Information Retrieval using the Singular Value Decomposition and Krylov subspaces

Katarina Blom

January 25, 1999

Department of Mathematics
Chalmers University of Technology

ISSN 0347-2809
NO 1999-5

Information Retrieval using the Singular Value Decomposition and Krylov subspaces

Katarina Blom

Department of Mathematics
Chalmers University of Technology
S-412 96 Göteborg, Sweden
blom@math.chalmers.se

Abstract. We present two algorithms for information retrieval. The first algorithm is an extension of the vector space model (or Latent Semantic Indexing) with relevance feedback added to it.

The second algorithm is a Krylov subspace method. Bidiagonalizing schemes are used to generate (very) low rank approximations of a large term document matrix. A new similarity measure used to rank the documents is presented. Two types of relevance feedback approaches used with the algorithm are discussed.

Keywords. Information Retrieval, Relevance feedback, Lanczos algorithm, Latent Semantic indexing, LSI, Vector space model, Singular value decomposition, SVD, Krylov subspace.

Acknowledgements. I would like to thank my supervisor Axel Ruhe for his encouragement, source of inspiration and for guiding me. I am greatful to Thomas Ericsson for good advice, to Michael Berry (University of Tennessee) for suplying a parser and for good advice, and to Osni Marques (Berkeley Lab) for a huge matrix. Special thanks to my mother and to Agneta Steffen for constant support.

Contents

1 Background and definitions

1.1 Information retrieval

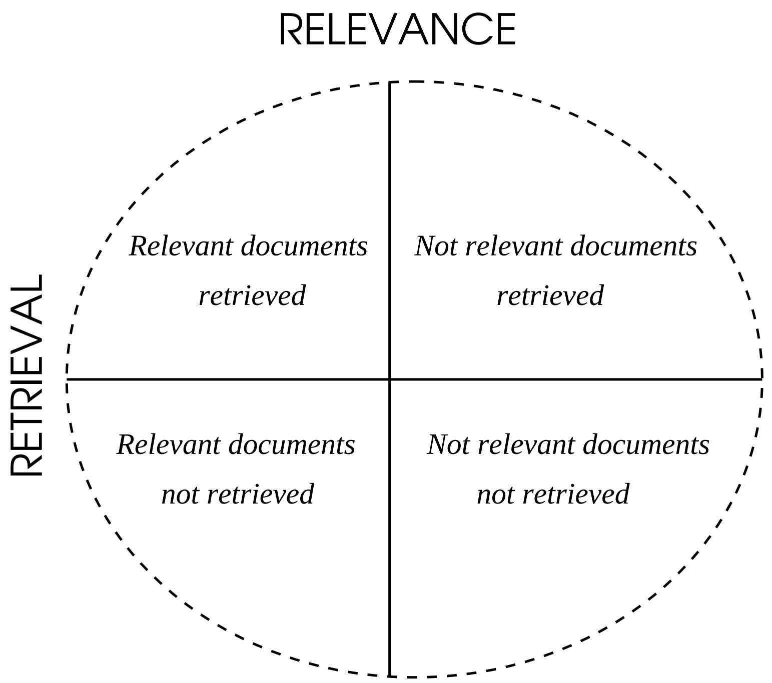
An information retrieval system matches user queries (formal statements of information needs) to documents stored in a database. Documents are often textual but may also contain other types of data such as pictures and graphs. Often the documents themselves are not stored directly in the system, but are represented in some other way that is easy to store and retrieve.
For example, a collection of articles may be represented by their titles, authors and abstracts. So instead of storing (and searching through) the entirety of every article when matching a query, the titles, authors and abstracts are searched through. This is typically done for reducing the size of the database and thus the searching time. The entirety of the articles will be returned upon question from the user, but the actual search will be performed on the titles, authors and abstracts.

Many universities and public libraries use information retrieval systems to provide access to books, journals and other documents. Dictionary and encyclopedia databases are widely available. Some of the most common search engines on the Internet may be thought of as information retrieval systems.

An information retrieval system has to support certain basic operations. There must be a way to enter documents into the database, and to delete them. There must also be a way to search for documents and to present them to the user. The performance of information retrieval systems can be evaluated in many ways. A few will be mentioned here:

- Retrieval efficiency is often measured by *precision* and *recall*. When a user issues a search for information on a topic, the system will start to give back documents that are relevant from the system's point of view. From the user's perspective the total database will be divided logically into four parts. There will be relevant and irrelevant documents retrieved. And among the documents not retrieved there will be both relevant and irrelevant documents. Precision is the ratio of the number of relevant documents retrieved for a given query over the total number of documents retrieved, and recall is the ratio of relevant documents retrieved over the total number of relevant documents for that query. Precision measures the overhead for a user associated with a particular search, and recall measures how well the system is able to retrieve relevant documents. Precision and recall are usually inversely related (when precision goes up, recall typically goes down and vice versa). Both precision and recall will take on values between 0 and 1 (ideally both precision and recall should be 1). Different judges will assign different relevance values to a document retrieved in response to a given query, so both precision and recall are subjective measures. The denominator in recall is (of course) unknown in operational systems.

- Execution efficiency is measured by the time it takes for an information retrieval system to perform a computation. This is the time needed for the system to perform a search, or for database maintenance operations (adding and deleting documents).
- Storage efficiency is measured by the number of bytes needed to store the data.



From the user's perspective the total database will be divided logically into four parts. There will be relevant and irrelevant documents retrieved. And among the documents not retrieved there will be both relevant and irrelevant documents. When performing a search the two parts *Relevant documents not retrieved* and *Not relevant documents retrieved* should be minimized. Recall will measure the size of *Relevant documents not retrieved* and precision will measure the size of *Not relevant documents retrieved*. The retrieved documents has been seen by the user and the not retrieved documents has not been seen by the user

There are quite a few different test collections that can be used for experiments. A test collection typically consists of a document database, a set of queries, and judgements relevant to the queries. The test collections we have used for our experiments are described in Appendix A.

There are various ways to organize an information retrieval system; the vector space model is one of them.

1.2 Vector space model

In the vector space model, text documents and user queries are represented by m-dimensional column vectors. The dimension m is the number of unique terms in the collection. We let the vectors of all the n documents build up a term document matrix $A = [a_{ij}] \in \mathcal{R}^{m \times n}$ where

$$a_{ij} = \begin{cases} weight_{ij} & \text{if term } i \text{ is present in document } j \\ 0 & \text{otherwise.} \end{cases}$$

There are several ways to compute the weights in the term document matrix. A few suggestions will be given below. The term document matrix will typically be quite large and very sparse.

A user's query will be represented by a column vector $q = [q_i]$, of length m.

$$q_i = \begin{cases} weight_i & \text{if term } i \text{ is present in the query} \\ 0 & \text{otherwise.} \end{cases}$$

To determine which documents best match a query, a variety of similarity measures can be used to rank the documents. One such similarity measure is the inner product (dot product) between the query vector and the document vectors in the term document matrix, [1]

$$\text{sim}(q, Ae_j) = |q^T(Ae_j)| = |\sum_{i=1}^{m} q_i a_{ij}|, \quad j = 1 \ldots n.$$

Another similarity measure is the cosine of the angles between the query vector and a document vector

$$\text{sim}(q, Ae_j) = \frac{|q^T(Ae_j)|}{\|q\|_2 \|Ae_j\|_2}.$$

If the query and the document are totally unrelated, they will not have any terms in common. Their corresponding vectors will be orthogonal and the value will be 0. As the cosine above approaches 1, the query vector and the document vector will be coincident.

A variety of experiments using different similarity measures combined with different types of normalizations of the document and query vectors have been performed, and it seems to be important to normalize the within-document weights in some way (i.e. to scale the columns in the term document matrix), both to moderate the effect of high-frequency terms in a document and to compensate for document length [6].

Local and global weightings are applied to increase or decrease the importance of terms within or among documents. The entries a_{ij} in the term document matrix can be computed using different combinations of local and global weightings. Let

$$a_{ij} = l_{ij} g_i$$

where l_{ij} is the local weighting for term i in document j, and g_i is the global weighting for term i. Some local weightings are:

- $l_{ij} = \dfrac{n_{ij}}{m_j}$, where n_{ij} is the number of occurrencies of term i in document j and m_j is the number of terms in document j.
- $l_{ij} = 1$ if term i is present in document j, zero otherwise.
- $l_{ij} = \dfrac{\log_2(\mathit{freq}_{ij} + 1)}{\log_2 \mathit{length}_j}$, where freq_{ij} is the frequency of term i in document j and length_j is the number of terms in document j (Harman [8]).

[1] Ae_j denotes the j:th column of A.

Some global weightings are:

- $g_i = \log_2 \frac{N - n_i}{n_i}$, where N is the number of documents in the collection and n_i is the total number of occurrences of term i in the collection (Croft and Harper [3]).
- $g_i = \log_2 N - \sum_{j=1}^{n} \frac{Freq_{ij}}{Freq_i} \log_2 \frac{Freq_i}{Freq_{ij}}$, where $Freq_{ij}$ is the frequency of term i in document j, $Freq_i$ is the frequency of term i in the collection and N is the number of documents in the collection (Lochbaum [14]).

If for example the first of the local weightings, is combined with the second of the global weightings an entropy weighting is obtained. For a more detailed description of different weightings and similarity measures, see the book by Frakes and Baeza-Yates [6] and the book by Kowalski [11].

1.3 The singular value decomposition (SVD)

Only a short description of the SVD of a matrix is given here. For more details see Golub and van Loan [7].

Theorem: If $A \in \mathcal{R}^{m \times n}$ then there exist matrices

$$U = [u_1 \dots u_p] \in \mathcal{R}^{m \times p}$$

and

$$V = [v_1 \dots v_p] \in \mathcal{R}^{n \times p}$$

with orthogonal columns, $U^T U = I$, $V^T V = I$, and such that

$$U^T A V = \mathrm{diag}(\sigma_1 \dots \sigma_p) \in \mathcal{R}^{p \times p} \quad p = \min(m, n)$$

where $\sigma_1 \geq \sigma_2 \geq, \dots, \geq \sigma_p \geq 0$. For a proof, see the textbook [7].

The σ_i are the singular values of A, and the vectors u_i and v_i are the i:th left and the i:th right singular vector respectively.
Example: Let

$$A = \begin{bmatrix} 2.88 & -1.16 \\ 2.84 & -2.88 \end{bmatrix}.$$

Then

$$A = U\Sigma V^T = \begin{bmatrix} 0.6 & -0.8 \\ 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \\ -0.6 & -0.8 \end{bmatrix}^T .$$

The singular values of A are the non-negative square roots of the eigenvalues of A^TA, and the columns of U and V are orthonormal eigenvectors of AA^T and A^TA respectively.

The SVD reveals information about the global structure of the matrix. Let the SVD of A be given and $\sigma_1 \geq \ldots \sigma_r \ > \sigma_{r+1} = \cdots = \sigma_p = 0$. Let $\mathcal{N}(\mathcal{A})$ and $\mathcal{R}(\mathcal{A})$ denote the nullspace and the range of A respectively. Then

- $\text{rank}(A) = r$.
- $\mathcal{N}(A) = \text{span}\{v_{r+1} \ldots v_n\}$.
- $\mathcal{R}(A) = \text{span}\{u_1 \ldots u_r\}$.
- $A = \sum_{i=1}^r \sigma_i u_i v_i^T = U_r \Sigma_{rr} V_r^T$ where $U_r = [u_1 \ldots u_r]$, $V_r = [v_1 \ldots v_r]$ and $\Sigma_{rr} = \text{diag}(\sigma_1 \ldots \sigma_r)$.

Thus the SVD of a matrix can be used to determine its rank. More importantly, it can be used to compute a low-rank approximation A_k of A. Let the SVD of $A \in \mathcal{R}^{m\times n}$ be given. If $k < r = \text{rank}(A)$ and $A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$ then A_k which is constructed from the k largest singular triplets of A, is the closest (in 2-norm) rank-k matrix to A.

$$\min_{\text{rank}(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}.$$

Example: A rank-1 approximation of the matrix

$$A = \begin{bmatrix} 2.88 & -1.16 \\ 2.84 & -2.88 \end{bmatrix}$$

from the previous example is given by keeping the largest singular value $\sigma_1 = 5$.

$$u_1\sigma_1 v_1^T = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \begin{bmatrix} 5 \end{bmatrix} \begin{bmatrix} 0.8 & -0.6 \end{bmatrix} = \begin{bmatrix} 2.4 & -1.8 \\ 3.2 & -2.4 \end{bmatrix} .$$

The *pseudoinverse* of a matrix A is a matrix A^+ of the same dimension as A^T such that $AA^+A = A$, $A^+AA^+ = A^+$, and AA^+ as well as A^+A are Hermitian.

Example: Let $A_{m\times n}$ have rank r and let $A = U\Sigma V^T$. Then the matrix $X = V\Sigma^+V^T$, where Σ^+ has the same dimension as Σ^T with the only non-zero entries $< \Sigma^+ >_{ii} = 1/\sigma_i$, $i = 1 \ldots r$, is a pseudoinverse of A.

2 Latent semantic indexing (LSI)

Latent semantic indexing (LSI) [5] is an extension of the vector space model for information retrieval. Instead of using the original term document matrix A when scoring documents, a reduced-rank representation A_k of A is used. Let

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

be the SVD of the $m \times (n-1)$ term document matrix[2] A. The matrix A_k is given by the best rank-k approximation $A_k = U_k \Sigma_{kk} V_k^T$ of A, where U_k and V_k are formed by the first k columns of U and V. The $k \times k$ diagonal matrix Σ_{kk} is formed by the first k rows and columns of Σ. The cosines of the angle between the query vector q and all the columns in A_k will be measured by

$$\mathrm{sim}(q, A_k e_j) = \frac{|q^T(A_k e_j)|}{\|q\|_2 \|A_k e_j\|_2}, \quad j = 1 \dots n-1.$$

All the documents where

$$\mathrm{sim}(q, A_k e_j) > s, \quad 0 \leq s \leq 1$$

will be returned to the user.

The effectiveness of LSI as measured by increased average precision, has been demonstrated for several text collections: Dumais [5] and [4], and Letch and Berry [13]. Experiments with small interactive databases have shown monotonic improvements in recall rate without overall loss of precision[5]. There are also text collections for which LSI is not significantly better than the original vector space model.

There is a tremendous diversity in the words people use to describe the same document. The rank k representation of A captures most of the important underlying structure in the term document matrix, yet at the same time removes the noise or variability in word usage. Minor differences in terminology will be ignored. Terms which occurs in similar documents will be near each other in the k-dimensional subspace spanned by the column vectors in U_k. The optimal value of the rank k is collection dependent. A too large value on k will reintroduce noise and local variability into the k-dimensional subspace. The benefits of LSI to the original vector space model

[2] To simplify notations in this section we have let the term document matrix be of size $m \times (n-1)$.

will be lost and performance will be reduced. A too small value of the rank k will capture too little of the underlying structure in A and information will be lost. (Moreover, in order to minimize memory requirements the rank k should be as small as possible). A suggestion for how to determine the rank k is given by Hongyuan in article [9]. In article [10] T Kolda and D O'Leary propose to replace the SVD with a semi-discrete decomposition (SDD) [10]. The SDD will require much less storage than the SVD.

2.1 Relevance feedback

When using LSI in the search for relevant documents, documents that give the largest scores in the search for angles will be returned. Among these documents some will be relevant and some will be irrelevant. The continued search may be greatly improved if the user returns scores for each returned document on which he/she has any opinion. These scores can be added as a partly specified extra row in the term document matrix. When the search engine continues to look at unseen document vectors, the returned scores are used to compute a better score of each new document, all the time showing the user those documents passing a threshold.

2.1.1 Vector space feedback

Let $\begin{bmatrix} q & A \end{bmatrix}_{m \times n}$, where q is the query vector and A is the $m \times (n-1)$ term document matrix, be the extended term document matrix. Assume that $p > 0$ documents have been seen (and scored) by the user and denote D by

$$D = \begin{bmatrix} A_p & A_{n-p} \\ a_p & a_{n-p} \end{bmatrix}$$

where A_p consists of the p columns of the extended term document matrix corresponding to the documents seen (and scored) by the user. The sub-matrix A_{n-p} consists of the $n-p$ columns of the extended term document matrix corresponding to the documents not yet seen by the user. The p-dimensional row vector a_p consists of the scoring of the document vectors in A_p. The entries in the remaining part of the added row, a_{n-p}, are unknown. It is reasonable to find an x that solves the least squares problem

$$\min \|x^T A_p - a_p\|_2. \tag{1}$$

By letting $s_{n-p} = x^T A_{n-p}$ an approximation s_{n-p} to a_{n-p} will be obtained. The algorithm becomes very simple:

Algorithm 2.1.

$A_p := q; \quad A_{n-p} := A$
$a_p = c$
do
$s_{n-p} := a_p A_p^+ A_{n-p}$

$\boxed{\textit{Output result to user}} \Rightarrow$

$\boxed{\textit{Input user's judgement}} \Leftarrow$

update a_p
$A_p := \begin{bmatrix} q & A(:, \text{seen documents}) \end{bmatrix}$
$A_{n-p} := A(:, \text{not yet seen documents})$
end

When starting, A_p will be equal to the query vector q given by the user. Thus q is considered to be a known and scored document vector with scoring c[3]. The submatrix A_{n-p} will equal the term document matrix. The pseudoinverse of A_p is denoted A_p^+. In our test runs we score the documents according to the entries in $|s_{n-p}|$, although other similarity measures will also be possible. When the user has seen (and scored) documents, the known part of the last row in D, a_p, increases in size and thus has to be updated. In the same way A_p will increase and A_{n-p} will decrease in column size. The unknown part of D becomes smaller.

If $c = \|q\|_2$ the first estimation s_{n-p} of a_{n-p} will simply be the dot product between the vector q and the document vectors in A. So if all the columns in A_p and A_{n-p} are normalized (using 2-norm), the first estimation of a_{n-p} will be equal to the cosine of the angles between q and all the document vectors in A.

Let d be the last row in D and let

$$d = \begin{bmatrix} a_p & s_{n-p} \end{bmatrix}$$

[3]The algorithm performs better if the query vector is included in the matrix A_p during the whole session. We have therefore chosen to present this algorithm with the query vector included in the whole section.

where a_p is the scoring computed in previous iterations and s_{n-p} is the approximation of a_{n-p} computed in this iteration. Let $\hat{s}_{n-p}$ be the scoring of the best scored documents (those documents that are to be shown to the user) in s_{n-p}. When the documents corresponding to the entries in $\hat{s}_{n-p}$ are shown to the user, the known part of the last row of D can be updated based on the user's judgement. The updated a_p will be

$$a'_p = \begin{bmatrix} a_p & \text{user's scoring} \end{bmatrix}$$

and a new scoring of yet unseen documents may be computed.

If the user accepts the computed scoring $\hat{s}_{n-p}$ (i.e. if no relevance feedback is given), the algorithm stops rescoring the documents. We then have

$$a'_p = \begin{bmatrix} a_p & \hat{s}_{n-p} \end{bmatrix} = a_p A_p^+ A_{p+r}$$

where $p+r$ equals the indices of the documents corresponding to the entries in a_p and s_{n-p}. The last row d of D is given by

$$d = a_p A_p^+ \begin{bmatrix} q & A \end{bmatrix}.$$

When all the documents corresponding to $\hat{s}_{n-p}$ have been shown to the user A_p will be updated to $A'_{p+r} = A_{p+r}$. The next estimation d' of d will be

$$d' = a'_p (A'_{p+r})^+ \begin{bmatrix} q & A \end{bmatrix} = a_p A_p^+ A_{p+r} A_{p+r}^+ \begin{bmatrix} q & A \end{bmatrix} = a_p A_p^+ \begin{bmatrix} q & A \end{bmatrix} = d$$

Since $x^T = a_p A_p^+$ is the solution to

$$\min \|x^T A_{p+r} - a_p A_p^+ A_{p+r}\|_2$$

the next to last equality follows. So if no relevance feedback is given, the scoring of the documents will remain the same.

Since we will have to solve (1) each time an answer is required, it may be necessary to keep A_p factorized in some way (see section 2.2). Let

$$A_p = QR$$

where the columns in $Q_{m\times p}$ are orthonormal and $R_{p\times p}$ is upper triangular.

In some of our test runs we have employed the user's scoring to modify R to become $\hat{R}$ and then used $\hat{A}_p = Q\hat{R}$ instead of A_p when solving the least squares problem (1).

2.1.2 LSI feedback

We may of course use the best (in 2-norm) rank-k approximation A_k instead of A when scoring documents in the algorithm described above. Let the singular value decomposition of the term document matrix $A = U\Sigma V^T$ and let $A_k = U_k \Sigma_k V_k^T$ be the best rank-k approximation of A. Assume $p > 0$ documents have been seen and scored by the user. We have

$$A_p = \begin{bmatrix} q & U_k \Sigma_k V_1^T \end{bmatrix}$$

where V_1 stands for the rows of V_k that correspond to a_p in d, and

$$A_{n-p} = U_k \Sigma_k V_2^T$$

where V_2 stands for the rows of V_k corresponding to a_{n-p} in d.

The choice of the rank k is important. As in the LSI case, a too large or too small value of k will reduce performance.

We could also reason slightly differently, obtaining a slightly different algorithm. Assume $p > 0$ documents have been scored by the user. It is reasonable to find a corresponding row u^T of U_k that generates the observed a_p as closely as possible.

$$a_p \approx u^T \Sigma_k V_1^T \rightarrow u^T \Sigma = a_p (V_1^T)^+$$

Now use the estimate of $u^T\Sigma$ to estimate the unspecified part a_{n-p}.

$$a_{n-p} \approx u^T \Sigma_k V_2^T \rightarrow s_{n-p} = a_p (V_1^T)^+ V_2^T$$

where s_{n-p} stands for the estimated value of a_{n-p}. This will correspond to solving a weighted least squares problem each time. If $p < k$ the system is underdetermined and if $p > k$ the system is overdetermined. If we want the query vector to be represented in the V-matrix we will have to update the singular value decomposition of A to include the query vector for every new query entered to the system. This can be done quite nicely using an algorithm described by Hongyuan in [9]. Let

$$(I - U_k U_k^T) q = u_q r \tag{2}$$

where $r = \|(I - U_k U_k^T)q\|_2$ and $\|u_q\|_2 = 1$. Note that $\begin{bmatrix} U_k & u_q \end{bmatrix}$ is orthonormal. Assume $r \neq 0$; then it can be verified that

$$\begin{bmatrix} A_k & q \end{bmatrix} = \begin{bmatrix} U_k & u_q \end{bmatrix} \begin{bmatrix} \Sigma_k & U_k^T q \\ 0 & r \end{bmatrix} \begin{bmatrix} V_k^T & 0 \\ 0 & 1 \end{bmatrix}.$$

Now let the singular value decomposition of

$$\begin{bmatrix} \Sigma_k & U_k^T q \\ 0 & r \end{bmatrix} = \begin{bmatrix} P_k & p_P \end{bmatrix} \begin{bmatrix} \hat{\Sigma}_k & 0 \\ 0 & \sigma_{k+1} \end{bmatrix} \begin{bmatrix} Q_k & q_Q \end{bmatrix}.$$

where P_k and Q_k has k columns and $\hat{\Sigma}_k \in \mathcal{R}^{k \times k}$. The $(k+1)$:th left and right singular vectors are denoted p_P and q_Q respectively. The best rank-k approximation of $\begin{bmatrix} \Sigma_k & U_k^T q \\ 0 & r \end{bmatrix}$ is given by

$$(\begin{bmatrix} U_k & u_q \end{bmatrix} P_k)\hat{\Sigma}_k(\begin{bmatrix} V_k & 0 \\ 0 & 1 \end{bmatrix} Q_k)^T.$$

2.2 Performance

The performance of the relevance feedback method will be discussed in terms of retrieval efficiency, execution efficiency and storage efficiency.

- Retrieval efficiency: As discussed earlier, if s_{n-p} is normalized properly the algorithm will not do worse than LSI in the full-rank case and in the first version of the low-rank cases, since the first scoring computed will be equal to the LSI scoring. In the second version of the low-rank case, the SVD constructed is not the exact SVD of $\begin{bmatrix} q & A_k \end{bmatrix}$. It is not even a good approximation to it, when the query vector added alters the original low-rank representation A_k significantly (i.e. when r in the relation(2) is small). See article [9] for details. So the relationship with the LSI-method may not be obvious in this case. Retrieval efficiency measured by precision and recall is given in Appendix A.

- Execution efficiency: When computing the scoring s_{n-p} in the search for relevant documents, a linear least squares problem (1) has to be solved. Assume p documents have been issued for feedback. If we keep A_p QR-factorized, that is,

$$A_p = Q_p R_p$$

where R_p is $p \times p$ and upper triangular and Q_p is $m \times p$ and has orthogonal columns, then the solution to the least squares problem (1) becomes

$$x^T = a_p A_p^+ = a_p R_p^{-1} Q_p^T = a_p R_p^{-1} R_p^{-T} A_p^T.$$

As more documents become known, the QR-factorization can be updated using some updating algorithm, see Golub and van Loan [7]. Also note that, in the full-rank case, the sparsity of A can be taken advantage of in the vector-matrix multiplications.

If more terms or documents must be added to the term document matrix (or if terms or documents must be deleted), this is done by adding (or deleting) columns or rows of A. In the full-rank case this will not be a problem, but in the LSI-feedback algorithm (described in section 2.1.2) the SVD of A may have to be updated in some way. Two alternatives currently exist: recomputing the SVD of a new term document matrix, or updating the existing SVD of A to include the added (or deleted) columns or rows. Methods for updating the SVD are discussed by Berry et. al. in article [2] and by Hongyuan in article [9].

- Storage efficiency: In the full rank case, we need to store the term document matrix A and in the low rank cases we will need to store the U_k, V_k and Σ_k in the singular value decomposition.

3 Krylov subspace methods for information retrieval

3.1 Introduction

When using LSI the best (in 2-norm) rank-k approximation A_k of the term document matrix $A \in \mathcal{R}^{m \times n}$ is computed. The cosines of the angles between the query vector q and the document vectors in A_k were computed and the best scored documents were returned to the user. We are going to describe a new method where we will use bidiagonalization derived from the Hermitian Lanczos algorithm to obtain several low-rank approximations to the term document matrix. The best scored documents in each case will be returned. The low-rank approximations are specific for each query vector q and will be denoted $A_\kappa(q)$. We will use the query vector q (or it's scores $A^T q$) as starting vector when applying the bidiagonalization scheme. We will in general keep the rank κ in $A_\kappa(q)$ low[4]. The best scored documents in each case will be returned.

In order to simplify computations and thus keep waiting times down for the user, we have used a slightly different similarity measure.

Moreover the performance will increase if relevance feedback is allowed. When relevance feedback is employed the user's feedback is employed when restarting the bidiagonalizing method. In a second approach we let the user's feedback change the residual given by the bidiagonalizing method.

A description of the bidiagonalization methods used is given in section 3.2. The method presented will be described in section 3.3 and the similarity measure used with it will be discussed in section 3.4. Relevance feedback will be discussed in section 3.5. The performance of the algorithms is briefly discussed in section 3.6.

3.2 Two bidiagonalization algorithms

In our algorithm we will use two bidiagonalization schemes derived from the Hermitian Lanczos method. The schemes were described in an article by Paige and Saunders [15]. A very short discussion of the Hermitian Lanczos algorithm, and the two bidiagonalizing schemes will follow in this section.

[4]In LSI the rank k usually has values between $100 - 500$ [9]. In our algorithm the rank κ usually will have values between 10-30.

Let v be any nonzero $m \times 1$ vector and let $M \in \mathcal{R}^{m \times m}$. A Krylov subspace algorithm extracts approximations from a subspace of the form

$$\mathcal{K} = \{v, Mv, M^2 v, ...\} \tag{3}$$

referred to as a Krylov subspace. Well-known Krylov subspace methods are the Hermitian Lanczos algorithm, the non-Hermitian Lanczos algorithm and the Arnoldi algorithm. These methods are based on projection methods onto Krylov subspaces. A very nice description of Krylov subspace methods is given in the textbook by Saad [16]; see also the textbook written by Bay [1]. Lanczos describes his method in the article [12].

Given a symmetric matrix M and a starting vector v the Hermitian Lanczos method generates a sequence of vectors and scalars such that the matrix is reduced to tridiagonal form. The algorithm successively orthogonalizes the vectors spanning a Krylov subspace (3). It can be shown that the successive orthogonal vectors satisfy a three-term recurrence, so that each new vector needs to be orthogonalized only against the previous two. It means that the matrix $T = Q^T M Q$, where the columns in Q are the successively orthogonalized vectors, is tridiagonal. At the end when $Q \in \mathcal{R}^{m \times m}$ then the relation

$$MQ = QT$$

holds.

Let $A \in \mathcal{R}^{m \times n}$ and $q \in \mathcal{R}^{m \times 1}$. If we apply the Hermitian Lanczos algorithm to the symmetric matrix

$$M = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$

with starting vector $\begin{bmatrix} q^T & 0 \end{bmatrix}^T$, then the tridiagonal matrix obtained will have the form

$$T = \begin{bmatrix} 0 & \alpha_1 & & & \\ \alpha_1 & 0 & \beta_2 & & \\ & \beta_2 & \ddots & \ddots & \\ & & \ddots & & \beta_{k+1} \\ & & & \beta_{k+1} & 0 \end{bmatrix},$$

where α_i and β_i are defined as in (3.1). The u_i:s and v_i:s (defined in (3.1)) will occur in a certain pattern in the Q matrix, and the relations (9), (10), (11) below will follow immediately.

Now consider the augmented least squares problem

$$\min \| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} q \\ 0 \end{bmatrix} \|_2 \tag{4}$$

where $A_{m\times n}$ and $q_{m\times 1}$ are given and λ is an arbitrary scalar used to regularize the solution. Let the residual vector $r = q - Ax$. The solution of the least squares problem (4) satisfies the quadratic system

$$\begin{bmatrix} I & A \\ A^T & -\lambda^2 I \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} q \\ 0 \end{bmatrix}. \tag{5}$$

When using the Lanczos tridiagonalizing scheme with the matrix from (5) and the vector $\begin{bmatrix} q^T & 0 \end{bmatrix}^T$ as starting vector, it also reduces to the scheme (3.1) defined below [15]. The tridiagonal matrix obtained can be permuted (after $2k+1$ iterations) to the form

$$T = \begin{bmatrix} I & B_k \\ B_k^T & -\lambda^2 I \end{bmatrix},$$

where B_k is defined in (8). The u_i:s and v_i:s will again occur in a certain pattern, and the relations (9) (10) (11) will follow.

Algorithm 3.1.
Reduction to lower bidiagonal form:

$\beta_1 u_1 := q; \qquad \alpha_1 v_1 := A^T u_1$
for $i = 1, 2 \ldots$
$\quad \beta_{i+1} u_{i+1} := A v_i - \alpha_i u_i$
$\quad \alpha_{i+1} v_{i+1} := A^T u_{i+1} - \beta_{i+1} v_i$
end

The scalars α_i and β_i are chosen to normalize the corresponding vectors[5].

---

[5] $\alpha_1 v_1 := A^T u_1$ for example implies the computations $\tilde{v}_1 := A^T u_1$; $\alpha_1 := \|\tilde{v}_1\|_2$; $v_1 := \frac{\tilde{v}_1}{\alpha_1}$.

Define

$$U_k = \begin{bmatrix} u_1 & u_2 & \dots & u_k \end{bmatrix} \quad V_k = \begin{bmatrix} v_1 & v_2 & \dots & v_k \end{bmatrix} \tag{6}$$
$$\tag{7}$$

and

$$B_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}. \tag{8}$$

If we had no rounding errors we would have $U_{k+1}^T U_{k+1} = I$ and $V_k^T V_k = I$. After k iterations we will have

$$\begin{aligned} U_{k+1}(\beta_1 e_1) &= q && (9) \\ AV_k &= U_{k+1} B_k && (10) \\ A^T U_{k+1} &= V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T && (11) \end{aligned}$$

and the solution x_k to the least squares problem (4) is given by

$$x_k = V_k y_k$$

where y_k is the solution to

$$\min \| \begin{bmatrix} B_k \\ \lambda I \end{bmatrix} y_k - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \|_2$$

If $\beta_{i+1} = 0$ then it is not hard to verify that

$$\begin{aligned} AV_i &= U_i B_{ii} \\ A^T U_i &= V_i B_{ii}^T \end{aligned}$$

where

$$B_{ii} = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_i & \alpha_i \end{bmatrix}.$$

Thus $\sigma(B_{ii}) \subset \sigma(A)$ [6]. If $\sigma(B_{ii})$ are equal to the largest i singular values of A then $U_i B_{ii} V_i^T$ is the best (in 2-norm) rank-i approximation of A . In fact (3.1) can be used when computing a singular value decomposition of A.

When using the Lanczos tridiagonalizing scheme with the matrix from (5) and the vector $\begin{bmatrix} 0 & -q^T A \end{bmatrix}^T$ as starting vector, it reduces to the scheme bellow.

Algorithm 3.2.
Reduction to upper bidiagonal form:

$\theta_1 v_1 := A^T q;\ \rho_1 p_1 := A v_1$
for $i = 1, 2 \ldots$
  $\theta_{i+1} v_{i+1} := A^T p_i - \rho_i v_i$
  $\rho_{i+1} p_{i+1} := A v_{i+1} - \theta_{i+1} p_i$
end

Let

$$P_k = \begin{bmatrix} p_1 \ldots p_k \end{bmatrix}, \tag{12}$$
$$V_k = \begin{bmatrix} v_1 \ldots v_k \end{bmatrix} \text{ and} \tag{13}$$
$$R_{kk} = \text{bidiag } (\rho_i, \theta_{i+1}). \tag{14}$$

After k iterations we will have

$$V_k(\theta_1 e_1) = A^T q$$

$$A V_k = P_k R_{kk}$$

$$A^T P_k = V_k R_{kk}^T + \theta_{k+1} v_{k+1} e_k^T.$$

The solution x_k to (4) is given by

$$x_k = V_k y_k$$

where y_k satisfies the system

$$(R_{kk}^T R_{kk} + \lambda^2 I) y_k = \theta_1 e_1. \tag{15}$$

Either scheme may be derived from the other by choosing the appropriate starting vector and interchanging A and A^T.

[6] $\sigma(M)$ means the singular values of the matrix M and $S_1 \subset S_2$ means that the set S_1 is a subset of the set S_2.

3.3 The algorithm

The algorithm is quite straight forward. Let $A_{m\times n}$ be the term document matrix and let q be the query vector. First iterate t steps using the bidiagonalizing scheme (3.1) described in section 3.2 with q as starting vector. Compute $\hat{q}$ (from section 3.4) and score the documents using the similarity measure (18). Return the best scored documents to the user. Then iterate another t steps with (3.1) and again score the documents. Present the best scored documents to the user.

Algorithm 3.3.

$\beta_1 u_1 := q;\ \alpha_1 v_1 := A^T u_1$
$\kappa := 0; \quad i = 1;$
$t :=$ number of bidiagonalization iterations
do while $\beta_i > 0$

(* Bidiagonalize *)
do while $i \geq \kappa + t$ **and** $\beta_i > 0$
$\beta_{i+1} u_{i+1} := A v_i - \alpha_i u_i$
$\alpha_{i+1} v_{i+1} := A^T u_{i+1} - \beta_{i+1} v_i$
$i := i + 1$
end

(* Compute $\hat{q}$ *)
$\kappa := \kappa + t;$
$\hat{q} := U_{\kappa+1} B_\kappa y_\kappa$

(* Score the documents *)
$$s_j^c := \frac{\hat{q}^T (A e_j)}{\|A e_j\|_2} \qquad j = 1 \ldots n$$

$\boxed{\textit{Output result to user}} \Rightarrow$

$i := \kappa$
end

The matrices $U_{\kappa+1}$ and B_κ are defined by (6) and (8), and y_κ is the solution to the least squares problem

$$\min \| \begin{bmatrix} B_\kappa \\ \lambda I \end{bmatrix} y_\kappa - \begin{bmatrix} \beta_1 e_1 \\ 0 \end{bmatrix} \|_2 \tag{16}$$

(see section 3.2). After iterating κ steps with the bidiagonalization scheme, a few singular values of B_κ may be equal to some of the singular values in A. These singular values will be referred to as converged singular values. The system of normal equations to the least squares problem (16) is given by

$$B_\kappa^T B_\kappa + \lambda^2 I = B_\kappa^T(\beta_1 e_1).$$

By choosing an appropriate value of λ unwanted converged singular values can be suppressed. It is also possible to simply omit unwanted singular values in B_κ when computing $\hat{q}$. A tricky issue is, of course, to decide which singular values are the wanted and which are the unwanted. If we want an LSI-like performance we will have to omit the converged small singular values and the non converged singular values (or at least suppress them). In general the large singular values of B_κ tend to be good approximations to the large singular values in A.

If $\beta_i = 0$ then $\sigma(B_{ii}) \subset \sigma(A)$, where the matrix B_{ii} is given by the first i rows of B_i. So if the singular values of B_{ii} are equal to the largest $\sigma(A)$, then $U_{i+1}B_{ii}V_i^T$ is the best (in 2-norm) rank-i approximation of A; and if the similarity measure in (17) is used, the scoring will be equal to the scoring given by LSI. If we want to continue we will have to restart the algorithm with another starting vector. The algorithm may of course be stopped before $\beta_i = 0$.

3.4 The similarity measure

In LSI the best (in 2-norm) rank-k approximation A_k of the term document matrix $A_{m\times n}$ was computed. The cosines of the angles between the query vector q and the document vectors in A_k,

$$\frac{q^T(A_k e_j)}{\|q\|_2 \|A_k e_j\|_2}, \qquad j = 1 \dots n, \tag{17}$$

were computed when scoring documents. We have used a slightly different similarity measure. Let

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

be the singular value decomposition of A, and thus the best rank k approximation $A_k = U_k \Sigma_k V_k^T$. We have

$$q^T A_k = q^T U_k \Sigma_k V_k^T = q^T U_k U^T U \Sigma V^T = q^T U_k U_k^T U \Sigma V^T = \hat{q}^T A$$

where $\hat{q} = U_k U_k^T q$. The cosines of the angles between the vector $\hat{q}$ and the document vectors in A are given by

$$\frac{\hat{q}^T (A e_j)}{\|\hat{q}\|_2 \|A e_j\|_2}, \qquad j = 1 \ldots n.$$

Since $\|\hat{q}\|_2$ will be constant for $j = 1 \ldots n$ it will be omitted. The similarity measure we use will be:

$$s_j^c = \frac{\hat{q}^T (A e_j)}{\|A e_j\|_2}, \qquad j = 1 \ldots n. \tag{18}$$

The similarity measure (18) will give values less than or equal to the similarity measure (17). We have since $\|A_k e_j\| \leq A e_j\|_2$

$$\frac{\hat{q}^T (A e_j)}{\|A e_j\|_2} \leq \frac{\hat{q}^T (A e_j)}{\|U_k^T A e_j\|_2} = \frac{q^T (A_k e_j)}{\|A_k e_j\|_2}.$$

Thus

$$c \frac{\hat{q}^T (A e_j)\|_2}{\|A e_j\|_2} \leq \frac{q^T (A_k e_j)}{\|q\|_2 \|A_k e_j\|_2}, \qquad j = 1 \ldots n$$

where $c = \dfrac{1}{\|q\|_2}$ is constant for $j = 1 \ldots n$.

3.5 Relevance feedback

We have tried two different approaches when adding relevance feedback to the algorithm given in section 3.3. In the first approach we simply do several restarts of the bidiagonalizing procedure with the user's feedback to decide the starting vector of the next iterations. In the second approach we employ the user's scorings to repeatedly change the residuals given by the bidiagonalization scheme. The first approach is discussed in section 3.5.1 and the second approach is discussed in section 3.5.2.

When relevance feedback is employed, we have sometimes used algorithm (3.2) instead of (3.1) when bidiagonalizing the matrix. This is because the user's scoring will have a direct relation to the scoring $A^T q$, so we have chosen to start the bidiagonalization procedure with the vector $A^T q$ instead of the vector q.

3.5.1 Relevance feedback using restarts

Let the computed scoring be s^c and denote the transformed scoring with

$$\ell \otimes s^c$$

where $\ell_j = \|A^T e_j\|_2$, $j = 1 \ldots n$, and $\otimes$ denotes element-wise multiplication [7]. Let the user's scoring be

$$s^u(j) = \begin{cases} c_j & \text{if } j \in D \text{ and user found document } j \text{ to be relevant} \\ 0 & \text{if } j \in D \text{ and user found document } j \text{ to be irrelevant} \\ s^c(j) & \text{if } j \notin D \end{cases} \tag{19}$$

where $D = \{ \ j \ : \ document_j$ has been shown to the user$\}$, c_j is a scalar (the scoring) chosen by the user and $j = 1 \ldots n$. When restarting the bidiagonalization scheme, we will employ the transformed user's scoring $\ell \otimes s^u$ as starting vector. The algorithm in its entirety will be:

[7] The transformed scoring can be expressed as a linear combination of the v:vectors generated by (3.1) (the computed scoring s^c cannot in general be expressed as a linear combination of the v:vectors).

Algorithm 3.4.

$\theta_1 v_1 := A^T q; \quad \rho_1 p_1 := A v_1$
$i := 1$
do while $\theta_i > 0$

(* Bidiagonalize *)
for $i = 1 \ldots \kappa$
$\theta_{i+1} v_{i+1} := A^T p_i - \rho_i v_i$
$\rho_{i+1} p_{i+1} := A v_{i+1} - \theta_{i+1} p_i$
if $\theta_i = 0$
$\kappa := i$ *; exit*
end
end

(* Compute $\hat{q}$ *)
$\hat{q} := P_{\kappa\kappa} R_\kappa y_\kappa$

(* Score the documents *)
$s_j^c := \dfrac{\hat{q}^T (A e_j)}{\|A e_j\|_2}$

$\boxed{\textit{Output result to user}} \Rightarrow$

$\boxed{\textit{Input user's scoring}} \Leftarrow$

(* Prepare for restart *)
$\theta_1 v_1 := \ell \otimes s^u$
$\rho_1 p_1 := A v_1$
end

P_κ and $R_{\kappa\kappa}$ are defined by (12) and y_κ is the solution to the system of linear equations (15). By choosing an appropriate value of λ, unwanted converged singular values may be suppressed (see section 3.3). If the value of θ_i in the inner loop becomes 0, this loop will have to be terminated with a lower value of the rank κ.

Various versions of the algorithm are possible. Before restarting, we could orthogonalize $\ell \otimes s^u$ against all the columns in V_κ and thus let the new starting vector become

$$\theta_1' v_1' = \ell \otimes s^u - V_\kappa V_\kappa^T (\ell \otimes s^u).$$

The transformed user's scoring could also be orthogonalized against parts of the columns in V_κ. While bidiagonalizing we could also make all the new computed v':vectors orthogonal to the columns (or part of the columns) in V_κ from previous computations.

Instead of using the bidiagonalizing scheme (3.2)the scheme (3.1) could be used with starting vector $A^T q$ and the matrix A^T. Note that the least squares problem (4) now becomes underdetermined. The restarts are performed in the same way as described above; letting

$$\beta_1 u_1 = \ell \otimes s^u$$

and

$$\alpha_1 v_1 = A^T u_1.$$

We have also tried the Lanczos tridiagonalizing scheme with the matrix $A^T A$ and starting vector $A^T q$.

3.5.2 Relevance feedback by changing the residual

Note that the v-vectors in (3.1) form an orthonormal basis for the Krylov space spanned by the vectors

$$(A^T q), ((A^T A) A^T q), \ldots, ((A^T A)^p A^T q). \tag{20}$$

This relevance feedback approach is based upon the following somewhat trivial observation. If the j:th document and the query have terms in common, then the dot product between the j:th column in A and the query vector $(Ae_j)^T q \neq 0$. We will have the relation $(A^T A)(Ae_j)^T q \neq 0$ for documents that have terms in common with documents that have terms in common with q; thus the second vector among the vectors in (20) may also be used as a similarity measure. Theoretically all the vectors from the vectors in (20) could be used as similarity measures, although when κ becomes too large the vectors will not work very well as similarity measures.

After κ iterations with (3.1) we have

$$A^T U_{\kappa+1} = V_k B_\kappa^T + \alpha_{\kappa+1} v_{\kappa+1} e_{\kappa+1}^T \tag{21}$$

Denote $\alpha_{\kappa+1}v_{\kappa+1} = A^T u_{\kappa+1} - \beta_{\kappa+1}v_\kappa$, the residual. The vector $A^T u_{\kappa+1}$ is the $(\kappa+1)$:th vector from the vectors (20). Let the user's scoring be given by s^u in section 3.5.1, and let the new residual be given by

$$\alpha'_{\kappa+1}v'_{\kappa+1} = \ell \otimes s^u - \beta_{\kappa+1}v_\kappa$$

The user's scoring will be seen as a correction of the $(\kappa+1)$:th vector from the vectors in (20). Continue the bidiagonalizing scheme a few more times with this updated residual. The algorithm becomes

Algorithm 3.5.

$\beta_1 u_1 := q; \qquad \alpha_1 v_1 := A^T u_1$
$\kappa := 0; \qquad i = 1;$
$t :=$ number of bidiagonalization iterations
do while $\beta_i > 0$

(* Bidiagonalize *)
do while $i \geq \kappa + t$ **and** $\beta_i > 0$
$\beta_{i+1}u_{i+1} := Av_i - \alpha_i u_i$
$\alpha_{i+1}v_{i+1} := A^T u_{i+1} - \beta_{i+1}v_i$
$i := i + 1$
end

(* Compute $\hat{q}$ *)
$\kappa := \kappa + t;$
$\hat{q} := U_{\kappa+1}B_\kappa y_\kappa$

(* Score the documents *)
$$s_j^c := \frac{\hat{q}^T(Ae_j)}{\|Ae_j\|_2} \qquad j = 1 \ldots n$$

$\boxed{\textit{Output result to user}} \Rightarrow$

(* Update residual *)
$\alpha_{\kappa+1}v_{\kappa+1} := \ell \otimes s^u - \beta_{\kappa+1}v_\kappa$
$i := \kappa + 1$
end

If the transformed user's scoring s^u can be written as a linear combination of v_κ and $v_{\kappa+1}$ then the residual change will just correspond to changing the starting vector slightly. After κ iterations with (3.1) we have the relation (21). The residual will only affect the last column in $V_\kappa B_\kappa^T$. Denote the desired residual $\hat{\alpha}_{\kappa+1} z_{\kappa+1}$ and let

$$\hat{\alpha}_{\kappa+1} z_{\kappa+1} = b_\kappa v_\kappa + \alpha_{\kappa+1} v_{\kappa+1}$$

where b_κ is an arbitrarily chosen scalar. We will then have

$$V_\kappa B_\kappa^T + \hat{\alpha}_{\kappa+1} z_{\kappa+1} e_\kappa^T = V_\kappa B_\kappa^T + b_\kappa v_\kappa + \alpha_{\kappa+1} v_{\kappa+1} e_{\kappa+1}^T = V_\kappa (B_\kappa^T + b_\kappa v_\kappa e_\kappa^T) + \alpha_{\kappa+1} v_{\kappa+1} e_\kappa^T$$

Thus

$$A^T U_{\kappa+1} = V_\kappa (B_\kappa^T - b_\kappa v_\kappa e_\kappa^T) + \hat{\alpha}_{\kappa+1} z_{\kappa+1} e_\kappa^T.$$

If the transformed user's scoring cannot be expressed as a linear combination of v_κ and $v_{\kappa+1}$ (which is more likely to happen), we have tried two different approaches:

- Simply ignore this fact and update the residual $\alpha_{\kappa+1} v_{\kappa+1}$ to become

$$\alpha'_{\kappa+1} v'_{\kappa+1} = \ell \otimes s^u - \alpha_{k+1} v_\kappa. \tag{22}$$

- Partly ignore this fact. Assume the residual can be expressed as a linear equation of all the previously computed v-vectors. That is

$$\hat{\alpha}_{\kappa+1} z_{\kappa+1} = V_\kappa b_\kappa + \alpha_{\kappa+1} v_{\kappa+1}$$

  where (b_κ) is a $\kappa \times 1$ vector. Now the $(\kappa + 1)$ row in B_κ has to be changed in order to compensate for this new residual. In this way a larger part of the user's scoring may be employed.

3.6 Performance

The performance of the methods proposed will be discussed in terms of retrieval efficiency, execution efficiency and storage efficiency.

- Retrieval efficiency: As discussed earlier, if the non-converged and the small converged singular values are omitted when computing the vector $\hat{q}$ in algorithm 3.3, the algorithm will behave similarly to LSI used with very low rank. We use another similarity measure, though, so the two methods' similar output will depend on how well the norms of the columns in the low-rank approximation $A_\kappa(q)$ has converged to the values of the norms of the columns in A. Also note that, since we keep the rank very low, the starting vector in the bidiagonalization schemes still plays an important role when computing the scoring s^c.

Retrieval efficiency measured by precision and recall is given in Appendix A.

- Execution efficiency: We will in general keep the rank κ very low in our algorithms, which means that the bidiagonalization procedures will be quite rapid. When computing the scoring s^c a least squares problem (16) (or a system of linear equations (15)) has to be solved. Again, since we keep the rank κ very low these systems will be very small.

  If more terms or documents must be added to the term document matrix (or if terms or documents must be deleted), this is done simply by adding (or deleting) columns or rows of the term document matrix A.

- Storage efficiency: We must store the term document matrix A.

A Appendix. Test collections and numerical tests

The test collections we have used consists of a document database and a set of queries for the data base for which relevance judgments are available.

- The CRANFIELD collection consists of 1400 documents and 225 queries. The documents deals with Aerodynamics.
- The MEDLINE collection consists of 1033 documents and 30 queries. The documents deals with Medicine.

The Medline and the Cranfield test collections are old and quite small. Both the test collections can be received from `ftp://ftp.cs.cornell.edu/pub/smart`. We will also use larger test collections received from the Text Retrieval Conference (TREC).

- The TREC-4 disc contains of three data collections, the *Financial Times*, 1991-1994 (FT), the *Federal Register*, 1994 (FR94) and the *Congressional Record*, 1993 (CR). The FT collection, FR94 collection and the CR collection consists of 210 158, 55 630 and 27 922 documents respectively.

We have tested our algorithms on the Cranfield matrix and the Medline matrix. Tests on the databases on the TREC-4 disc is to be made in a near future. Plots for the Medline matrix will follow in this section, similar plots for the Cranfield matrix can be received upon question.

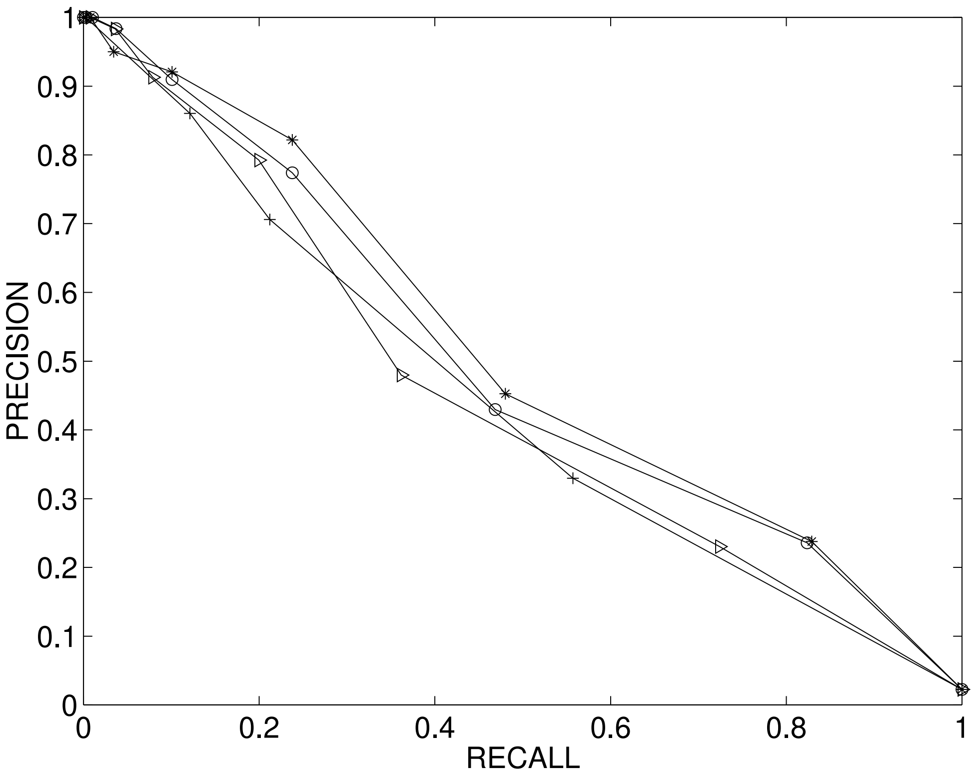


Figure 1: Figure 1 shows the recall/precision curves when LSI was used with different values of the rank k. The curves correspond to values between 20 and 122 of the rank.

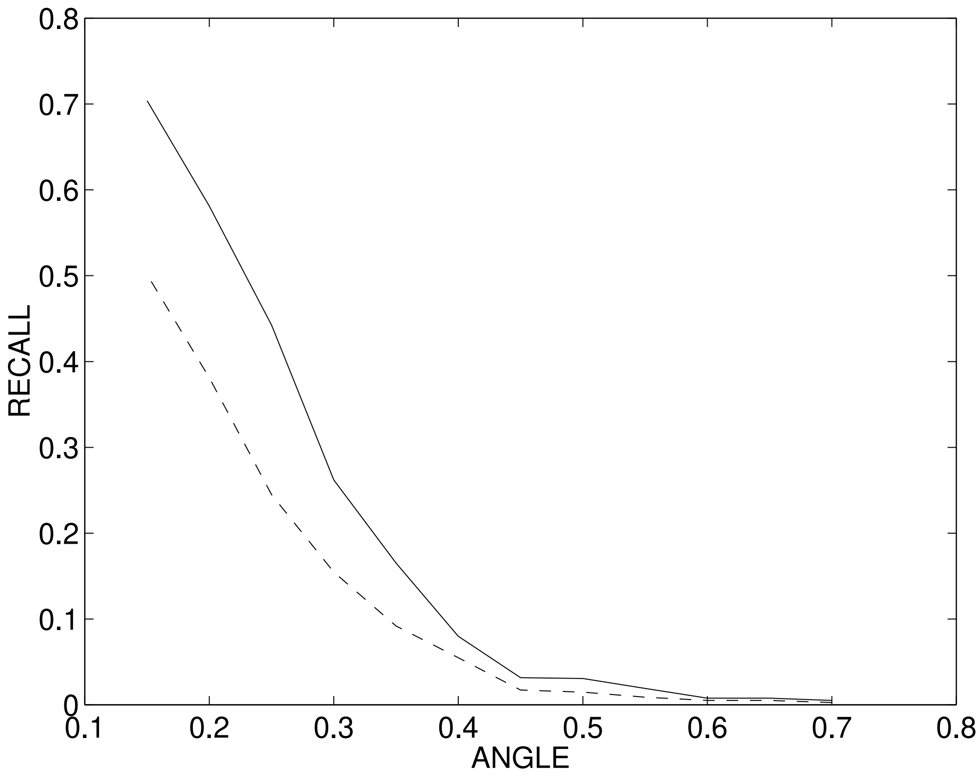


Figure 2: Figure 2 shows the recall curves for different values of the angle when algorithm 2.1 was used. The dashed line shows recall after one iteration and the solid line shows the recall after two iterations. Recall increases after relevance feedback. For this figure the full rank term document matrix was used, and the dashed line will correspond to the recall for the vector model.

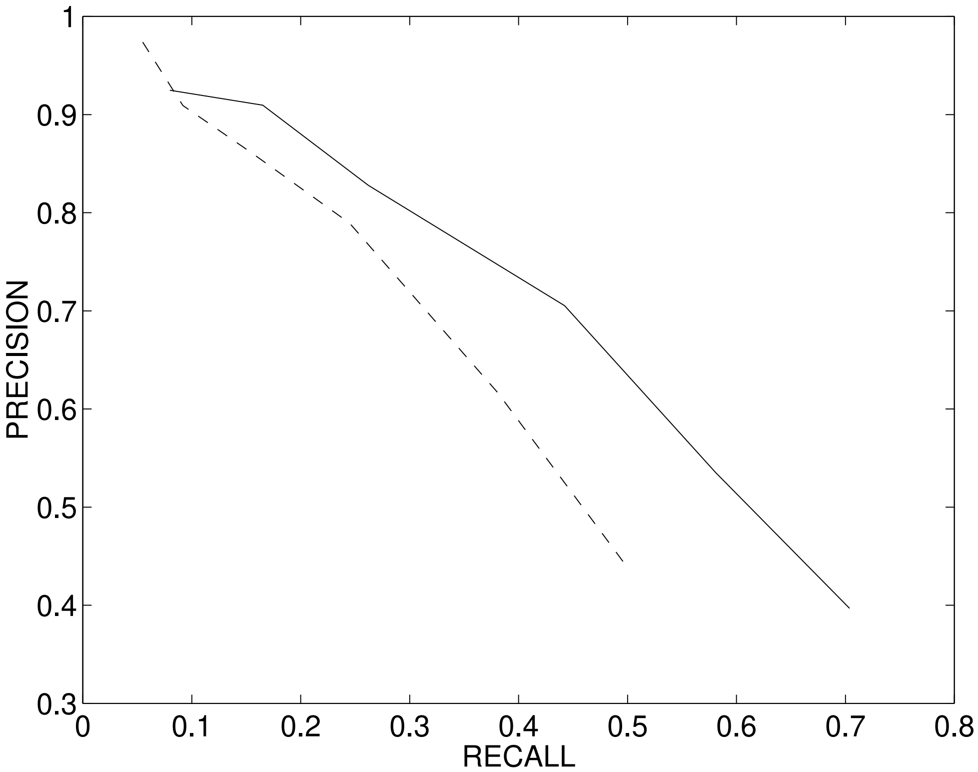


Figure 3: Figure 3 shows the recall/precision curves when algorithm 2.1 was used. The dashed line shows recall/precision after one iteration and the solid line shows the recall/precision after two iterations. For this figure the full rank term document matrix was used, and the dashed line will correspond to the recall/precision for the vector model. Both recall and precision increases after relevance feedback.

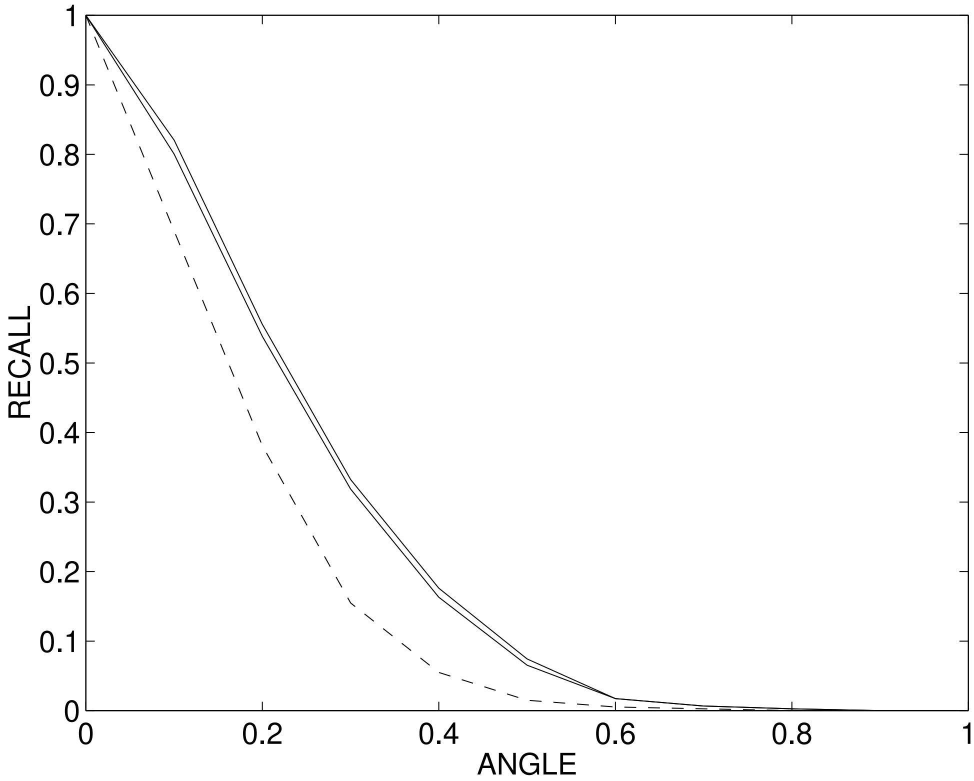


Figure 4: Figure 4 shows the recall curves for different values of the angle when algorithm 3.3 was used. The dashed line shows recall for the ordinary vector model and the solid lines shows recall after one and two iterations respectively. Recall increases after one iteration with the algorithm but remains the same for the second output. The rank κ that was used is 4 for the first output and 8 for the second.

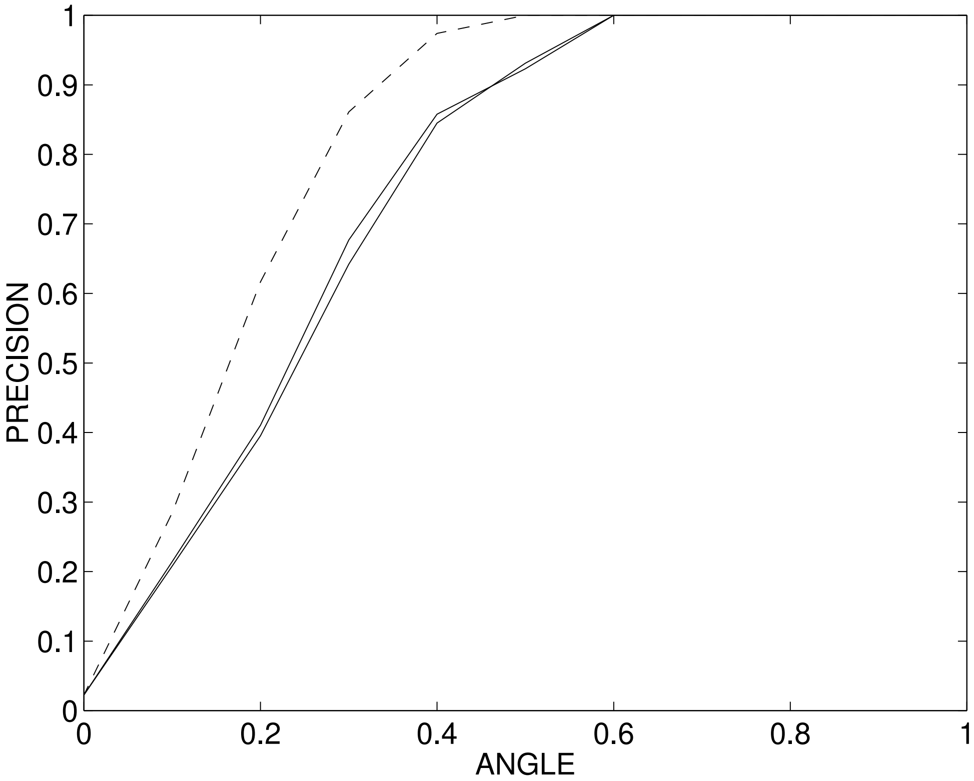


Figure 5: Figure 5 shows the precision curves for different values of the angle when algorithm 3.3 was used. The dashed line shows precision for the ordinary vector model and the solid lines shows recall after one and two iterations respectively. Precision decreases after one iteration with the algorithm but remains the same for the second output. The ranks used are the same as in figure 4.

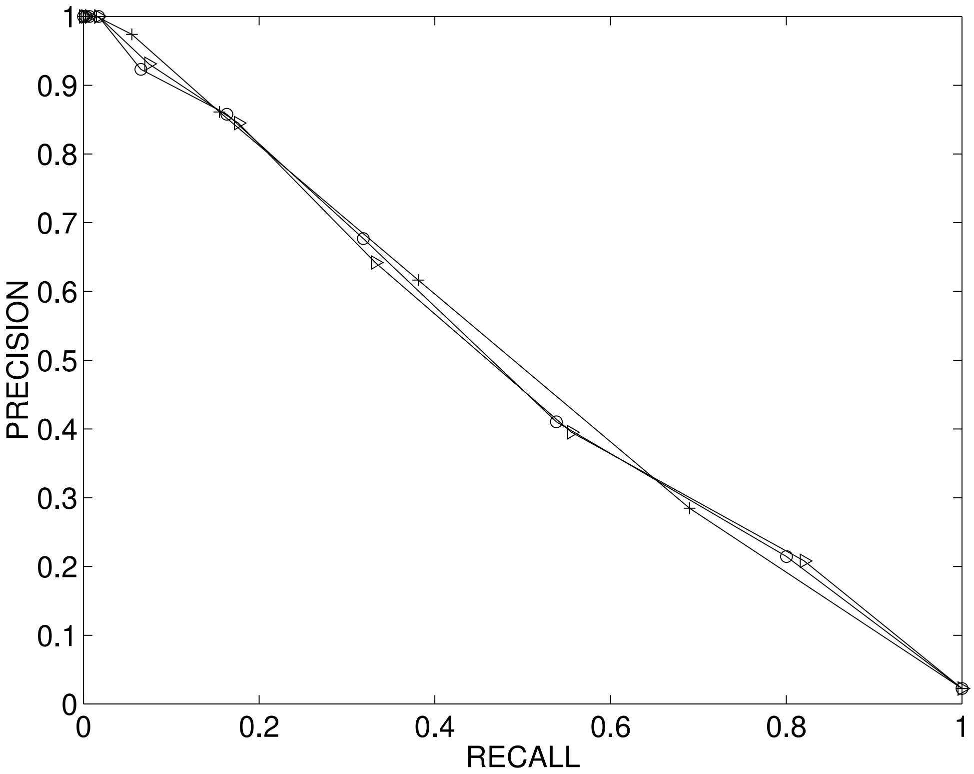


Figure 6: Figure 6 shows the recall/precision curves for figure 4 and 5. The "+"-marked curve shows recall/precision for the vector model and the other two shows recall/precision after one and two iterations respectively. Since recall increased and precision decreased the recall/precision curves will look similar.

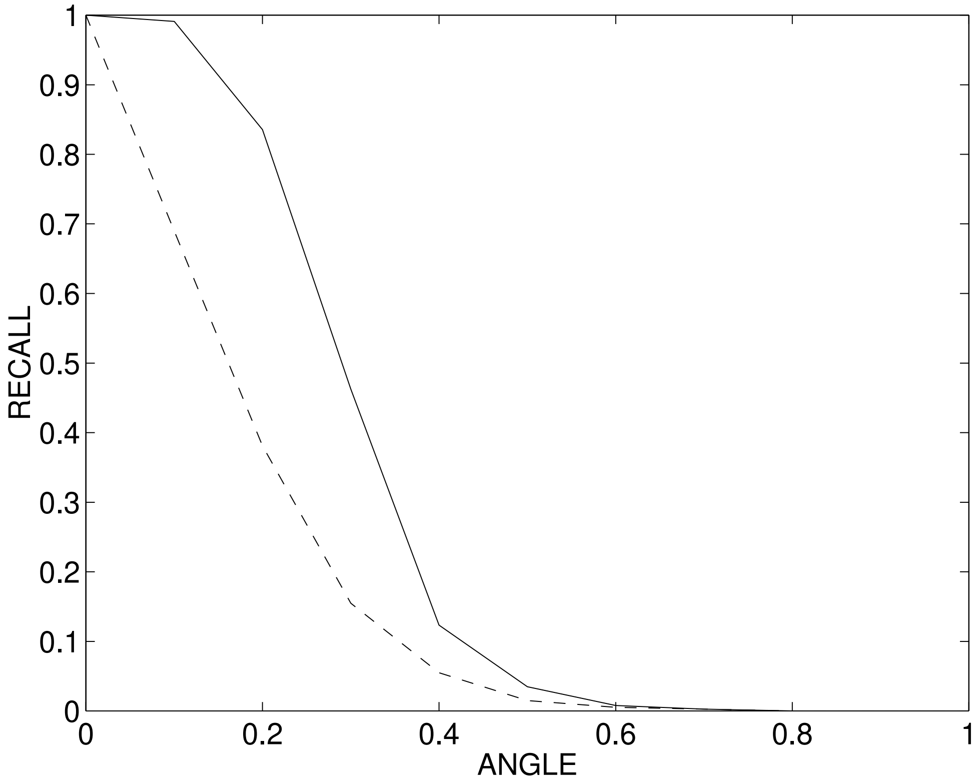


Figure 7: Figure 7 shows the recall curves for different values of the angle when algorithm 3.4 was used. The dashed line shows recall for the ordinary vector model and the solid line shows recall after one restart. The rank κ of the matrix was equal to 5 and the restarting vector was created as suggested in (19) section 3.5.1. Recall will increase (but precision will decrease).

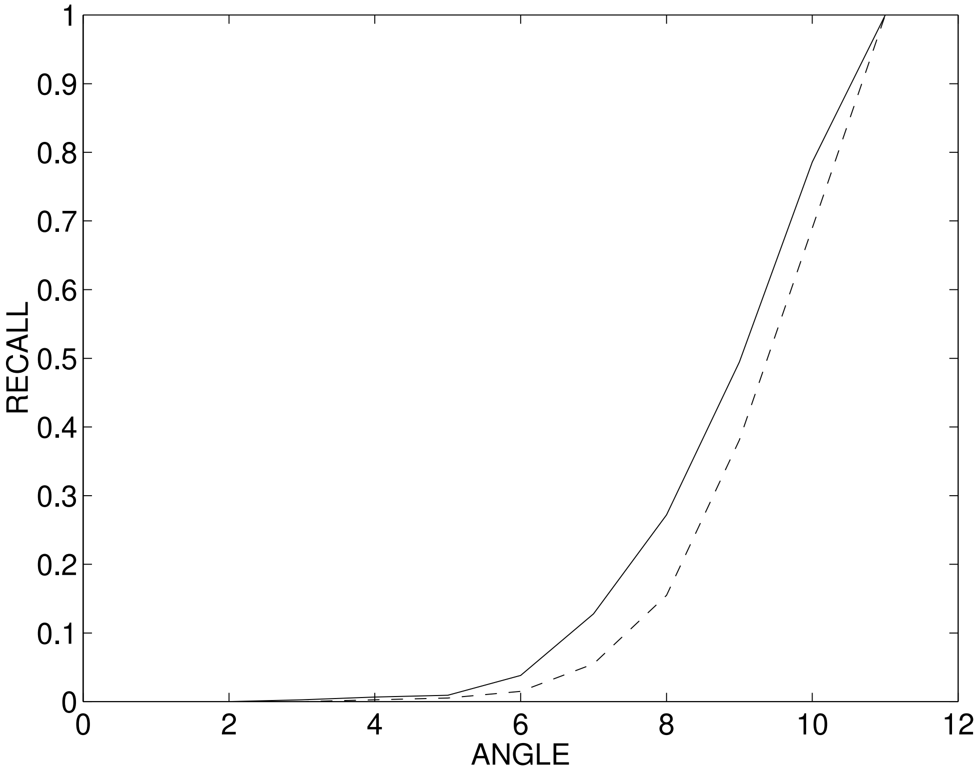


Figure 8: Figure 8 shows the recall curves for different values of the angle when algorithm 3.5 in section 3.5.2 was used. The dashed line shows recall for the ordinary vector model and the solid line shows recall after one residual change. The rank κ of the matrix was equal to 5 and the new residual was created assuming that the transformed users scoring could be expressed as a linear combination of the two previously computed v-vectors. Recall increases but precision decreases slightly

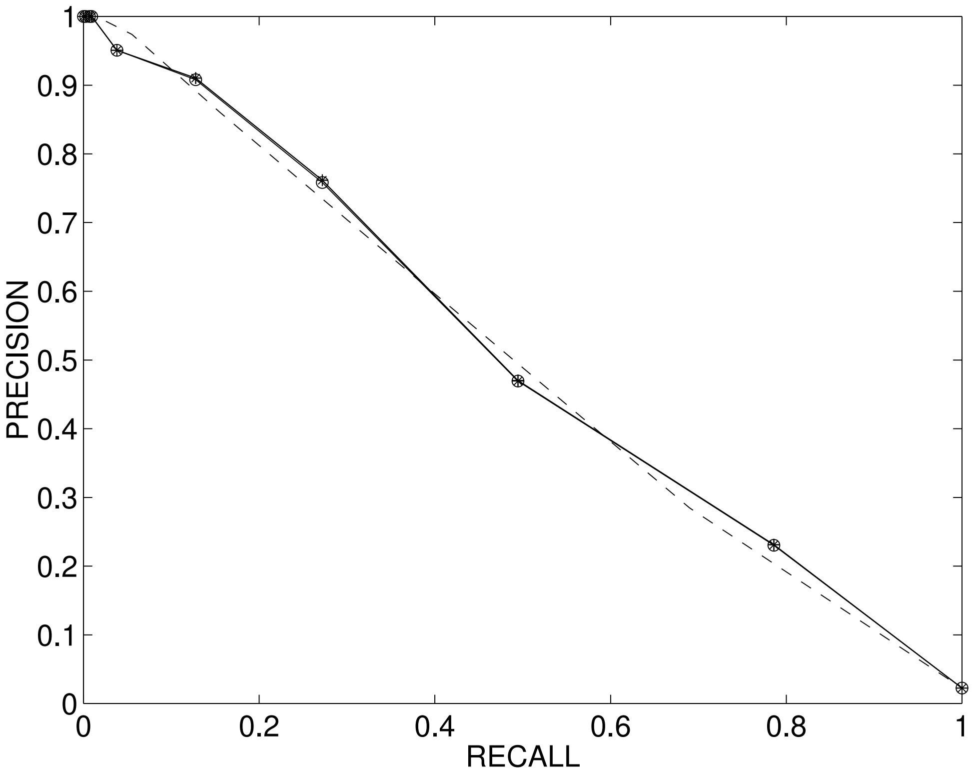


Figure 9: Figure 9 shows the recall/precision curves for figure 8. The dashed curve shows recall/precision for the vector model and the other two shows recall/precision after one and two iterations respectively. Since recall increased and precision decreased the recall/precision curves will look similar.

References

[1] Z. Bai, J. Demmel, J. Dongarra, A. Edelman, M. Gu, B. Kågström, A. Ruhe, Y. Saad, G. Sleijpen, D. Sorensen, and H. Van der Vorst, *Eigentemplates.* work in progress.

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

[3] W. B. Croft and D. J. Harper, *Using probabilistic models of document retrieval without relevance information*, Documentation, 35(4) (1979), pp. 285–95.

[4] S. T. Dumais, *Using LSI for information filtering: TREC-3 experiments*, in D K Harman Editor, Overview of TREC-3 Conference, NIST Special Publication, 1995, pp. 500–225, 219–230.

[5] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harsman, *Indexing by latent semantic analysis*, Journal of the American Society for Information Science, 41 (1990), pp. 391–407.

[6] W. B. Frakes and R. Baeza-Yates, *Information Retrieval, data structures and algorithms*, Prentice Hall, 1992.

[7] G. H. Golub and C. F. van Loan, *Matrix computations*, The Johns Hopkins University Press, Baltimore, Maryland, 2nd ed., 1989.

[8] D. Harman, *An experimental study of factors important in document ranking.* Paper presented at the ACM conference on research and development in information retrieval, Pisa Italy, 1986.

[9] Z. Hongyuan, *A subspace-based model for information retrieval with applications in latent semantic indexing*, in Lecture notes in computer science, Springer-Verlag, 1998, pp. 29–42. Vol. 1475.

[10] T. G. Kolda and D. P. O'Leary, *A semi-discrete decomposition for latent semantic indexing in information retrieval*, ACM-TOIS, (to appear).

[11] G. Kowalski, *Information Retrieval systems, theory and implementation*, Kluwer Academic Publishers, 1997.

[12] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, Journal of Research, Nat. Bur. of Standards, 45 (1950), pp. 255–282.

[13] T. A. Letche and M. W. Berry, *Large-scale information retrieval with latent semantic indexing*, Information Sciences - Applications, 100 (1997), pp. 105–137.

[14] K. E. Lochbaum and L. Steeter, *Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval*, Information Processing and Management, 25(6) (1989), pp. 665–76.

[15] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Soft, 8 (1982), pp. 43–71.

[16] Y. Saad, *Numerical methods for large eigenvalue problems*, Manchester University press, 1992.