Polynomial Filtering in Latent Semantic Indexing for Information Retrieval*

E. Kokiopoulou † Yousef Saad†

May 12, 2004

Abstract

Latent Semantic Indexing (LSI) is a well established and effective framework for conceptual information retrieval. In traditional implementations of LSI the semantic structure of the collection is projected into the k-dimensional space derived from a rank-k approximation of the original term-by-document matrix. This paper discusses a new way to implement the LSI methodology, based on polynomial filtering. The new framework does not rely on any matrix decomposition and therefore its computational cost and storage requirements are low relative to traditional implementations of LSI. Additionally, it can be used as an effective information filtering technique when updating LSI models based on user feedback.

Key words: Latent Semantic Indexing, Polynomial filtering

1 Introduction

The vast amount of electronic information that is available today requires effective techniques for accessing relevant information from it. The methodologies developed in information retrieval aim at devising effective means to extract relevant documents in a collection when a user query is given. Typically, information is retrieved by literally matching terms among a user's query and all available documents. However, information retrieval techniques based on exact literal matching may be inaccurate due to the problems of word usage. It is common that a set of different words is used to express the same concept (*synonymy*). On the other hand, a word may have several different meanings (*polysemy*) depending on the context. This word variability may obscure the conceptual structure of the collection.

LSI [7] has been successful in addressing this problem by revealing the underlying semantic content of collection documents. It is implemented by projecting the original term-by-document matrix into a reduced rank space by means of a truncated singular value decomposition, which diminishes the obscuring "noise" in word usage. Thus, the retrieval is based on the semantic content of the documents rather than their lexical content. As a consequence, a relevant document may be retrieved even if it does not share any literal terms with the user's query.

Most of the current implementations of LSI rely on matrix decompositions (see e.g., [3],[13]), with the truncated SVD (TSVD) being the most popular [1],[2]. In TSVD it is assumed that the smallest singular triplets are noisy and therefore only the largest singular triplets are used for the rank-k representation of the term-by-document matrix A. We show that a technique based on polynomial filtering can render the

*This work was supported by NSF under grant ACI-0305120, and by the Minnesota Supercomputing Institute.

†Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455. email: {nli,saad}@cs.umn.edu

same effect as TSVD, without resorting to the costly SVD decomposition. Polynomial filtering offers several advantages including low computational and storage requirements.

Relevance feedback is a mechanism that is incorporated into LSI techniques to enhance its performance. It uses information extracted from relevant documents to modify the original query in order to allow better retrieval performance. We show that a relevance feedback mechanism can be easily retrofitted to LSI techniques based on polynomial filtering.

The remaining sections of this paper are organized as follows: Section 2 provides an overview of the vector space model and the LSI approach based on truncated SVD. In Section 3 the implementation of LSI using polynomial filtering is described and in Section 4 the relevance feedback mechanism is discussed. Finally, Section 5 provides a few numerical results illustrating the advantages characterizing the proposed scheme.

2 Vector space models and LSI

Assume that we have a collection of m terms and n documents and let f_{ij} be the frequency of term i in document j. The collection can be represented by an $m \times n$ term-by-document matrix

$$A = [a_{ij}]$$

where a_{ij} is the *weight* of term i in document j. A term-by-document matrix is in general very sparse since each term occurs only in a subset of documents. A term weight has three components: local, global, and normalization. Let

$$a_{ij} = L_{ij} G_i N_j$$

where L_{ij} is the local weight for term i in document j, G_i is the global weight for term i, and N_j is the normalization factor for document j. The local weight L_{ij} is a function of how many times the term i appears in document j. The global weight G_i characterizes how often term i appears in the collection. The normalization factor N_j compensates for length variability among different documents. Since weighting schemes are critical to the performance of the vector space model, several of them have been developed in the literature, see, e.g., [4].

A query is represented as a *pseudo-document* in a similar form, $q = [q_j]$, where q_j represents the weight of term i in the query. After we have weighted the documents and the query, we can measure the similarity between the query q and a document vector d_j by computing their inner product. This similarity measure is related to the angle θ_j between these two vectors, defined from the relation,

$$\langle d_j, q \rangle = d_j^\top q = \|d_j\|_2 \|q\|_2 \cos\theta_j,$$

and it is maximized when they are collinear. Hence, ranking all documents in the collection will simply necessitate computing the n-dimensional similarity vector

$$s = A^\top q \ .$$

The vector space model is a quite simple and computationally attractive scheme for information retrieval, but it has major drawbacks which make it ineffective in practice. Its main weakness comes from its underlying assumption that the query and its relevant documents will share common terms so that it is sufficient to look for exact term matches. However, this is not always the case because of word variability. This is precisely the problem addressed by LSI.

In the classical LSI approach, the vector space model is improved by replacing the original term-by-document matrix by a low-rank approximation derived from its truncated Singular Value Decomposition (SVD). The SVD of a rectangular $m \times n$ matrix A of rank r, can be defined as

$$A = U\Sigma V^\top, \tag{1}$$

where $U = [u_1, ..., u_m]$ and $V = [v_1, ..., v_n]$ are unitary matrices and $\Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_n)$, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0$. The σ_i's are the *singular values* of A and the u_i's and v_i's are, respectively, *the left and right singular vectors* associated with σ_i, $i = 1, ..., r$. We define the i-th singular triplet of A as $\{u_i, \sigma_i, v_i\}$. It follows from the decomposition that the matrix A can be expressed as a sum of r rank-one matrices as,

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^\top \ .$$

Additionally, it is known that

$$\min_{\text{rank}(B) \leq k} \|A - B\|_F = \|A - A_k\|_F$$

where $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^\top$ [11]. It is helpful for what follows to rewrite the matrix A_k as

$$A_k = U_k \Sigma_k V_k^\top, \tag{2}$$

where U_k (resp. V_k), consists of the first k columns of U (resp. V), and Σ_k is a diagonal matrix of size $k \times k$. Thus, if we truncate the SVD to keep only the k largest singular triplets we obtain the closest (in a least-squares sense) approximation to A. This leads to the new similarity vector

$$s_k = A_k^\top q = V_k \Sigma_k U_k^\top q \ . \tag{3}$$

It is assumed that the reduced rank representation of the matrix helps reveal the underlying semantic structure of the collection and remove the noise due to word usage. The subspace spanned by the columns of U_k and V_k is called *term-space* and *document-space* respectively. In LSI, a user's query is represented as the sum of term vectors indicated by the terms included in the query, scaled by the singular values, i.e.

$$\hat{q} = q^\top U_k \Sigma_k, \tag{4}$$

where q is a vector containing the weighted term frequencies of query terms. Using the cosine similarity measure, the query vector $\hat{q}$ is compared against all the document vectors in the document space and the list of returned documents is sorted according to the value of the similarity measure.

One important issue when implementing LSI with TSVD, is the selection of the parameter k. This remains an open problem in information retrieval. In practice, k is chosen to be significantly less than $\min(m, n)$ which has the obvious advantage of reducing the computational cost of the method as well as its storage requirements. However, the matrices U_k and V_k are dense and k must be kept small in order for the TSVD method to be practically feasible. However, the optimal value of k may be much larger than the value of k that the computational constraints impose.

Because of the high computational cost of TSVD, substantial research efforts have recently been devoted to the use of alternative matrix decomposition methods that will reduce the cost and/or the storage requirements of truncated SVD approach while maintaining its performance quality (see e.g., [13]). However, these methods, indeed all methods which rely on matrix decompositions, also incur a non negligible extra cost related to the frequent updates of the decomposition when terms or documents are added or removed from the collection. This update cost can be as high as that of the entire re-computation of the SVD.

3 Latent Semantic Indexing by Polynomial Filtering

Polynomial filtering allows to closely simulate the effects of the reduced rank approximation used in LSI models. Let q be the user's query vector. In order to estimate the similarity measurement, we use a polynomial ϕ of $A^\top A$ and consider:

$$\begin{aligned} s &= \phi(A^\top A)A^\top q \\ &= \phi(V\Sigma^\top\Sigma V^\top)V\Sigma^\top U^\top q \\ &= V\phi(\Sigma^\top\Sigma)V^\top V\Sigma^\top U^\top q \\ &= V\phi(\Sigma^\top\Sigma)\Sigma^\top U^\top q. \end{aligned} \tag{5}$$

Compare the above expression with (3). Choosing the function $\phi(x)$ appropriately will allow us to interpretate this approach as a compromise between the vector space and the TSVD approaches. Assume now that ϕ is not restricted to being a polynomial but can be any function (even discontinuous). When $\phi(x) = 1\ \forall x$, then $\phi(\Sigma^\top\Sigma)$ becomes the identity operator and the above scheme would be equivalent to the vector space model. On the other hand, taking ϕ to be the step function

$$\phi(x) = \begin{cases} 0, & 0 \le x \le \sigma_k^2 \\ 1, & \sigma_k^2 \le x \le \sigma_1^2 \end{cases} \tag{6}$$

results in $\phi(\Sigma^\top\Sigma) = \begin{bmatrix} I_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$ where I_k is the identity matrix of size k and $\mathbf{0}$ is a zero matrix of an appropriate size. Then equation (5) may be re-written as:

$$\begin{aligned} s &= V\phi(\Sigma^\top\Sigma)\Sigma^\top U^\top q \\ &= \begin{bmatrix} V_k & V_{n-k} \end{bmatrix} \begin{bmatrix} \Sigma_k^\top & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} U_k^\top \\ U_{m-k}^\top \end{bmatrix} q \\ &= \begin{bmatrix} V_k\Sigma_k^\top & \mathbf{0} \end{bmatrix} \begin{bmatrix} U_k^\top \\ U_{m-k}^\top \end{bmatrix} q = V_k\Sigma_k^\top U_k^\top q \\ &= A_k^\top q \end{aligned}$$

which is precisely the rank-k approximation provided by the TSVD method.

Using the above framework when ϕ is a polynomial, the estimation of the similarity measurement simplifies to a series of matrix vector products, where the corresponding matrix and vector are very sparse. This can be implemented very efficiently using data structures that exploit sparsity. Therefore, the approach of polynomial filtering in LSI models can a result that is close to that of TSVD without resorting to the costly SVD or any other matrix decomposition. Furthermore, the need to store additional (dense or sparse) matrices as is the case in TSVD, is completely avoided as is the need to update these matrices when entries of A change.

The selection of the cut-off point is somewhat similar to the issue of choosing the parameter k in the TSVD method. However, there is a salient difference between the two: choosing a large k in TSVD may render the method much more expensive, while selecting a high cut-off in polynomial filtering does not affect cost significantly.

3.1 Approximating the step function

We now consider ways of approximating the ideal step function ϕ given in (6) using a polynomial $\hat{\phi}$. One approach would be to discretize the step function in the interval $[0, b]$, with $b \equiv \sigma_1^2$, and then to find the

coefficients of the polynomial which interpolates the produced data points in a least-squares sense. As is well-known (see also below) this approach will produce a polynomial with potentially large fluctuations between the data points resulting in a poor retrieval performance.

Another approach is to rely on Hermite interpolation by imposing smoothness conditions at both endpoints of the interval. Assume that we enforce the following conditions at endpoints 0 and b,

$$\hat{\phi}(0) = 0, \quad \hat{\phi}^{(1)}(0) = \hat{\phi}^{(2)}(0) = \cdots = \hat{\phi}^{(i)}(0) = 0$$
$$\hat{\phi}(b) = 1, \quad \hat{\phi}^{(1)}(b) = \hat{\phi}^{(2)}(b) = \cdots = \hat{\phi}^{(j)}(b) = 0$$

Using the above $i+j+2$ conditions, we can employ Hermite interpolation in order to determine the coefficients of a polynomial of degree $i+j+1$ that will satisfy the given constraints. The derived polynomial $\hat{\phi}(x)$ moves from 0 to 1, as x moves from 0 to b. It can be shown [9] that the critical point, called *inflexion point*, where $\hat{\phi}$ moves rapidly from 0 to 1 is at:

$$t_{\text{infl}} = \frac{b}{1 + j/i} \quad \text{or} \quad \frac{j}{i} = \frac{b}{t_{\text{infl}}} - 1. \tag{7}$$

Therefore, the ratio $\frac{j}{i}$ determines the localization of the inflexion point. This approach has the disadvantage that the degree of the polynomial needs to become adequately large in order for the approximation to be qualitative.

The most successful approach when approximating a function with polynomials, is the piecewise polynomial approximation where instead of using only one large degree polynomial on the whole interval, we use several smaller degree polynomials at appropriate subintervals of the original interval. The problem with the piecewise polynomials is that they cannot be easily evaluated when their argument is a matrix. Erhel et al in [9] suggest a new technique, called PPF, which approximates any piecewise polynomial filter by a polynomial in some least-squares sense. This technique is used in [9] for solving ill-conditioned linear systems in image restoration where the problem matrix is symmetric semi-positive definite with a large number of singular values close to zero and the right-hand side vector is perturbed with noise. In this paper we apply a similar technique in the totally different context of information retrieval. We provide more details about this technique in the next subsection.

3.2 Piecewise polynomial filters

In this section we give a brief description of the least-squares polynomial filtering technique. The reader is referred to [9] for a thorough analysis of the methodology. Let $\mathbf{P}_{k+1,2}$ be the set of polynomials p of degree $k+1$ such that $p(0) = p'(0) = 0$, i.e. $\mathbf{P}_{k+1,2} = \langle t^2, \cdots, t^{k+1} \rangle$. The proposed iterative algorithm builds a sequence of polynomials $\phi_k \in \mathbf{P}_{k+1,2}$ which approximates the ideal function ϕ in a least-squares sense,

$$\phi_k(t) = \sum_{j=1}^{k} \langle \phi, P_j \rangle P_j(t), \tag{8}$$

where $\{P_j\}$ is a basis of orthonormal polynomials for some L_2 inner-product, denoted by $\langle \cdot, \cdot \rangle$. The L_2-inner product can be selected appropriately so that its computation can be done without resorting to numerical integration. In order to do this, we subdivide the interval $[0, b]$ into L subintervals such that

$$[0, b] = \bigcup_{l=1}^{L} [a_{l-1, a_l}], \quad \text{with} \quad a_0 = 0, \quad a_L = b \quad (= \sigma_1^2)).$$

The inner product on $[0, b]$ is equal to the weighted sum of the inner products on each subinterval $[a_{l-1}, a_l]$. Each of these inner products in turn is computed using expansions in the basis of Chebyshev polynomials of the approximation. Since Chebyshev polynomials are orthogonal in each sub-interval, numerical integration is avoided.

A Krylov subspace [12, ch.6] $\mathcal{K}_m$ of a square matrix C with respect to r_0, is defined as

$$\mathcal{K}_m(C, r_0) \equiv \text{span}\{r_0, Cr_0, C^2 r_0, \ldots, C^{m-1} r_0\}.$$

Observe now that the filtered vector $\phi_k(C)r$, where $C = A^\top A$ and $r = A^\top q$, belongs to the Krylov subspace

$$\mathcal{R}_k(C, r) \equiv \mathcal{K}_k(C, C^2 r).$$

In order to compute the basis of orthogonal polynomials $\{P_j\}$ we use the Stieljes procedure [6] for constructing orthogonal polynomials using a 3-term recurrence

$$P_{j+1}(t) = \frac{1}{\beta_{j+1}} \left[t P_j(t) - \alpha_j P_j(t) - \beta_j P_{j-1}(t) \right],$$

for $j = 1, \ldots, m$. There is a strong connection between the filtered space $\mathcal{R}_k(C, r)$ and the polynomial space $\mathbf{P}_{k+1,2}$. The correspondence is:

$$\phi_k \in P_{k+1,2} \rightarrow \phi_k(C)r \in \mathcal{R}_k(C, r). \tag{9}$$

This mapping has the important advantage that we can provide the Krylov subspace $\mathcal{R}_k(C, r)$ with an inner product that is derived from the polynomial space. Furthermore, we can compute the Lanczos [5] sequence $v_j = P_j(C)r$, via a 3-term recurrence where the scalars α_j's and β_j's are computed by the Stieljes procedure. The sequence of vectors $r_k = \phi_k(C)r$ is in the filtered space $\mathcal{R}_k(C, r)$. If we let $\gamma_j = \langle \phi, P_j \rangle$ then using equation (8) we get the expansion

$$r_k = \sum_{j=1}^{k} \gamma_j v_j.$$

At this point we can describe the algorithm PPF which is sketched in Table 1. Observe that the same scalars α_k and β_k are used in the Stieljes procedure (line 9) and the Lanczos process (line 10). This is due to the mapping (9).

4 Relevance feedback in Latent Semantic Indexing

Standard implementations of LSI offer the users the ability to associate terms and document according to their preferences. Substantial research efforts (see e.g. [14], [8]) have been devoted to improving LSI with the additional feature of allowing the model to update itself based on user feedback. This can be accomplished via the *relevance feedback* mechanism. Relevance feedback uses the terms in relevant documents to modify the user's original query to achieve better retrieval performance. In the sequel, we analyze the relevance feedback in the TSVD implementation of LSI and then describe how the relevance feedback mechanism can be retrofitted into a polynomial filter–based LSI method.

4.1 Relevance feedback in TSVD

Let $d \in \mathbf{R}^n$ be a vector whose nonzero elements are indices specifying the relevant document vectors. In TSVD the relevant feedback query is modified from the original query given in (4) by adding to it the sum of relevant documentv vectors. In particular, the modified query is

$$\tilde{q} = q^\top U_k \Sigma_k + d^\top V_k. \tag{10}$$

Algorithm **PPF**: **Piecewise Polynomial Filter**
Input: ϕ piecewise polynomial on $[0, b]$
Output: the approximation r_k to the filtered vector $\phi(C)r$.
1. $\beta_1 = \langle t^2, t^2 \rangle^{\frac{1}{2}}$
2. $P_1(t) = \frac{1}{\beta_1} t^2$
3. $\gamma_1 = \langle \phi, P_1 \rangle$
4. $v_1 = \frac{1}{\beta_1} C^2 r, \ v_0 = 0$
5. $r_1 = \gamma_1 v_1$
6. **For** $k = 1, ...$ **Do:**
7. Compute $tP_k(t)$
8. $\alpha_k = \langle tP_k, P_k \rangle$
9 $\mathcal{S}_k(t) = tP_k(t) - \alpha_k P_k(t) - \beta_k P_{k-1}(t)$
10. $s_k = Cv_k - \alpha_k v_k - \beta_k v_{k-1}$
11. $\beta_{k+1} = \langle \mathcal{S}_k, \mathcal{S}_k \rangle^{\frac{1}{2}}$
12. $P_{k+1}(t) = \frac{1}{\beta_{k+1}} \mathcal{S}_k(t)$
13. $v_{k+1} = \frac{1}{\beta_{k+1}} s_k$
14. $\gamma_{k+1} = \langle P_{k+1}, \phi \rangle$
15. $r_{k+1} = r_k + \gamma_{k+1} v_{k+1}$
16. **End**

Table 1: The PPF algorithm for least-squares polynomial filtering.

Therefore the new similarity vector used for ranking the collection documents becomes

$$s' = V_k \tilde{q}^\top = V_k(\Sigma_k U_k^\top q + V_k^\top d) = s + V_k V_k^\top d$$

where s is the original similarity vector. Assuming that A has full rank, we can expand d in the basis of the right singular vectors v_i of A. Then $d = \sum_{i=1}^{n} \gamma_i v_i$, where $\gamma_i = v_i^\top d$ and since $V_k = [v_1, ..., v_k]$ it holds that,

$$\begin{aligned} V_k V_k^\top d &= V_k V_k^\top \left[\sum_{i=1}^{n} \gamma_i v_i \right] \\ &= V_k V_k^\top \left[\sum_{i=1}^{k} \gamma_i v_i + \sum_{i=k+1}^{n} \gamma_i v_i \right] \\ &= \sum_{i=1}^{k} \gamma_i v_i \end{aligned}$$

since $V_k V_k^\top$ is an orthogonal projector. Therefore, the relevance feedback component added to the new similarity measurement s' has components only in the directions of the right singular vectors associated with the k largest singular values.

4.2 Relevance feedback with polynomial-filters

Following the analysis of Section 3, the polynomial ϕ has been designed such that when it is applied on a vector d, it dampens its components that correspond to the smallest singular values. Therefore, it is possible to use polynomial filtering to produce the action of the projection $V_k V_k^\top d$ employed in the relevance feedback mechanism of TSVD, without having the matrix V_k available. Observe that:

$$\phi(A^\top A)d = \phi(V \Sigma^\top \Sigma V^\top)d = V \phi(\Sigma^\top \Sigma) V^\top d \ .$$

	MEDLINE	CRANFIELD	TIME
No of doc.	1033	1398	425
No of terms	8322	5204	13057
No of queries	30	225	83
N_{avg}	23.2	8.17	3.9
Sparsity (%)	0.63	1.08	1.6

Table 2: Characteristics of the datasets. N_{avg} denotes the average number of relevant documents per query.

Choosing ϕ as in equation (6) it follows that,

$$\begin{aligned} V\phi(\Sigma^\top\Sigma)V^\top d &\approx \begin{bmatrix} V_k & V_{n-k} \end{bmatrix} \begin{bmatrix} I_k & 0 \\ 0 & 0 \end{bmatrix} V^\top d \\ &= \begin{bmatrix} V_k & 0 \end{bmatrix} \begin{bmatrix} V_k^\top \\ V_{n-k}^\top \end{bmatrix} d = V_k V_k^\top d \ . \end{aligned}$$

Therefore, polynomial filtering allows to incorporate relevance feedback in LSI models in an efficient manner without resorting to the costly SVD. The numerical results which follow, indicate that the retrieval performance using polynomial filtering techniques both for LSI and for LSI with relevance feedback, can compete with the TSVD implementations. At the same time, the computational and storage costs of these methods can be much lower than those of the SVD-based counterparts. For example, in terms of storage, the new methods require only a handful of vectors to be saved. Also there is no factorization to update when the matrix A changes.

5 Numerical results

In this section we compare the retrieval performance of the following four methods: TSVD-RF and TSVD implementing LSI with truncated SVD with and with relevance feedback respectively, as well as PPF-RF and PPF implementing LSI via polynomial filtering with and without relevance feedback accordingly. The PPF methods were implemented in C++ and the TSVD methods were implemented in Fortran 77. For the truncated SVD calculation we used the `dlansvd_irl` routine from PROPACK[1], a Fortran 77 software package for large-scale SVD computations based on Lanczos bidiagonalization.

The numerical experiments were performed under Linux 2.4.21 in a workstation with a PIII processor @ 933 MHz, 256 KB cache and 512 MB memory. We used the MEDLINE, CRANFIELD and TIME collections, which are publicly available[2]. The characteristics of the three datasets are tabulated in Table 2. Notice that all matrices are very sparse, with the number of non zero entries less than 2%. The term-by-document matrices for all collections were created using the TMG tool [15]. In the parsing phase, we employed the option of stemming [10] and as a stoplist, SMART's English stoplist.

In order to introduce the performance metric used in the experiments, we need to define two important factors, *precision* and *recall*. Consider a ranked list of documents returned by an information retrieval method. Observing the first i (window size=i) documents, we define precision as the ratio $\mathcal{P}_i = \frac{r_i}{i}$, where r_i is the number of relevant documents among the top i documents. Similarly, we define recall as the ratio $\mathcal{R}_i = \frac{r_i}{r_q}$, where r_q is the total number of relevant documents to the specific query in the collection. The last number can be estimated by the relevance judgement for each query that comes with each test collection. In

[1] `http://soi.stanford.edu/~rmunk/PROPACK/index.html`

[2] All collections are available from `ftp://ftp.cs.cornell.edu/smart/`

other words, precision is the proportion of the top i documents that are relevant and recall is the proportion of all relevant documents in the collection that are retrieved. Both precision and recall take values in the interval $[0, 1]$.

A common performance metric used in literature is the n-point interpolated *average precision* defined as

$$\frac{1}{n}\sum_{i=0}^{n-1}\bar{p}\left(\frac{i}{n-1}\right), \text{ where } \bar{p}(x) = \max_{(r_i/r_n)\geq x}\mathcal{P}_i. \tag{11}$$

In practice, the $n = 11$-point interpolated average precision is used for each query separately. Another typical performance metric for collection with multiple queries is the precision-recall diagram, where we measure the pairs $(\mathcal{R}_i, \mathcal{P}_i)$ at various values of the window size i. This is done for each query individually and then averaged over all queries.

In all the following experiments we employed the *log* local weighting scheme and *entropy* global term weighting scheme [4]. The *log* × *entropy* weighting scheme has been experimented over five test collections and was 40% more effective than raw term weighting [2]. Additionally, we normalized the columns of A. In what follows, we model the inflexion point of Hermite interpolation applied on the whole interval $[0, b]$ as $t_{\text{infl}} = \eta b$, where η expresses the percentage of the interval where the cut-off point will be located. This is somewhat similar to choosing k in the TSVD method. Then we use equation (7) in order to determine the ratio $j/i = 1/\eta - 1$.

In PPF we used two subintervals. Thus, we have to choose an additional parameter a which is the right endpoint of the first interval. The first piece is a Hermite interpolating polynomial and the second piece is equal to 1. The parameter a was set to $2\eta b$. We kept the same inflexion point $t_{\text{infl}} = \eta b$ as described in the case of Hermite interpolation applied on the whole interval, which results in $j/i = 1$.

Example A. In this example we explore ways to determine the degree of the polynomial in the PPF method. Given the desired inflexion point we exploit the ratio j/i of equation (7). Thus, we have the flexibility to determine the actual value of i and j provided that their ratio satisfies equation (7). Therefore the degree of the resulting polynomial will be in the general case $\lceil \gamma i + \gamma j + 1 \rceil$, where γ is a scalar multiplicative factor that acts both on i and j and has no effect on the ratio between them.

We plot precision-recall curves for various values of γ, experimented on all datasets. We used $k = 50$, which resulted in $\eta = 0.0929$ for the MEDLINE, $\eta = 0.0779$ for the TIME and $\eta = 0.0484$ for the CRANFIELD collection. Tables 3 and 4 depict the mean average precision and the corresponding *query times* achieved by the PPF and PPF-RF methods. We use the term *query time* to denote the total time required by a certain method to process all the available queries in the collection.

Observe in Figure 1 for the MEDLINE dataset, that increasing γ to more than 2 is not very helpful in terms of precision-recall and mean average precision. On the other hand, according to Figure 2 corresponding to the CRANFIELD dataset, the precision-recall curve for $\gamma = 3$ is much better than the one corresponding to $\gamma = 2$. This is also illustrated in the PPF-RF case, by the mean average precision improvement from 0.5172 to 0.6045, at a cost of timing increase from 57.59 to 84.91 sec. Additionally, it appears that further increase of γ is worthless. Concerning the TIME collection depicted in Figure 3, the choice of $\gamma = 2$ seems to be the most beneficial in terms of precision and cost. It also appears that for larger values of γ than 4 may lead to poor performance due to overfitting.

Example B. In this example we explore the effectiveness of all methods, in terms of computational cost and precision. We experiment with various values of the dimension k of the TSVD, and measure the mean average precision and the corresponding query time. The parameter η which is responsible for the

	MEDLINE		CRANFIELD		TIME	
γ	PPF	PPF-RF	PPF	PPF-RF	PPF	PPF-RF
1	0.814	0.961	0.270	0.368	0.558	0.972
2	0.835	0.969	0.336	0.517	0.664	0.998
3	0.819	0.928	0.383	0.604	0.681	0.998
4	0.817	0.846	0.388	0.661	0.568	0.952

Table 3: Mean average precision for $\gamma = 1, 2, 3$ and 4, experimented on all datasets.

	MEDLINE		CRANFIELD		TIME	
γ	PPF	PPF-RF	PPF	PPF-RF	PPF	PPF-RF
1	2.28	4.5	24.41	43.71	10.95	19.34
2	3.21	5.93	31.1	57.59	14.09	26.08
3	3.94	7.12	37.9	84.91	16.54	31.71
4	4.66	8.47	44.28	90.6	20.12	37.27

Table 4: Query times (sec) for $\gamma = 1, 2, 3$ and 4, experimented on all datasets.

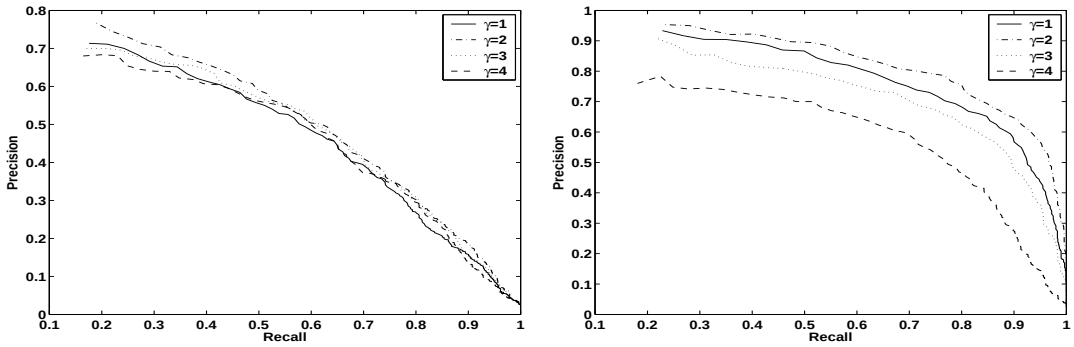

Figure 1: Precision-recall curves for various values of γ, on the MEDLINE collection. Left plot: PPF. Right plot: PPF-RF.

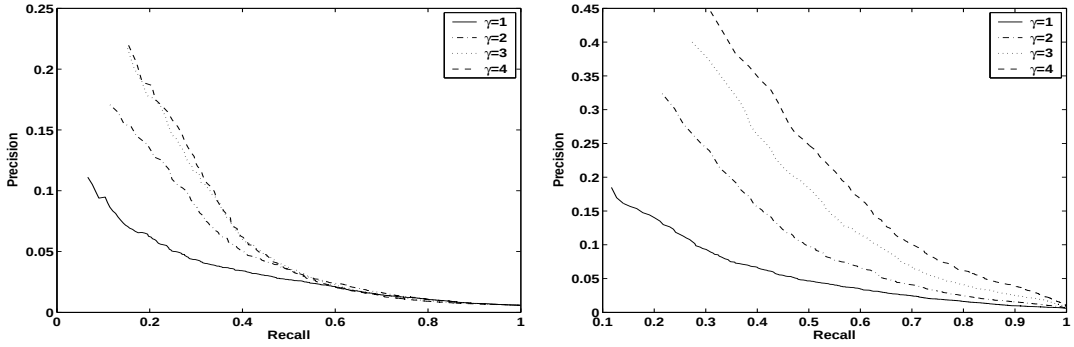

Figure 2: Precision-recall curves for various values of γ, on the CRANFIELD collection. Left plot: PPF. Right plot: PPF-RF.

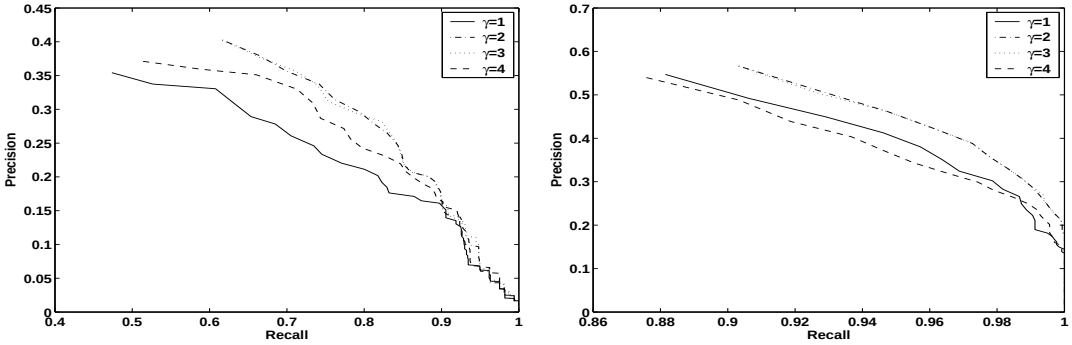

Figure 3: Precision-recall curves for various values of γ, on the TIME collection. Left plot: PPF. Right plot: PPF-RF.

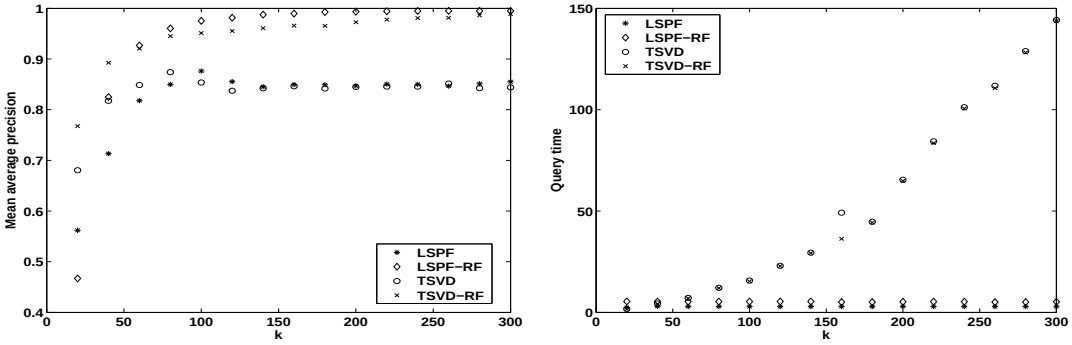

Figure 4: A comparison of all methods for $k = 20 : 20 : 300$, on the MEDLINE collection (30 queries). Left plot: mean average precision. Right plot: timings (in sec).

localization of the inflexion point, is determined by $\eta = \frac{\sigma_k^2}{b}$. Using equation (7) this results in $i = j$; we employed $\gamma = 3$.

Figure 4 illustrates the mean average precision and corresponding query times versus $k = 20 : 20 : 300$, experimented on the MEDLINE collection. Observe that the PPF methods compete their TSVD counterparts in terms of precision, and clearly outperform them in terms of query times. Figures 5 and 6 illustrate the same measurements for the CRANFIELD and TIME datasets respectively. In both cases, notice again that the PPF methods offer comparable precision to TSVD methods, at a much lower cost. Finally, observe that in the CRANFIELD case, the query time of PPF intersects the timing curve of TSVD at a higher level than in Figure 4. This is due to the large number of queries (225) of the current collection.

Example C. In the sequel, we proceed with another experiment which compares all methods in terms of precision-recall curves. We employed a typical value for k, namely $k = 50$ for all datasets. The results are illustrated in Figure 7. Notice that in the majority of cases, PPF methods achieve better results than the TSVD methods. It seems that both PPF methods present much better retrieval performance than their TSVD counterparts, especially for the CRANFIELD and TIME datasets. Observe also in the TIME plot, that the recall levels (horizontal axis) are high from the very beginning. This is due to the fact that $N_{\text{avg}} = 3.9$, as illustrated in Table 2. This means that the average number of relevant documents per query is modest, causing the initial levels of recall to be high.

We have begun to explore the interesting issue of the effectiveness of the PPF methods for much larger

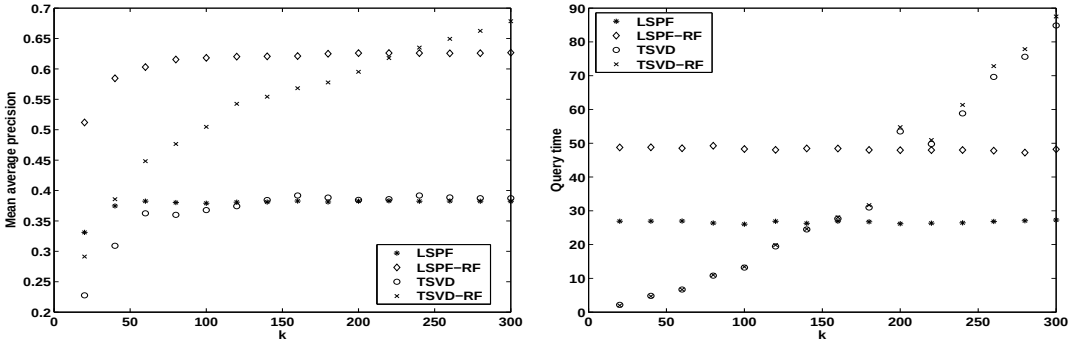

Figure 5: A comparison of all methods for $k = 20 : 20 : 300$, on the CRANFIELD collection (225 queries). Left plot: mean average precision. Right plot: timings (in sec).

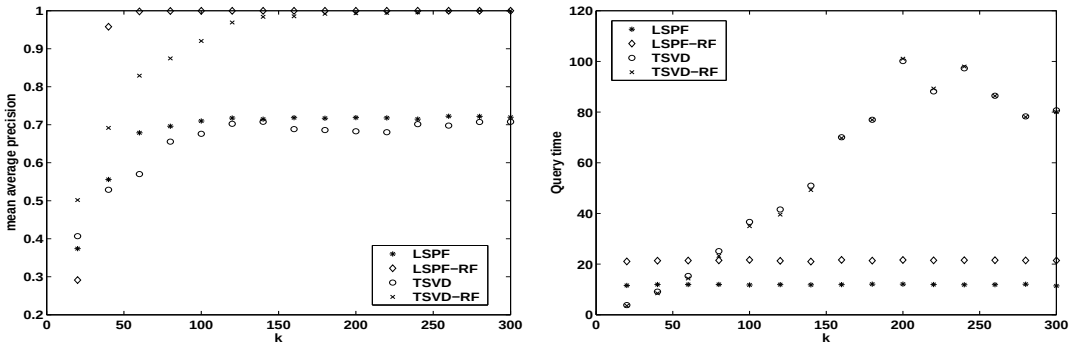

Figure 6: A comparison of all methods for $k = 20 : 20 : 300$, on the TIME collection (83 queries). Left plot: mean average precision. Right plot: timings (in sec).

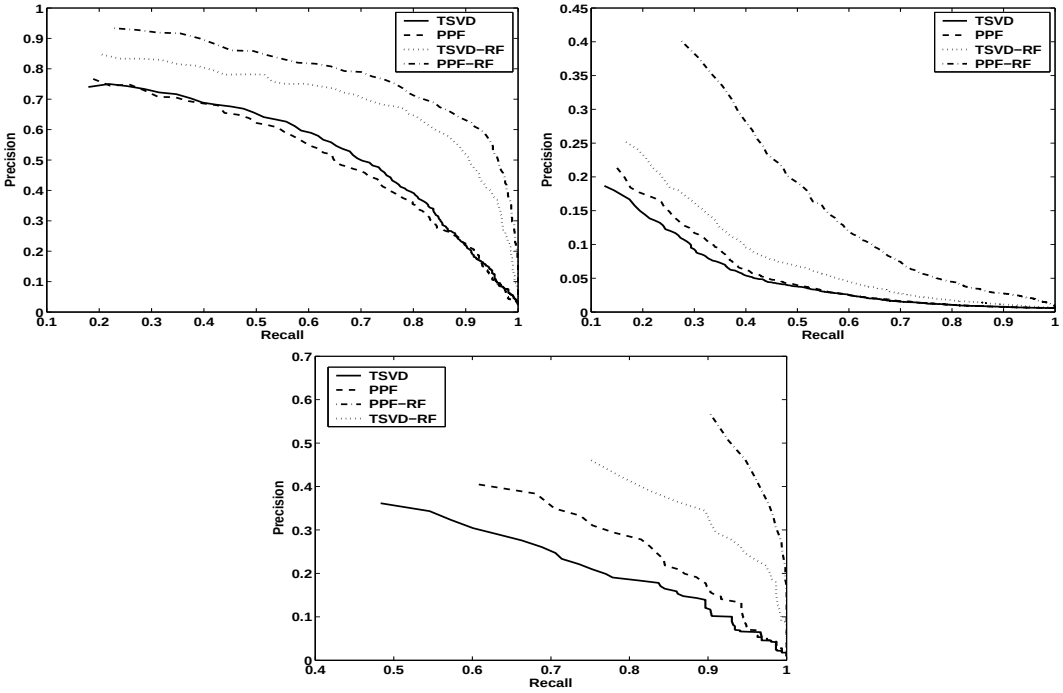

Figure 7: Precision-recall curves for all methods averaged over all queries. Top-Left plot: MEDLINE dataset. Top-Right plot: CRANFIELD dataset. Bottom: TIME dataset.

data sets, such as the TREC[3] collections. Though we expect that the conclusions will be similar, we also note that evaluating the performance of LSI-related methods on such sets is challenging in the absence of reliable and comprehensive relevance information.

6 Conclusion

A new methodology for implementing LSI was presented, which does not rely on any matrix decomposition of the term-by-document matrix. The method has appealing computational cost and storage requirements. In addition, because there is no matrix factorization, there is no need for any updates as is the case for methods based on SVD or other decompositions. The numerical experiments indicate that the proposed framework has robust retrieval performance characteristics and is very competitive with the truncated SVD implementations of LSI. Finally, the method is able to incorporate relevance feedback allowing even better retrieval performance.

References

[1] M. Berry and M. Browne. *Understanding search engines.* SIAM, 1999.

[2] M. Berry, S. Dumais and G. O' Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573-595,1995.

[3] `http://trec.nist.gov/data.html`

[3] M. Berry and R. Fierro. Low-rank orthogonal decompositions for information retrieval applications. *Numer. Lin. Alg. Appl.*, 1:1-27,1996.

[4] E. Chisholm and T. Kolda. New term weighting formulas for the vector space method in information retrieval. Technical report, Oak Ridge National Laboratory, 1999.

[5] J. Cullum and R. Willoughby. Computing eigenvalues of very large symmetric matrices–an implementation of a Lanczos algorithm with no reorthogonalization. *J. Comput. Phys.*, 44:329-358,1981.

[6] P. Davis. *Interpolation and approximation*. Blaisdell, Waltham, MA, 1963.

[7] S. Deerwester, S. Dumais, G. Furnas, T. Landauer and R. Harshman. Indexing by latent semantic analysis. *J. Soc. Inf. Sci.*, 41:391-407,1990.

[8] S. Dumais. Improving the retrieval of information from external sources. *Behav. Res. Methods, Instr. Comput.*, 23:229-236, 1991.

[9] J. Erhel, F. Guyomarc and Y. Saad. Least-squares polynomial filters for ill-conditioned linear systems. Technical report umsi-2001-32. Minnesota Supercomputing Institute. 200 Union Street S.E.. Minneapolis. MN 55455, 2001.

[10] W. Frakes. Stemming algorithms. In *Information Retrieval: Data Structures and Algorithms*, pages 131-151, Englewood Cliffs, NJ, 1992. Prentice Hall.

[11] G. H. Golub and C. Van Loan. *Matrix Computations, 3rd edn.* The John Hopkins University Press, Baltimore, 1996.

[12] Y. Saad. *Iterative methods for sparse linear systems*. 2nd edition, SIAM, 2003.

[13] T. Kolda and D. O' Leary. A semi-discrete matrix decomposition for latent semantic indexing in information retrieval. *ACM Transactions on Information Systems*, 16(4):322-346, October 1998.

[14] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *J. Amer. Soc. Info. Sci.*, 41:288-297,1990.

[15] D. Zeimpekis and E. Gallopoulos. TMG: A MATLAB-based term-document Matrix Constructor for Text Collections. Technical report, Comp. Eng. and Inf. Dept., University of Patras, December 2003.