not only obtain documents which are considered relevant, but according to their similarity (or distance) to the query vector, we can order them and obtain a rank for every document in the answer set. If we need a metrics instead of similarity measure, we can use the deviation metric  $d_{dev}(x, y) = \arccos(SIM_{cos}(x, y))$ .

We can define a *threshold*  $t$ , too. All documents closer than  $t$  will be considered relevant, whilst the rest will be irrelevant. However, the choice of  $t$  is not exact and its value is usually determined experimentally.

The main problem of the vector model is that the document vectors have a big dimension (e.g. 150,000) and are quite sparse (i.e. most co-ordinates are zero). If we store them as classical vectors, the storage volume is huge – consider size of a term-by-document matrix consisting of 100,000 terms and 200,000 documents.

We can use existing compression schemes for the term-by-document matrix representation to decrease memory usage, but then the access time is much longer and we are limited by the fact, that we cannot access either the term or the document vectors quickly. Another way is to use combined storage with both row and column compression, but updating would still pose a problem.

The second problem is the so-called “*curse of dimensionality*”, which causes classical indexing structures like M-trees, A-trees, iDistance, etc. (see [5]), to perform in the same way or even worse than sequential scan in high dimensions. This is caused by the distribution of document vectors, which prevents partitioning into meaningful regions.

Third, the synonyms of terms and other semantically related words are not taken into account.

The first two problems can be addressed for queries containing only a few words by *inverted list*, which is in fact a compressed storage of term vectors. Only term vectors for terms contained in a query  $Q$  are loaded and processed, computing rank for all documents containing at least one of the terms at once. However, the inverted list is not efficient when searching for similar documents, because significant part of index must be processed.

### 3 Dimension reduction methods

We used three methods of dimension reduction - latent semantic indexing, random projection, and FastMap, which are briefly described bellow.

#### 3.1 Latent semantic indexing

*LSI* [3] is an algebraic extension of classical vector model. 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 vector model [3], which was proved for pure (only one topic per document) and style-free collections [11].

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.

**Theorem 1 (Singular value decomposition [3]).** *Let  $A$  is an  $n \times m$  rank- $r$  matrix and values  $\sigma_1, \dots, \sigma_r$  are calculated from eigenvalues of matrix  $AA^T$*

as  $\sigma_i = \sqrt{\lambda_i}$ . Then there exist column-orthonormal matrices  $U = (u_1, \dots, u_r)$  and  $V = (v_1, \dots, v_r)$ , where  $U^T U = I_n$  and  $V^T V = I_m$ , and a diagonal matrix  $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_r)$ , 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  $\sigma_1, \dots, \sigma_r$  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  $A$ .

**Definition 1 ([3]).** 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 1.

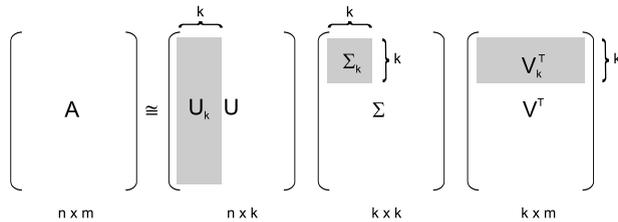


Fig. 1. rank- $k$  SVD

Rank- $k$  SVD is the best rank- $k$  approximation of the original matrix  $A$ . This means that any other decomposition will increase the approximation error, calculated as a sum of squares (*Frobenius norm*) of error matrix  $B = A - A_k$ . However, it does not implicate that we could not obtain better precision and recall values with a different approximation.

The value of  $k$  was experimentally determined as several tens or hundreds (e.g. 50–250), it is known to be dependent on the number of topics in collection, however its exact value cannot be simply determined.

The LSI is hard to compute with complexity  $O(mn^2)$  for dense and  $O(mnc)$  for sparse matrices having on the average  $c$  nonzero values per column [11]. Once computed, it reflects only the decomposition of original term-by-document matrix. If several hundreds of documents or terms have to be added to existing decomposition (*folding-in*), the decomposition may become inaccurate. Because the recalculation of LSI is expensive, so it is impossible to recalculate LSI every time documents and terms are inserted. The *SVD-Updating* [3] is a partial solution, but since the error slightly increases with inserted documents and terms, If the updates happen frequently, the recalculation of SVD may be needed soon or later.

### 3.2 Approximate LSI calculation

Several approximate methods for faster SVD calculation were offered, such as application of Monte-Carlo method [8] and using random projection (see section 3.3) of document vectors into suitable  $l$ -dimensional subspace before LSI calculation for resulting  $k$  dimensions [11].

We used the latter method, applying LSI on a matrix with reduced document vectors created by random projection. This method has a complexity of  $O(ml(l+c))$ .

### 3.3 Random projection

*Random projection* is a fast method of dimension reduction. Unlike LSI method, it does not require expensive computation of decomposition. Instead, it uses a randomly-generated projection matrix to reduce dimension of vector space. Vector from original space with dimension  $n$  is multiplied with projection matrix to obtain a vector in reduced space of dimension  $l$ , where  $l \ll n$ .

Results of dimensionality reduction by random projection are of course worse than in case of LSI and we do not obtain latent semantics. If the reduced dimension is high enough and random values building projection matrix have a zero-mean unit-variance distribution such as  $N(0, 1)$ , the Euclidean distances and angles between vectors are well-preserved.

The minimal “safe” dimension can be obtained from Johnson-Lindenstrauss lemma, however the currently known bound is still quite high and experiments showed that even smaller dimensions can be used [4]. Interestingly, the resulting dimension does not depend on original one, only on number of original points. With current best known bound, the lemma looks as follows:

**Theorem 2 (Johnson-Lindenstrauss [1]).** *For every set  $P$  of  $m$  points in  $\mathbb{R}^n$ , given  $\varepsilon > 0$ ,  $\beta > 0$  and  $l > 0, l \geq l_0 = \frac{4+2\beta}{\varepsilon^2/2-\varepsilon^3/3} \log m$ , there exists with probability at least  $1 - n^{-\beta}$  mapping  $f : \mathbb{R}^n \rightarrow \mathbb{R}^l$ , such that for all  $u, v \in P$*

$$(1 - \varepsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \varepsilon)\|u - v\|^2$$

Since we JL lemma considers only Euclidean distances, we don't have yet any bounds for cosine measure. Papadimitriou et al showed in [11] that a bound can be found for cosine measure, too. In that case

$$f(v_i).f(v_j) \leq (1 - \varepsilon)v_i.v_j + \varepsilon(v_i^2 + v_j^2).$$

Are the lengths of all  $v_k \leq 1$ , changes the inner product at most by  $2\varepsilon$ . Again, real-life data indicate that the bound is still too high and smaller dimensions can be used.

When calculating the Euclidean distances, we need to apply a scaling factor  $\sqrt{n/l}$  first, to obtain correct results because less coordinates are being used.

Classical implementations of random projection used orthogonalisation, normalisation and a dense projection matrix with Gaussian distribution. Achlioptas showed that orthogonalisation and normalisation can be skipped. He also proposed yet another powerful simplification – instead of using real coefficients of  $N(0, 1)$  distribution, he offered two possible distributions for elements  $r_{ij}$  of projection matrix  $R$  [1]:

$$r_{ij} = \sqrt{3} \cdot \begin{cases} -1 & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \\ +1 & \text{with probability } \frac{1}{6}. \end{cases} \quad r_{ij} = \begin{cases} -1 & \text{with probability } \frac{1}{2} \\ +1 & \text{with probability } \frac{1}{2}. \end{cases}$$

The  $\sqrt{3}$  component does not have to be stored in projection matrix. It can be used together with scaling factor after calculation of projected vector coordinate. If we are calculating cosine measure, it can be even discarded and instead of multiplication, we can use addition and subtraction.

We used this method in our tests, since previous results [10] indicated almost the same performance as in the case of classic random projection. The complexity of random projection is  $O(mcl)$ .

### 3.4 FastMap

FastMap [7] is a pivot-based technique of dimension reduction, suitable for Euclidean spaces.

In first step, it chooses two points, which should be most distant for calculated reduced dimension. Because it would be expensive to calculate distances between all points, it uses following heuristics:

1. A random point  $c_0$  is chosen.
2. The point  $b_i$  having maximal distance  $\delta(c_i, b_i)$  from  $c_i$  is chosen, and based on it we select the point  $a_i$  with maximal distance  $\delta(b_i, a_i)$
3. We iteratively repeat step 2 with  $c_{i+1} = a_i$  (authors suggest 5 iterations).
4. Points  $a = a_i$  and  $b = b_i$  in the last iteration are pivots for the next reduction step.

In second step (having the two pivots  $a, b$ ), we use the cosine law to calculate position of each point on line joining  $a$  and  $b$ . The coordinate  $x_i$  of point  $p_i$  is

calculated as

$$x_i = \frac{\delta^2(a_i, p_i) + \delta^2(a_i, b_i) - \delta^2(b_i, p_i)}{2\delta(a_i, b_i)}$$

and the distance function for next reduction step is modified to

$$\delta'^2(p'_i, p'_j) = \delta^2(p_i, p_j) - (x_i - x_j)^2$$

The pivots in original and reduced space are recorded and when we need to process a query, it is projected using the second step of projection algorithm only. Once projected, we can again use the original distance function in reduced space.

The complexity of FastMap is  $O(mck)$  for sparse and  $O(mnk)$  for dense matrices.

## 4 Qualitative measures of Retrieval Methods

Since we need an universal evaluation of any retrieval method, we use some measures to determine quality of such method. In case of Information Retrieval we usually use two such measures - *precision* and *recall*. Both are calculated from the number of objects relevant to the query  $Rel$  – determined by some other method, e.g. by manual annotation of given collection and the number of retrieved objects  $Ret$ . Based on these numbers we define precision ( $P$ ) as a fraction of retrieved relevant objects in all retrieved objects and recall ( $R$ ) as a fraction of retrieved relevant objects in all relevant objects. Formally:

$$P = \frac{|Rel \cap Ret|}{|Ret|} \text{ and } R = \frac{|Rel \cap Ret|}{|Rel|}$$

So we can say that recall and precision denote, respectively, completeness of retrieval and purity of retrieval. Unfortunately, it was observed that with the increase of recall, the precision usually decreases [14]. This means that when it is necessary to retrieve more relevant objects, a higher percentage of irrelevant objects will be probably obtained, too.

For the overall comparison of precision and recall across different methods on a given collection, we usually use the technique of rank lists [2], where we first sort the distances from smallest to greatest and then go down through the list and calculate maximal precision for recall closest to each of the 11 standard recall levels (0.0, 0.1, 0.2, ..., 0.9, 1.0). If we are unable to calculate precision on  $i$ -th recall level, we take the maximal precision for the recalls between  $i - 1$ -th and  $i + 1$ -th level.

## 5 Projection properties

### 5.1 Intrinsic dimensionality

The search in a collection of high-dimensional document vectors is negatively affected by a phenomenon called the *curse of dimensionality* [6], which causes

almost all regions to be overlapped by nearly every “reasonable” query region; so that searching deteriorates to sequential scan over all the classes. To judge the indexability of given dataset (in a metric space), we can use the concept of *intrinsic dimensionality* [6], defined as  $\rho = \frac{\mu^2}{2\sigma^2}$ , where  $\mu$  and  $\sigma^2$  are the mean and the variance of the dataset’s *distance distribution*. 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 (i.e. the mean is high and/or the variance is low), which means that the dataset is poorly intrinsically structured.

## 5.2 Projection stress

Sometimes, we need to verify, how well are the distances between objects preserved in the reduced dimension. To do so, we usually calculate the *stress* of projection  $f$  as

$$stress = \sqrt{\frac{\sum_{i,j=1}^m (d'(f(x_i), f(x_j)) - d(x_i, x_j))^2}{\sum_{i,j=1}^m d^2(x_i, x_j)}},$$

where  $d$  is the distance function in original and  $d'$  in projected space. The lower the stress, the better. If  $stress = 0$ , then the projection did not change the distances at all.

However, the stress function must not be overrated – even if the distances are not well-preserved, they might have been scaled by some factor, making the only difference in the choice of similarity threshold.

## 6 Experimental results

For testing of our approach, we used a subset of TREC collection [16], consisting of 16,889 Los Angeles Times articles (years 1989 and 1990) assessed in TREC-8 ad-hoc queries. We indexed this collection, removing well-known stop-words and terms appearing in more than 25% of documents, thus obtaining 49,689 terms.

We calculated random projection into dimensions  $l \in \{100, 250, 500, 1000\}$ ; both classic and approximate LSI (with random projection into  $l = 1000$ ) for  $k \in \{100, 250\}$ . For FastMap, we used for every value of  $k$  suggested 5 iterations to choose “most distant” points. Additionally we calculated FastMap for  $k = 100$  and 3 iterations (which yielded slightly worse results). Classic LSI was included to provide a baseline, since its improvement of recall is well-known.

The reduction and query projection times are shown in Table 1 <sup>1</sup>.

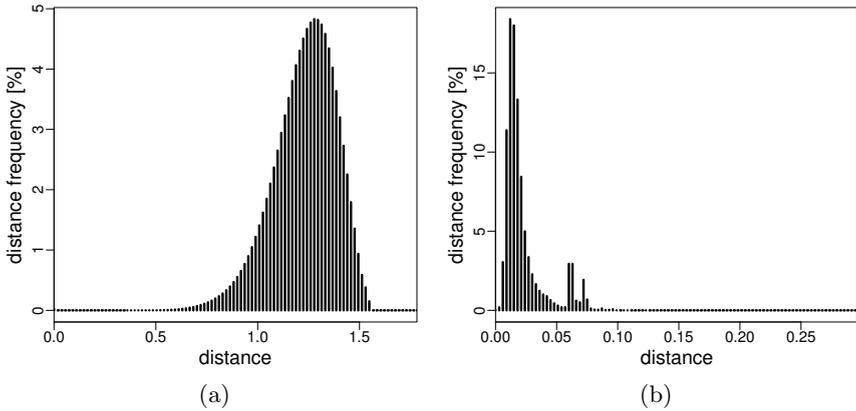
### 6.1 Analytical results

We calculated stress and intrinsic dimension for each projection method for deviation metrics.

<sup>1</sup> Since the LSI was calculated on a different computer, the LSI calculation times are only approximate

**Table 1.** Times of (a) dimension reduction and (b) projection of 45 TREC queries [s]

$k$	Reduction method			$k$	Reduction method			
	LSI	FastMap	RP		LSI	FastMap	RP	RP+LSI
50	–	771	5.43	50	0.05	0.14	0.02	0.05
100	> 5400	910	11.98	100	0.12	0.28	0.03	0.12
250	> 14400	2193	26.81	250	0.35	1.00	0.07	0.35
500	–	–	50.51	500	–	–	0.13	0.78
1000	–	–	93.44	1000	–	–	0.25	–


**Fig. 2.** Distance distribution histograms for Deviation metrics and (a) vector model, (b) FastMap

The stress, summarised in Table 2a is quite low for both LSI and random projection, however in case of FastMap are the deviations not well-preserved. From the look at distance distribution histograms of original and FastMap reduced space in Figure 2 one can observe that the distances are highly reduced. The question, if the change affects only the dissimilarity threshold will be partly solved in the next section.

In Table 2b, we can observe high intrinsic dimensions for both LSI variants and especially for random projection, whilst the intrinsic dimension for FastMap is surprisingly low. Additional tests on real data structures are required for FastMap, to verify the indexability of reduced data. In case of LSI, we recently offered a modified  $\sigma$ -LSI model [15], which trades the precision for better indexing with Metric Access Methods, namely M-trees.

**Table 2.** (a) Stress and (b) intrinsic dimensionality of reduced datasets

Reduction method					Reduction method				
$k$	LSI	FastMap	RP	RP+LSI	$k$	LSI	FastMap	RP	RP+LSI
50	0.210	0.978	0.296	0.247	50	25.1	0.2	53.3	46.8
100	0.224	0.978	0.284	0.259	100	51.1	0.5	100.2	93.9
250	0.242	0.980	0.282	0.270	250	121.1	0.9	217.1	206.4
500	–	–	0.279	0.275	500	–	–	343.7	329.7
1000	–	–	0.278	–	1000	–	–	489.3	–
					VM		← 31.8 →		

(a)

(b)

## 6.2 Query Evaluation

Firstly, we used rank lists and measured interpolated average precision of the above mentioned TREC Queries at the 11 standard recall levels. Results are summarised in Figure 3. We can see that while classic LSI provides even better results than vector model due to latent semantics, other reduction techniques try with a different success to reach the results of vector model. In our case, we got results close to vector model for random projection with  $l=1000$  and FastMap with  $k=250$ .

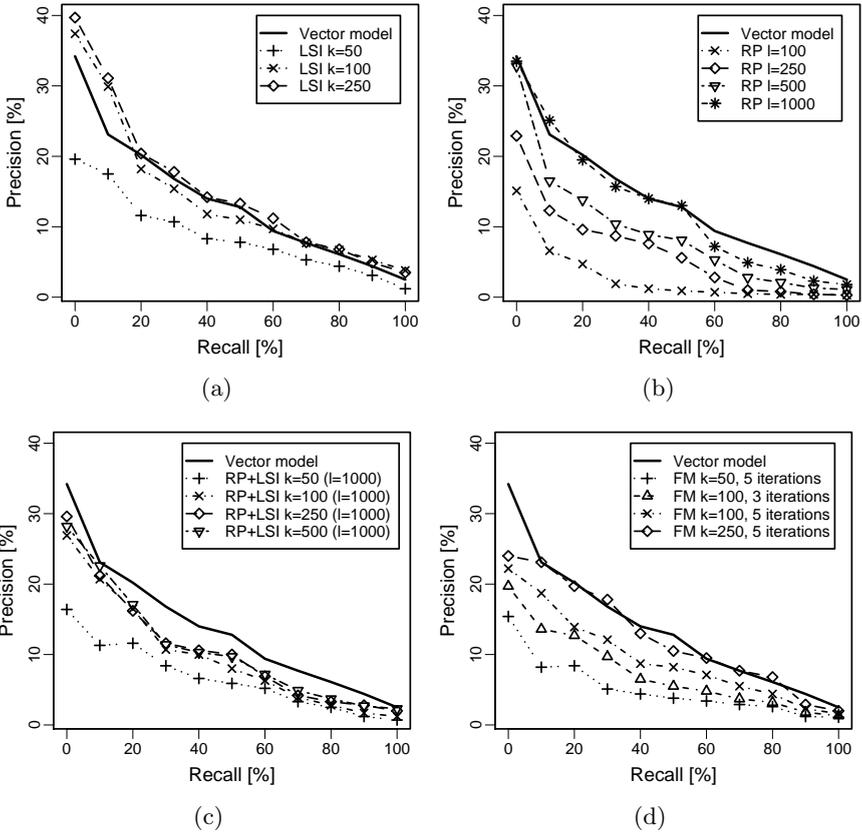
Since the important part of precision-recall curve is close to the 100% recall, we also calculated the mean average precision for all relevant documents in rank lists. The relative results against vector model (100%) are shown in Table 3.

**Table 3.** Mean average precision of different reduction methods

Reduction method				
$k$	LSI	FastMap	RP	RP+LSI
50	128%	31%	13%	85%
100	155%	58%	24%	98%
250	112%	80%	37%	79%
500	–	–	59%	77%
1000	–	–	74%	–

## 7 Conclusion

In this paper, we have compared three well-known dimension reduction methods from the view of indexability, distance preservation and results on real-live text data (using cosine measure as similarity function). Whilst the LSI is known to provide latent semantics, it is computationally expensive and in case we only need to battle the “curse of dimensionality” by reducing the dimension, FastMap



**Fig. 3.** Precision at the 11 standard recall levels: (a) LSI, (b) random projection, (c) approximate LSI calculation and (d) FastMap

or random projection may suffice. As expected, LSI was the slowest, but most exact method, followed by FastMap, which is faster but less accurate, and Random projections which are fast, but accurate only in high dimensions and have high intrinsic dimensionality.

There are some other newly-proposed methods, which may be interesting for future testing, e.g. the SparseMap [9]. Additionally, faster pivot selection technique based on text corpus properties may be considered. Finally, testing FastMap with deviation metrics on metric structures should answer the question of projected data indexability.

## References

1. D. Achlioptas. Database-friendly random projections. In *Symposium on Principles of Database Systems*, 2001.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, New York, 1999.
3. M. Berry, S. Dumais, and T. Letsche. Computation Methods for Intelligent Information Access. In *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*, 1995.
4. E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Knowledge Discovery and Data Mining*, pages 245–250, 2001.
5. 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.
6. 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.
7. C. Faloutsos and K. Lin. FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. *ACM SIGMOD Record*, 24(2):163–174, 1995.
8. 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.
9. G. R. Hjaltason and H. Samet. Properties of Embedding Methods for Similarity Searching in Metric Spaces. *IEEE transactions on pattern analysis and machine intelligence*, 25(5):530–549, 2003.
10. P. Moravec, M. Krátký, and V. Snášel. Random Projections for Dimension Reduction in Information Retrieval Systems. In *Proceedings of IMAMM'03 Conference*, 2003.
11. C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the ACM Conference on Principles of Database Systems (PODS)*, pages 159–168, 1998.
12. G. Salton. *The SMART Retrieval System – Experiments in Automatic Document Processing*. Prentice Hall Inc., Englewood Cliffs, 1971.
13. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
14. G. Salton and G. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
15. T. Skopal and P. Moravec. Modified LSI Model for Efficient Search by Metric Access Methods. In *Proceedings of ECIR'05 Conference*, Santiago de Compostela, Spain, March 2005.
16. E. M. Voorhees and D. Harman. Overview of the sixth text REtrieval conference (TREC-6). *Information Processing and Management*, 36(1):3–35, 2000.