INFORMATION RETRIEVAL USING KRYLOV SUBSPACE METHODS

KATARINA BLOM, April 14, 2004

Submitted to Mathematical Sciences, department of Mathematics, Chalmers University of Technology and University of Göteborg. In partial fulfillment of the degree of Doctor of Philosophy.

School of Mathematical Sciences
Chalmers University of Technology
Göteborg, Sweden

INFORMATION RETRIEVAL USING KRYLOV SUBSPACE METHODS.
Katarina Blom.
ISBN: 91-7291-453-X

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 2135
ISSN 0346-718X

Mathematical Sciences, Department of Mathematics
Chalmers University of Technology, SE-412 96
Göteborg, Sweden.
Telephone: +46 (0)31 772 10 00
Telefax: +46 (0)31 16 19 73

Cover: Reidar Petersen

Those who seek shall find

INFORMATION RETRIEVAL USING KRYLOV SUBSPACE METHODS

KATARINA BLOM

Abstract

In this dissertation we discuss how simple Krylov subspace methods can be used for information retrieval (IR). The dissertation consists of two parts. The first part gives a background of IR and introduces the vector space model for IR and the Krylov subspace methods that we use. The second part consists of four articles.

The first article introduces the concept of subspace methods and in particular introduces how simple Krylov subspace methods can be used for IR.

In the second article we show how simple modifications of the original Krylov subspace method for IR can help to steer the process of what documents to bring in and to avoid, and there by increase retrieval performance.

Retrieval performance for IR-systems improves significantly if proper term weighting is used. Terms with high search values are weighted up and terms with low search values are weighted down. Several term weighting schemes appear in the IR community. In the third article we experiment with different term weighting schemes.

In the fourth article we discuss how the Krylov subspace method is able to indicate even weak connections between groups of relevant documents. We also show how simple modifications of the method can be used to decrease the scoring for irrelevant documents. All experiments in the fourth article are made on sets from the TREC (Text REtrieval Conference) collection.

Keywords Information retrieval, Relevance feedback, Bidiagonalization, Lanczos algorithm, band Lanczos algorithm, Latent Semantic Indexing, LSI, Vector space model, Krylov subspace, SVD, Singular value decomposition, Numerical linear algebra.

Acknowledgements

I would like to thank my supervisor Prof. Axel Ruhe for his encouragement, source of inspiration and for guiding me through this work. I am greatful to Thomas Ericsson for all the good advice, all the help, all the comments and all support.

I would like to thank Prof. Viggo Kann and Bernt Wennerg for reading and commenting on part of the material.

Special thanks to my mother and to Agneta Steffen for constant support.

Thanks to Fredrik for putting up with his mother so frequently working late.

Part of this work has been supported by the Swedish Research Council, Vetenskapsrådet, contract 2002-4152.

Contents

1 Introduction **15**
1.1 Evaluation of information retrieval systems 18
1.2 Vector space models 21
1.3 Extended Vector space models 23

2 Krylov methods **26**
2.0.1 The Lanczos method 26
2.0.2 The Golub-Kahan bidiagonalization procedure 27
2.0.3 The band Lanczos procedure 29
2.1 Krylov subspace methods and IR 30

3 Relevance feedback **38**

4 Conclusions **40**

List of figures

1 Recall-precision graph. 21
2 Recall-precision graph for the Medline collection. 32
3 Ranking of relevant documents for one query from the Medline collection. 34
4 Mean average precisions for the Medline collection. 37
5 Recall-precision graph for the Medline collection. 38

Figures 2 – 5 were made using the Medline collection[1].

The term document matrix was constructed using all nonzero length strings found in the set of documents as terms. A stop-list consisting of all terms appearing in more than 10% of all documents was constructed and the stop-words were removed. The weighting scheme `tnc·txx` [6] was used (i.e. the raw termfrequency was used for all nonzero elements in the term document matrix and the query vectors, one row normalization followd by one column normalization of the term document matrix using euclidean norm were then performed).

[1]E.A. Fox at the Virginia Polytechnic Institute and State University has assembled nine small test collections on a CD-ROM. These test collections have been used heavily throughout the years for evaluation of information retrieval systems. Among these sets are the Medline collection, a small collection with a small number of queries. The documents are abstracts in biomedicine received from the National Library of Medicine.

This thesis consists of an introduction and the following papers:

Paper I

K. Blom and A. Ruhe, *Information Retrieval using a Krylov subspace method*, submitted for publication, (2003)

Paper II

K.Blom, *Modified Krylov subspace methods for Information Retrieval*, tech. rep., Dept. of Mathematics, Chalmers University of Technology, 2003

Paper III

K. Blom, *Experimenting with different weighting schemes for the Krylov subspace method used for IR*, tech. rep., Dept. of Mathematics, Chalmers University of Technology, 2003

Paper IV

K. Blom, *A Krylov subspace method meets TREC*, tech. rep., Dept. of Mathematics, Chalmers University of Technology, 2003

Information Retrieval using Krylov subspace methods

Katarina Blom

April 14, 2004

1 Introduction

An information retrieval (IR) 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 images and graphs.

Many universities and public libraries use IR-systems to provide access to books, journals and other documents. Dictionary and encyclopedia databases are widely available. On the World Wide Web (WWW) there is an enormous amount of electronic information available. All this information is useless unless it can be searched efficiently.

Todays huge amount of electronic information turns IR into a large scale computer system problem. The Google search engine for example needs to efficiently (and rapidly) search more than 3 billion webpages[1] for more than 3000 search queries per second at peak traffic time [38]. Moreover the amount of information to search through steadily increases. A WWW search engine today needs to manage and process hundreds of terrabytes of information reliably and efficiently.

Also in retrieval performance we expect a lot from our search engines. We ask them questions about topics we are unfamiliar with ourselves and expect an organized response.

In numerical linear algebra Krylov subspace methods are one of the most important classes of methods available for computing eigenvalues and eigenvectors of large matrices and for solving linear systems. The techniques are

[1] in 2001.

based on projection methods onto Krylov subspaces. The methods have been used for large eigenvalue problems for a long time.

In this dissertation we explore Krylov subspace methods for IR. In experiments retrieval performance is in general better for the Krylov subspace methods we explore than for other vector space models, but performance is for the user to judge.

The methods we focus on work on textual documents but are likely to work well in a hyper linked environment such as the WWW as well.

We are not the first to use eigenvalues and eigenvectors in IR algorithms. *Latent Semantic Indexing* (LSI) Berry et al [4], Dumais et al [15], use the singular value decomposition (SVD) to separate the global and general structure, corresponding to the large singular vectors[2], from local or noisy information, which hides among the small singular vectors[3]. J. Kleinberg [24] studies the information inherent in the link structure of a hyperlinked environment (such as the WWW). He uses the leading singular vectors to determine what documents (webpages) are relevant to a search query.

Notation The notation used in this thesis is rather standard in the numerical linear algebra community. We use uppercase letters for matrices and lowercase letters for vectors. Lower case Greek letters usually denote scalars. Component indices are denoted by subscript. For example, a vector c and a matrix M might have entries c_i and m_{ij} respectively. On the occasions when both an iteration index and a component index are needed, the iteration is indicated by a parenthesized superscript, as in $c_j^{(r)}$ to indicate the jth component of the rth vector in a sequence. Otherwise c_j may denote either the jth component of a vector c or the jth column of a matrix C. The particular meaning will be clear from its context.

Mathematical background The *euclidean norm* of a vector $v = [v_1, v_2, \ldots, v_n]$ is equal to $\sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$ and will be denoted $\|v\|$.

The subspace spanned by the column vectors $y_1, \ldots, y_n$ of the $m \times n$ matrix Y is the *range* of Y and is denoted $R(Y) \subset \mathcal{R}^m$

[2] by large singular vectors we mean the singular vectors corresponding the large singular values

[3] i.e. the singular vectors corresponding to the small singular values

Let M be an $n \times n$ matrix. Let λ be a scalar and x a nonzero vector such that

$$Mx = \lambda x,$$

then λ is called an *eigenvalue* and x is a corresponding *eigenvector*. All the eigenvalues of M is called the *spectrum* of M.

The *Krylov subspace* $\mathcal{K}_r(M, x)$ of the square matrix M and starting vector x is spanned by the r vectors

$$x, Mx, M^2x, \ldots, M^{r-1}x$$

where x is any nonzero starting vector. The *block Krylov subspace* $\mathcal{K}_r(M, X)$ is spanned by the pr vectors in the block Krylov sequence

$$X, MX, M^2X, \ldots, M^{r-1}X$$

where the columns in the starting block $X = \begin{bmatrix} x_1 & x_2 & \ldots & x_p \end{bmatrix}$ are linearly independent.

A short introduction to the *Singular value decomposition* (SVD) of a matrix is given here. For more details see for example Golub and van Loan [19].

Theorem If $Y \in \mathcal{R}^{m \times n}$ then there exists matrices $U = \begin{bmatrix} u_1 & u_2 & \ldots & u_p \end{bmatrix} \in \mathcal{R}^{m \times p}$ and $V = \begin{bmatrix} v_1 & v_2 & \ldots & v_p \end{bmatrix} \in \mathcal{R}^{n \times p}$ with orthogonal columns such that

$$U^T Y V = \Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_p \end{bmatrix} \text{ and } Y = U \Sigma V^T \tag{1}$$

where $p = \min(m, n)$ and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$. For a proof see Golub and van Loan [19].

The σ_i are the *singular values* of the matrix Y and the vectors u_i and v_i are the i:th left and right singular vector respectively.

The singular values of Y are the non-negative square roots of the eigenvalues of $Y^T Y$ and the columns of U and V are orthonormal eigenvectors of YY^T and $Y^T Y$ respectively.

The SVD reveals information about the global structure of the matrix. It can also be used to compute a *reduced rank approximation* Y_k of Y. Let

the SVD of $Y = U \Sigma V^T$ be given and let

$$Y_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T. \tag{2}$$

If $k < r = rank(Y) \leq \min(m, n)$ then Y_k, which is constructed from the k largest singular triplets of Y, is the closest (in 2-norm) rank-k matrix to Y, i.e.

$$\min_{rank(B)=k} \|Y - B\|_2 = \|Y - Y_k\|_2 = \sigma_{k+1}.$$

1.1 Evaluation of information retrieval systems

An IR-system has to support certain basic operations. There must be a way to enter documents into the database and to delete them. There must be a way to search for documents and to rank them in relevance order with respect to a user query. There must also be a way to present them to the user.

A complete presentation of IR and IR-systems can be found in textbooks Frakes and Baeza-Yates [17], Baeza-Yates and Riberio-Neto [1] and Kowalski [27], for example.

The performance of an IR-system can be evaluated in many ways. We will consider *retrieval efficiency*, *execution efficiency* and *storage efficiency*.

Retrieval efficiency When a user issues a search for information on a topic, the system will start to give back documents that are relevant from the systems 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.

The retrieval efficiency depends on two main factors. The first is the ability of the system to retrieve relevant information and the second is the ability to dismiss irrelevant information. The ability to retrieve relevant information is measured by *recall*, the ratio of relevant documents retrieved over the total number of relevant documents for that query. A systems ability to reject irrelevant documents is measured by *precision*, the ratio of the number of relevant documents retrieved for a given query over the total number of documents retrieved. Precision and recall are usually inversely related (when precision goes up, recall goes down and vice versa).

One measure commonly used by the IR community to measure retrieval performance is *average precision*. Assume t documents are relevant to a search query q. When we evaluate q, all the documents are ranked in relevance order (with the most relevant from the systems perspective first) and we obtain an ordered list $\mathcal{L}$ of documents. Let $\ell_i, \quad i = 1, \ldots, t$ be the position for the ith document relevant to q in $\mathcal{L}$.

The average precision for a single query is defined as

$$\frac{1}{t}\sum_{i=1}^{t}\frac{i}{\ell_i}.$$

The *mean average precision* for multiple queries is defined as the mean of the average precisions for all queries.

Precision can be computed at any *actual recall level*

$$\frac{i}{t}, \quad i = 1, \ldots, t$$

(where t is the number of relevant documents to the query).

The number of relevant documents differ between queries. To compensate for this *interpolated precision* is computed at *standard recall levels* (usually recall values $0, 0.1, 0.2, \ldots, 1$).

Let r_j be the jth recall level from the 11 standard recall levels $0, 0.1, 0.2, \ldots, 1$. The interpolated precision for a query at standard recall level r_j is the maximum precision obtained for any actual recall level greater that or equal to r_j.

The *Recall level precision averages* for multiple queries are the means of the interpolated average precision values at each (standard) recall level for the queries.

The most commonly used method for comparing the performance of IR systems is to use recall-precision graphs. Recall level precision averages are used as input for plotting the recall-precision graphs.

The measures are illustrated by an example: Assume a document collection has 20 documents, four ($= t$) of which are relevant to a search query q. Further assume a retrieval system ranks the relevant documents first, second, fourth and fifteenth (thus the relevant documents appear at position 1, 2, 4 and 15 in the ordered list of ranked documents $\mathcal{L}$).

Precision and recall In order to compute precision and recall we need to determine the number of documents shown to the user. If assuming 10

documents were retrieved 3 relevant documents are among the retrieved ones making the precision $\frac{3}{10} = 0.3$ and the recall $\frac{3}{4} = 0.75$.

Average precision The average precision is a single valued measure. In the ordered list of documents $\mathcal{L}$ the four relevant documents were sorted first second fourth and fifteenth, the average precision is

$$\frac{1}{t}\sum_{i=1}^{t}\frac{i}{\ell_i} = \frac{1}{4}\left(\frac{1}{1}+\frac{2}{2}+\frac{3}{4}+\frac{4}{15}\right) \approx 0.75.$$

Since we have only one search query the mean average precision is 0.75.

Interpolated precision There are four relevant documents to q, thus there are four actual recall levels $\frac{i}{t}$:

$$\frac{1}{4} = 0.25, \frac{2}{4} = 0.5, \frac{3}{4} = 0.75 \text{ and } \frac{4}{4} = 1.$$

The interpolated precision for all standard recall levels up to 0.5 is 1, the interpolated precision for standard recall levels 0.6 and 0.7 is 0.75, and the interpolated precision for standard recall levels 0.8, 0.9 and 1 is 0.25. Since we have only one query the recall level precision averages are the interpolated precisions values. The recall level precision averages are used as input for plotting the recall-precision graphs. A recall-precision graph for this search query is in figure 1. Typically these graphs slope downward from left to right (the more relevant documents are retrieved (recall increases) the more nonrelevant documents are retrieved (precision decreases)).

For further details on measures, see Harman [21].

Execution efficiency Execution efficiency is measured by the time it takes for an IR-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). Execution efficiency will be measured in worst case running time and the ordinary big-O notation will be used.

Storage efficiency Storage efficiency is measured by the number of bytes needed to store the data.

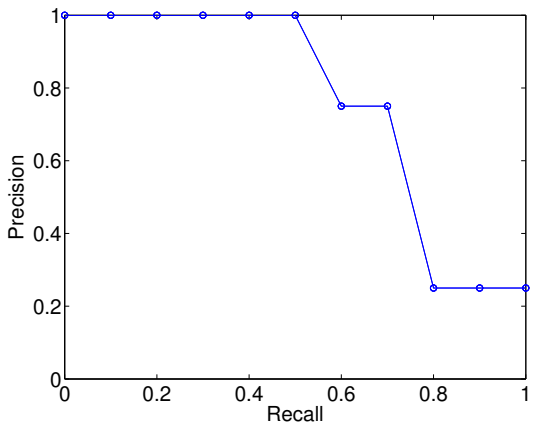

Figure 1: Recall-precision graph.

1.2 Vector space models

The *vector space model* was proposed over 30 years ago and SMART [37] (System for the Mechanical Analysis and Retrieval of Text) was one of the first implementations of a vector space IR model. In vector space models both queries and documents are encoded as vectors in m-dimensional space, where m is the number of unique terms in the collection. The n documents are stored as columns in a $m \times n$ *term document matrix* A and the queries q are stored as $m \times 1$ vectors. The nonzero elements in A correspond to the occurrences of each term in a particular document, i.e.

$$A = [a_{ij}] \tag{3}$$

where a_{ij} is nonzero if term i occurs in document j, zero otherwise. Similarly we let the ith element in the query vector be nonzero if term i appear in the query, zero otherwise. Global, local weightings and normalization factors are applied to increase/decrease the importance of terms within and among documents (and queries). Often $a_{ij} = g_i l_{ij} d_j$ where l_{ij} is the local weighting for term i in document j, g_i is the global weighting for term i and d_j is the document normalization factor for document j. There are several ways to

compute these weights. For summaries see for example Frakes and Baeza-Yates [17], Salton and McGill [37] or Kolda [25].

In [6] we explore the effect of 107 different combinations of term weighting schemes for the term document matrix together with 27 different weighting schemes for the query vectors. Retrieval performance for the Krylov method and the vector model (the vector model (4) is described below) are measured. Similar experiments have been carried out for vector space models and extended vector space models (extended vector space models are discussed in section 1.3). See for example Harman [20], Dumais [13], Salton et al [36] and Kolda et al [26, 25]. There is a large difference in retrieval performance between the best performing weighting scheme and the worst performing weighting scheme, so the weights for the nonzero elements in A need to be chosen with care.

Since every term does not normally appear in each document, the term document matrix is (very) sparse. A few terms, however, appear in all (or almost all) documents. These terms have no discrimination value during a search and are called stop words. A *stop list* consists of terms whose frequency and/or semantic use make them of no value as searchable words. Eliminating the terms appearing on the stop list usually decreases the total amount of terms used in the database dramatically. Since a sparse storage scheme can be used to store the term document matrix the number of bytes needed to store the matrix also decreases significantly, thus storage efficiency increases.

To further increase retrieval performance and to decrease the number of unique terms (the number of rows m in the term document matrix) *stemming* is sometimes used. Instead of using the original term, the suffixes and sometimes the prefixes are removed, and the stem of the term is used. Several algorithms for finding the stems of terms exist (for a summary see for example Baeza-Yates [17]). An often used stemmer is the Porter stemmer [33]. The retrieval effect of stemming is dependent on the nature of the vocabulary used. Used in small document collections stemming could improve recall. The effect of stemming also depends on what language the terms are in. For example in a stemming experiment using $54,000$ Swedish news articles a clear raise in precision was indicated [10]. But stemming might also cause non related terms to be mapped to the same stem (overstemming), which then will decrease precision.

In vector space models query matching can be viewed as a search in the column space of the term document matrix A. One of the most common

similarity measures used for query matching is to use the angle between the query vector q and the document vectors in A. The smaller the angle is, the more relevant the document is. In the vector model the cosines of the angles between the query vector q and the document vectors a_j are used to sort the documents in relevance order,

$$c_j = \frac{q^T a_j}{\|q\| \|a_j\|}, \quad j = 1, \ldots, n. \tag{4}$$

Documents d_j corresponding to large values in c are ranked highly.

The vector model has some major drawbacks. The terms used in the query vectors are often not the same as those by which the information searched has been indexed in the term document matrix. In the vector model all document vectors having no terms in common with the query vector will be orthogonal to the query vector and there by be ranked irrelevant. Many terms have more than one distinct meaning. In different contexts or when used by different people the same term takes on varying significance. Thus the use of a specific term in a query does not necessarily mean that all documents with this term are relevant to this query.

1.3 Extended Vector space models

Many extensions of the vector model have been proposed:

Latent Semantic Indexing (LSI) Berry et al [4], Dumais et al [15] see also Berry and Browne [3] uses the singular value decomposition (SVD) of the term document matrix A to separate the global and general structure, corresponding to the large singular vectors[4], from local or noisy information, which hides among the small singular vectors[5]. Instead of using the original term document matrix A when scoring documents for relevancy, a reduced rank representation A_k is used. Let $A = U\Sigma V^T$ be the SVD (1) of the term document matrix and let the reduced-rank representation (2)

$$A_k = U_k \Sigma_{kk} V_k^T, \tag{5}$$

where U_k and V_k are formed by the first k columns of U and V and Σ_{kk} is the first k rows and columns of Σ.

[4] by large singular vectors we mean the singular vectors corresponding the the large singular values

[5] by small singular vectors we mean the singular vectors corresponding the the small singular values

The documents are scored for relevancy, measuring the angles between the query vector q and each document vector in the reduced-rank representation A_k. The smaller angle, the more relevant the document is.

LSI has been reported to perform quite well on both rather large and small document collections, see for example Dumais [14], Dumais et al [15], Letche et al [29], Lochbaum et al [30]. It can handle synonymy (when two words mean the same) and polysemy (when one word has several distinct meanings depending on context) quite well. There are also text collections for which LSI is not significantly better than the original vector model.

In [23] Jessup and Martin analyse the behavior of LSI. They conclude that LSI gives improved performance for some document collections, provided a good matrix-rank decision was taken, compared to the vector model. The retrieval performance was found to be good for surprisingly low matrix-ranks and high matrix approximation errors[6].

LSI needs a substantial computational work to get the SVD, and there is no simple way to determine a good matrix-rank. The term document matrices are in general well conditioned. In general they have no gap in the singular value spectrum. Work on finding good reduced rank approximations for the term document matrices have been done by Berry [4] and Zha et al [40].

Since the singular vector matrices U_k and V_k are often dense, storage requirements for the reduced-rank representation A_k is often significantly larger than for the original term document matrix A. Kolda and O'Leary [26] propose to replace the SVD with a semi-discrete decomposition (SDD), where the matrix A is approximated with

$$A \approx X_s D_s Y_s^T.$$

The elements in X_s and Y_s take on values from {-1, 0, 1} (represented by two bits each) and D_s is diagonal. The rank s in the SDD will be larger than the rank of the singular value decomposition used in LSI, but since the X and Y matrices consist only of values from {-1, 0, 1} the SDD will require much less storage than the SVD used in LSI.

Cluster based rank reductions. Document clustering has been used to enhance information retrieval. This is based on the hypothesis that documents

[6] The error in matrix approximation e measure how well the reduced-rank representation A_k approximates the original matrix A. It is defined by the difference between A and A_k in Frobenius norm, $e = \|A - A_k\|_F$. The matrix approximation error decreases as the rank k increases.

having similar contents are also relevant to the same query. A fixed collection of text is clustered into groups or clusters that have similar contents. The similarity between documents is commonly measured using the cosine of the angle between the document vectors but other similarity measures appear.

Clustering approaches, where vectors are grouped around a carefully selected set of centroid or *concept vectors*, have a clear intuitive appeal, see Dhillon and Modha [11]. Park et al [32] compare the use of singular and centroid vectors in a general formulation of low rank approximations of the term document matrix A.

J Kleinberg [24] studies the information inherent in the link structure of a hyperlinked environment (such as the WWW). Here each column of the matrix A is still a document (web page) but now an element a_{ij} is nonzero if there is a link from the i-th page to the j-th. We borrow terminology from bibliometry of scientific publication, see Garfield [18], and call these term rows *cocitations*. Kleinberg calls the cited documents (columns) *authorities* and the citing (rows) *hubs*. The leading singular vectors $u = u_1$ and $v = v_1$ (1) of A determine the hub weight u_j and authority weight v_j of the j:th document (web page). Kleinberg seeks documents of high authority weight in a subset determined by the query. One problem with that approach is that the leading singular vectors stand for a general weight factor and the same high authority weight documents will be returned irrespective of the query, Kleinberg calls this *diffusion*. See Ding et al [12, 22] for further studies of these link structures. They let textual similarity give weights to the links, bridging the gap between term document and link similarity.

2 Krylov methods

Let M be a square matrix. One of the oldest techniques for solving eigenvalue problems is the *power method*. The method consists of generating the sequence of vectors

$$M^k v, \quad k = 0, 1, 2, \ldots \tag{6}$$

where v is some nonzero initial vector. This sequence of vectors, normalized appropriately, converges to the dominant eigenvector (i.e. an eigenvector associated with the eigenvalue of largest modulus)[7].

A *Krylov subspace* of a (square) matrix M, starting at the vector v, is a subspace of the form

$$\mathcal{K}_r(M, v) = \text{span}\{v, Mv, M^2 v, \ldots M^{r-1} v\},$$

i.e. a subspace spanned by the iterates of the power method.

Krylov subspace methods are based on projections methods, either orthogonal or oblique, onto Krylov subspaces. Krylov methods are one of the most important classes of methods available for computing eigenvalues and eigenvectors of large matrices and for solving linear equation systems. Well known Krylov subspace methods are the Hermitian Lanczos algorithm, the non-Hermitian Lanczos algorithm and the Arnoldi algorithm. For a full description of Krylov subspace methods see Bai et al [2]. A very nice description of Krylov subspace methods can also be found in Saad [35].

2.0.1 The Lanczos method

Given a symmetric $n \times n$ matrix M and a starting vector b, the Lanczos method generates a sequence of vectors v_k and scalars α_k and β_k such that M is reduced to tridiagonal form

[7] provided that there is only one eigenvalue λ_1 of largest modulus and that λ_1 is non-defective, and that the initial vector v has components in the invariant subspace associated with λ_1.

ALGORITHM LANCZOS(M,b,r):
Start with $\beta_1 v_1 = b$, $v_0 = 0$
for $k = 1, 2, \ldots, r$ **do**
$\quad w = Mv_k - \beta_k v_{k-1}$
$\quad \alpha_k = v_k^T w$
$\quad \beta_{k+1} v_{k+1} = w - \alpha_k v_k$
end.

where each $\beta_k \geq 0$ is chosen so that $\|v_k\| = 1$.

Let $V_r = \begin{bmatrix} v_1 & v_2 & \ldots & v_r \end{bmatrix}$ and $T_r = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \ddots & \ddots & \\ & & \ddots & & \beta_r \\ & & & \beta_r & \alpha_r \end{bmatrix}$. After r steps in the Lanczos procedure we have

$$MV_r = V_r T_r + \beta_{r+1} v_{r+1} e_r^T,$$

where e_r is the rth identity vector. In exact arithmetic we have $V_r^T V_r = I$, in reality, good orthogonality of the v_k-vectors is only observed at the beginning of the process.

The procedure terminates with $\beta_{r+1} = 0$ for some $r \leq n$. Lanczos iterations quickly produce approximate eigenvalues near the ends of the spectrum of M, so if the procedure is stopped before $\beta_{r+1} = 0$ some of the eigenvalues of T_r approximates some extreme eigenvalues of M.

Lanczos original work can be found in [28].

For the IR algorithms we will use the *Golub-Kahan bidiagonalization procedure* and the *band Lanczos procedure*. The procedures are described shortly in sections 2.0.2 and 2.0.3. For a more complete description please see for example Bai et. al. [2].

2.0.2 The Golub-Kahan bidiagonalization procedure

The Golub-Kahan algorithm applied to the rectangular matrix Y gives the Lanczos tridiagonalization of the symmetric matrices $Y^T Y$ and YY^T by transforming Y to a lower bidiagonal matrix B.

The algorithm starts with a normalized starting vector and computes two orthonormal matrices P and Q, adding one column for each step k, see [19] section 9.3.3. When using the method for IR, we start the algorithm with the normalized query vector $q_1 = q/\|q\|$ and use the term document matrix A (3) to compute the bases.

ALGORITHM BIDIAG(A,q,r):
Start with $q_1 = q/\|q\|$, $\beta_1 = 0$
for $k = 1, 2, \ldots, r$ **do**
$\quad \alpha_k p_k = A^T q_k - \beta_k p_{k-1}$
$\quad \beta_{k+1} q_{k+1} = A p_k - \alpha_k q_k$
end.

The scalars α_k and β_k are chosen to normalize the corresponding vectors.

Define

$$Q_{r+1} = \begin{bmatrix} q_1 & q_2 & \ldots & q_{r+1} \end{bmatrix}, \tag{7}$$

$$P_r = \begin{bmatrix} p_1 & p_2 & \ldots & p_r \end{bmatrix}, \tag{8}$$

$$B_{r+1} = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_r \\ & & & \beta_{r+1} \end{bmatrix}. \tag{9}$$

After r steps we have the basic recursions

$$\begin{aligned} A^T Q_r &= P_r B_r^T \\ AP_r &= Q_{r+1} B_{r+1}. \end{aligned} \tag{10}$$

The columns of Q_r will be an orthonormal basis of the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$ and the columns of P_r forms an orthonormal basis for the Krylov subspace $\mathcal{K}_r(A^T A, A^T q)$. The lower bidiagonal matrix $B_{r+1} = Q_{r+1}^T A P_r$ is the projection of A onto these Krylov subspaces and some of the singular values of B_{r+1} will be approximations to some of the singular values in A.

With r large enough the bidiagonalization procedure BIDIAG(A,q,r) can be used to compute a solution x_L for the least squares problem

$$\min_x \|Ax - q_1\|.$$

Let $0 < k < r$, then $x^{(k)} = P_k B_{k+1}^{+} e_1$ is an approximation to x_L received after k iterations in the BIDIAG procedure.

In most cases the first basis vectors in Q_{r+1} will be dominated by components from the singular vectors corresponding to the largest singular values[8].

It is also possible to reduce the matrix A to *upper* bidiagonal form by computing orthonormal bases for the Krylov subspaces $\mathcal{K}_r(A^T A, p)$ and $\mathcal{K}_{r+1}(AA^T, Ap)$, using the bidiagonalization procedure

$$\begin{aligned} \rho_k u_k &= Ap_k - \theta_k u_{k-1} \\ \theta_{k+1} p_{k+1} &= A^T u_k - \rho_k p_k \end{aligned}$$

with $k = 1, 2, \ldots, r$, $p_1 = p/\|p\|$ and $\theta_1 = 0$. If we start the iteration with $p = A^T q$ this bidiagonalizing procedure can be derived from the BIDIAG procedure discussed previously. The relationships between the bidiagonalizations are discussed by Paige and Saunders [31] and also by Golub and van Loan [19].

2.0.3 The band Lanczos procedure

The band Lanczos algorithm Ruhe [34] (see also Bai et al [2]) is based on block Krylov subspaces induced by a matrix M and a block of s linearly independent starting vectors

$$y_1, y_2, \ldots, y_s.$$

The band Lanczos algorithm constructs orthonormal vectors that form a basis for the subspace spanned by the first linearly independent vectors of the *block Krylov sequence*

$$y_1, y_2, \ldots, y_s, My_1, My_2, \ldots, My_s, M^2 y_1, M^2 y_2 \ldots$$

When we use the Band Lanczos method for IR we use the $m \times n$ term document matrix A (3). The procedure is defined below:

[8] However if the query vector q has large components along some singular vectors that do not correspond to the largest singular values of the term document matrix A then the first few basis vectors in Q_{r+1} (7) will contain large components along these singular vectors. If the components in q are not large enough or if the components correspond to the largest singular values then the first basis vectors in Q_{r+1} will be dominated by components from the singular vectors corresponding to the largest singular values [16].

Let $h_{ji} = 0$ when $i < 1$.

ALGORITHM BANDL(A, Q_s, r)
Start with s orthonormal vectors forming $Q_s = \begin{bmatrix} q_1 & q_2 & \ldots & q_s \end{bmatrix}$.
for $j = 1$ **to** r **do**
$\quad h_{jj} p_j = A^T q_j - h_{ji} p_i \qquad \leftarrow i = j - s, \ldots, j - 1$
$\quad h_{ij} = (Ap_j)^T q_i \qquad \leftarrow i = j + 1, \ldots, j + s - 1$
$\quad h_{(j+s)j} q_{j+s} = Ap_j - h_{ij} q_i \quad \leftarrow i = j, \ldots, j + s - 1$
end

The scalars h_{jj} and $h_{(j+s)j}$ are chosen so that $\|p_j\| = \|q_{j+s}\| = 1$. With $H_{r+s,r} = \begin{bmatrix} h_{ij} \end{bmatrix}$, the matrix $H_{r+s,r}$ is of size $r + s \times r$ and lower (s+1)-diagonal. Define $Q_{r+s} = \begin{bmatrix} q_1 & q_2 & \ldots & q_{r+s} \end{bmatrix}$ and $P_r = \begin{bmatrix} p_1 & p_2 & \ldots & p_r \end{bmatrix}$. In exact arithmetic we will have $Q_{r+s}^T Q_{r+s} = I$ and $P_r^T P_r = I$. After $r + s$ iterations the basic relations

$$\begin{aligned} A^T Q_r &= P_r H_{rr}^T \\ AP_r &= Q_{r+s} H_{r+s,r} \end{aligned}$$

will hold. The columns of Q_{r+s} will be an orthonormal basis of the block Krylov subspace $\mathcal{K}_{r+1}(AA^T, Q_s)$ in the document space, spanned by the starting block Q_s and the columns of A.

The columns of P_r similarly span a basis of the block Krylov subspace $\mathcal{K}_r(A^T A, A^T Q_s)$ in the term space spanned by the rows of A. The singular values of $H_{r+s,r}$ will be an approximation of some of the singular values in A.

2.1 Krylov subspace methods and IR

Let A be an $m \times n$ term document matrix and q the query vector and consider the Krylov subspace

$$\mathcal{K}_r(A^T A, A^T q) = \{A^T q, (A^T A) A^T q, \ldots, (A^T A)^{r-1} A^T q\}. \tag{11}$$

If the columns of the term document matrix and the query vector are normalized, the first iterate $A^T q$ is simply the cosines of the angles between the query vector q and each document vector in A. Thus sorting the documents

for relevance according to this vector will rank the documents according to the vector model (4).

The vector model only measure closeness between the query vector q and each vector in the set. However there are usually many ways to express a given concept, so the actual terms in the query may not match those of relevant documents. If instead sorting the documents in relevance order according to the second iterate $(A^TA)A^Tq$ also the document vectors closeness in A will be taken into account when ranking documents. Relevant document vectors orthogonal to the query vector could (at least in theory) be ranked high.

Under reasonable mild conditions the iterates (11) converge to the dominant eigenvector of A^TA, the right singular vector

$$v_1 \tag{12}$$

corresponding to the first singular value σ_1 of A. Thus sorting the documents for relevance according to one of the later iterates in the sequence (11) will rank documents that share many terms with other documents high. The influence from the query vector will be gone (i.e. we get the same ranking of documents for all queries)[9].

In the recall-precision graph (figure 2) we compare the performances when the documents from the Medline collection are sorted in relevance order according to the four first iterates from the Krylov sequence (11) respectively. Performances are measured using interpolated mean average precision. For this particular set using the second iterate $(A^TA)A^Tq$ to score the documents is best in average. Using this iterate improve the scoring for 90% of the queries[10] compared to the vector model scoring A^Tq. Note that there are queries for which the third and fourth vectors from the Krylov sequence give best performance.

Consider the Krylov subspace

$$\mathcal{K}_{r+1}(AA^T, q) = \{q, AA^Tq, \ldots, (AA^T)^r q\}. \tag{13}$$

[9]Kleinberg [24] let each document vector in A correspond to a webpage, in a subset determined by the query, and each element a_{ij} in A is nonzero if there is a link from page i to page j. He let the leading right singular vector v_1 of A determine the authority weight of the page [24], pages with high authority weight correspond to large elements in the singular vector v_1.

[10]27 out of 30 queries.

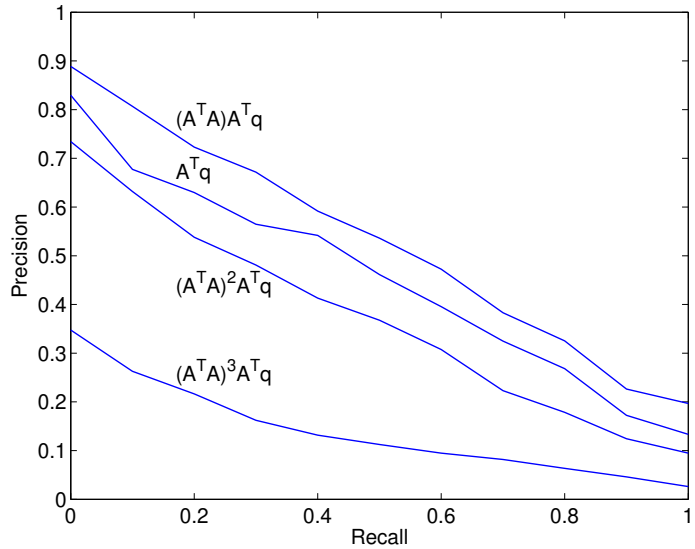

Figure 2: Recall-precision graph for the Medline collection.

The ith element in the first vector, q, in the sequence (13) is nonzero if term i appears in the query vector q indicating only presence or absence of terms in the query. The element in row i and column j in AA^T will be nonzero if document i and document j have at least one term in common, thus the ith element in AA^Tq is nonzero if any of the nonzero term elements appearing on row i in AA^T also appear in the query vector. AA^Tq can be seen as an *extended query vector*. The iterates further on in the sequence can also be viewed as extended query vectors.

Under reasonable mild conditions (6) the iterates (13) converge to the the left singular vector u_1 corresponding to the largest singular value σ_1 of A.

If we score the documents in A measuring the cosine of the angle between each document vector and any of the extended query vectors from the sequence (13) respectively we get the same scorings as in figure 2.

Instead of using only one vector from the subspaces (11,13) when scoring the documents we may use the information inherited in several. In the *subspace projection measure* the documents in A are sorted according to their closeness measured in angles to the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$. The closer the document is the more relevant it is.

In figure 3 we follow the document ranking for one query from the Medline collection using the subspace projection measure for $r = 0, 1, \ldots, 4$ ($r = 0$ corresponds to the vector model). Only two relevant documents are ranked worse for $r > 0$ compared to the vector model (4) the other ten relevant documents improve their ranking for some $r > 0$. The improvement in ranking is significant for four of the documents (right plot). The best ranking for all the relevant documents does not appear at the same value of r (which makes it hard to find an optimal value on r). Eleven of the 12 relevant documents are ranked 15 or better when $r = 4$, which is much better than the vector model.

The bidiagonalization procedure in Bidiag (from section 2.0.2) computes orthonormal bases Q_{r+1} (7) and P_r (8). The columns of Q_{r+1} have the dimension of a kind of query, while the columns of P_r can be interpreted as choices among the documents in the collection A. The first column p_1 contains those documents that contain terms in the query, say its brothers and sisters, and the next p_2 can similarly be interpreted as cousins and so on. The number of documents reached will grow in a chain letter fashion, so we can hope that rather few steps k will be sufficient to reach all documents that have any connection with the original query vector q.

Each step k can be interpreted as first applying the current query q_k,

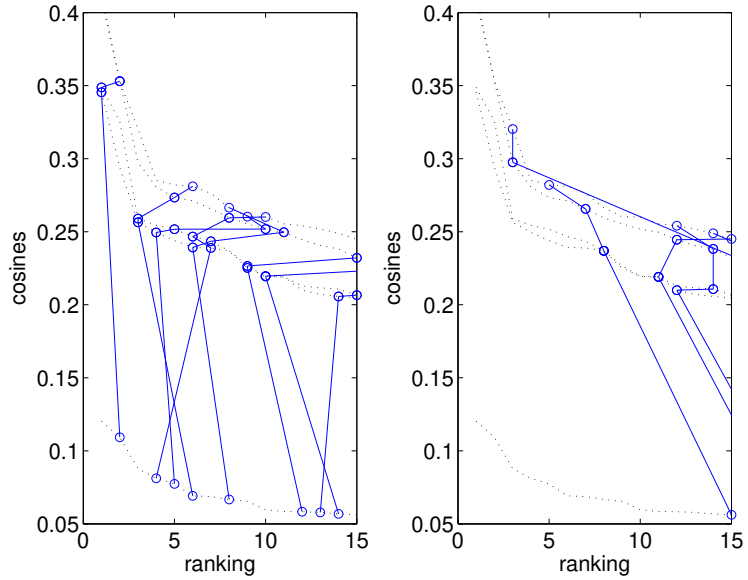

Figure 3: Ranking of relevant documents for one query from the Medline collection using the subspace projection measure. The dotted lines (...) are (from bottom to top) the cosines of the angles between each document vector and the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$ for $r = 0, 1, \ldots, 4$. The circles (–o) marks the ranks for each relevant document and the solid lines follow each document's ranking when r grows. The **left plot** show the documents that were ranked bellow 15 when $r = 0$ ($r = 0$ corresponds to the vector model (4)). The **right plot** show the documents that were ranked 15 or worse when $r = 0$.

(in matrix language computing $A^T q_k$), giving a new choice, that is strongly different from previous choices (in matrix language it is orthogonal). Then all terms from the chosen documents are combined in the multiplication Ap_k, to give a new query q_{k+1} orthogonal to all previous queries $q_1, \ldots, q_k$. After k steps we have made k queries to the data base, all of them orthogonal to each other. Some readers may remember the children's game "master mind".

Using the bases matrices Q_{r+1} (7) and P_r (8) (and the lower bidiagonal matrix B_{r+1} (9)) several options to rank the documents for relevancy appear. The subspace projection measure was already mentioned. A few measures will be mentioned here, for a more detailed description please see Blom, Ruhe [9, 8].

- In the *subspace projection measure* the documents are sorted in decreasing order according to the cosines of the angles between the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$ and the document vectors in A

$$c_j^{(r)} = \|Q_{r+1}^T a_j\|, \quad j = 1, \ldots, n. \tag{14}$$

- The matrix $\hat{A}_r = Q_{r+1} B_{r+1} P_r^T$ from the basic recursions (10) can be interpreted as an approximation to the reduced rank approximation $A_r = U_r \Sigma_r V_r^T$ (2) from the SVD of A. For the *LSI-like measure* the documents are scored measuring the angles between the query vector and each document vector in $\hat{A}_r$. The documents are sorted in decreasing order according to the cosines of the angles between the query vector q and the document vectors in the reduced rank matrix $\hat{A}_r$

$$c_j^{(r)} = \frac{q^T \hat{A}_r e_j}{\|q\| \|\hat{A}_r e_j\|}, \quad j = 1, \ldots, n, \tag{15}$$

  where e_j is the jth identity vector.

- For the *expanded query measure* we let the *reached subspace* W form an orthonormal basis for the column vectors in AP_r. A projected query vector

$$\hat{q} = WW^T q \tag{16}$$

  is constructed using the reached subspace and the query vector q. The documents are sorted for relevancy measuring the angle between $\hat{q}$ and each document vector in the term document matrix A. The documents are sorted in decreasing order according to the cosines of the angles between the projected query vector $\hat{q}$ and the document vectors in the term document matrix $A = [a_j]$

$$c_j^{(r)} = \frac{\hat{q}^T a_j}{\|\hat{q}\| \|a_j\|}, \quad j = 1, \ldots, n. \tag{17}$$

  The expanded query measure $c^{(r)}$ can be interpreted as a correction of the vector model scoring (for a derivation see [9]), that is

$$c^{(r)} = A^T q + \gamma p_{r+1} \tag{18}$$

  where

$$\gamma = \alpha_{r+1} h_{r+1,r+1} h_{1,r+1}.$$

  The vector p_{r+1} is the $(r+1)$th basis vector from the BIDIAG procedure in section 2.0.2. The scalar α_{r+1} comes from the BIDIAG procedure and $h_{r+1,r+1}$ and $h_{1,r+1}$ are elements in $H = [h_{ij}]$ the orthogonal factor in the QR factorization $B_{r+1,r} = HR$. It is easy to verify [9] that γ tend to zero when r grows.

In figure 4 mean average precisions for the three similarity measures subspace projection measure (14), LSI-like measure (15) and expanded query measure (17) are compared for number of iterations r in the BIDIAG procedure ranging from 0 to 10.

For $r = 0$ all three similarity measures are equal to the vector model (4). The expanded query measure converges rapidly to the vector model again and for $r > 4$ the similarity measures are roughly equal.

Both the LSI-like measure and the subspace projection measure give best mean average precision when $r = 1$, the LSI-like measure never score better than the expanded query measure. For a more general result please see [6].

The best choice of r varies between different queries, so if computing average precisions always for the best choice of r mean average precision increase for all three measures.

It is important to notice that there are several reasons for keeping r, the number of iterations in the BIDIAG procedure, low. Since a new bidiagonalization is performed for each new query vector q the execution efficiency would be to low if we allow r to be large. Moreover if iterating too long in the procedure the influence of the query vector q in the bases Q_{r+1} (7) and P_r (8) will be lost. In a real case we will have one or maybe two singular values converged in B_{r+1} (9) when the BIDIAG procedure is interrupted.

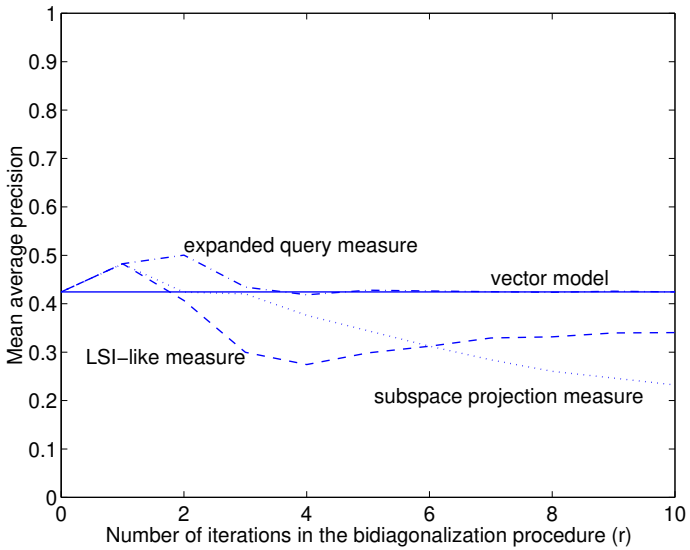

Figure 4: Mean average precisions for the Medline collection.

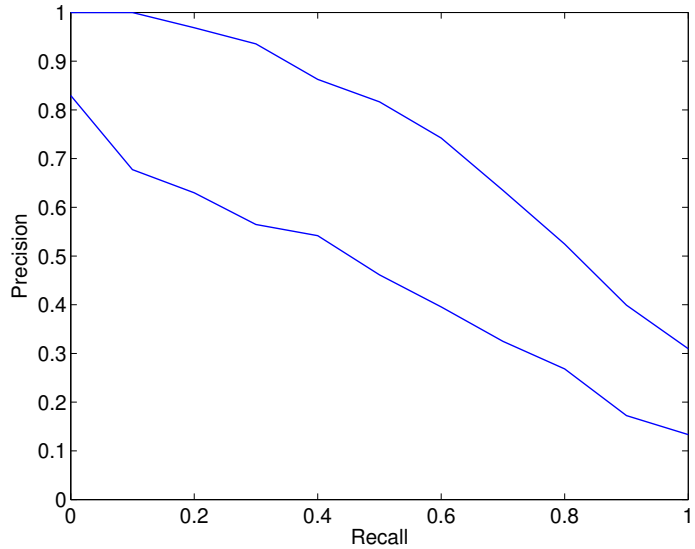

Figure 5: Recall-precision graph for the Medline collection. The vector model (lower curve) is compared with the relevance feedback algorithm (upper curve) using the second bidiagonalization procedure presented in section 2.0.2.

3 Relevance feedback

In a relevance feedback cycle, the user is presented a list of retrieved documents, and after examining them, marks those that are relevant. The main idea of using *relevance feedback* is to use the information provided by the user to make a new improved search[11].

In figure 5 we used the vector model (4) and let the user judge the 10 best ranked documents for relevancy. Based on the users's judgement we

[11]Relevance feedback can also be performed without a involving a user. In *pseudo relevance feedback* new queries are constructed using the top retrieved documents, see for example Xu and Croft [39].

constructed a starting vector p for the second bidiagonalizing procedure presented in section 2.0.2. We let the ith element in p be nonzero if the ith document was retrieved and relevant to the query. We sorted the documents for relevancy according to the vector WW^Tp where W is a basis for the column vectors in A^TP_r from the second bidiagonalization procedure. The relevant documents that were ranked high in the vector model will be ranked high also in the relevance feedback algorithm, but in the relevance feedback algorithm also relevant documents that were ranked low in the vector model significantly increase in ranking.

Naturally the retrieval performance for the Krylov subspace methods increase if relevance feedback is used. In Blom [7] other modified Krylov subspace methods are discussed. Common numerical linear algebra methods, such as *explicit restart* are used to improve retrieval efficiency. The band Lanczos method (section 2.0.3) gives an opportunity to start the bidiagonalization with several (relevant) document vectors and is further discussed in [7].

4 Conclusions

In this dissertation we introduce the Krylov subspace methods for information retrieval. Krylov subspace methods together with IR is an interesting technique. It opens many possibilities. The process is query based and a new approximation is made for every new query, which makes it possible to adapt the bidiagonalization to each query entered. Connecting relevance feedback to the process is simple and a natural continuation (for an example of steering the process please see [5]). Since the process is query based it becomes more flexible than LSI.

We do believe that using linear algebra methods for IR opens a lot of possibilities to improve performance for IR algorithms. The linear algebra methods we use are well known and were built to handle large sets of data, thus the computations in the BIDIAG and BANDL procedures have low computational complexity. The behaviour of the methods have been well examined.

With this thesis we want to give some insight to how linear algebra techniques can be used to improve IR. Rather than presenting optimal algorithms we want to point to several possible IR linear algebra techniques. We also hope that this dissertation can stimulate further development of using linear algebra techniques for IR.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley, 1999.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[3] M. W. Berry and M. Browne, *Understanding Search Engines. Mathematical Modeling and Text Retrieval*, SIAM, 1999.

[4] 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.

[5] K. Blom, *A Krylov subspace method meets TREC*, tech. rep., Dept. of Mathematics, Chalmers university of Technology, 2003.

[6] ——, *Experimenting with different weighting schemes for the Krylov Subspace method used for IR*, tech. rep., Dept. of Mathematics, Chalmers university of Technology, 2003.

[7] ——, *Modified Krylov subspace methods for information retrieval*, tech. rep., Dept. of Mathematics, Chalmers university of Technology, 2003.

[8] K. Blom and A. Ruhe, *Information Retrieval using very short Krylov sequences*, in Computational Information Retrieval, M. W. Berry, ed., SIAM, 2000, pp. 39–52.

[9] ——, *Information Retrieval using a Krylov Subspace method*, submitted for publication, (2003).

[10] J. Carlberger, H. Dalianis, M. Hassel, and O. Knutsson, *Improving Precision in Information Retrieval for Swedish using Stemming*, in NODALIDA01 - 13th Nordic Conference on Computational Linguistics, Uppsala, 2001.

[11] I. S. Dhillon and D. S. Modha, *Concept decompositions for large sparse text data using clustering*, Machine Learning, 42 (2001), pp. 143–175.

[12] C. Ding, H. Zha, X. He, P. Husbands, and H. Simon, *Analysis of hubs and authorities on the web*, Tech. Rep. CSE-01-013, Department of Computer Science and Engineering, Pennsylvania State University, 2001.

[13] S. T. Dumais, *Improving the retrieval of information from external sources*, Behavior Research Methods, Instruments, & Computers, 23 (1991), pp. 229–236.

[14] S. T. Dumais, *Using LSI for information filtering: TREC-3 experiments*, in Overview of TREC-3 Conference, D. K. Harman, ed., 1995.

[15] 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.

[16] L. Eldén, *Partial Least Squares vs. Lanczos Bidiagonalization I: Analysis of a Projection Method for Multiple Regression*, Tech. Rep. LiTH-MAT-R-2002-24, University of Linköping, Dept. of Mathematics, 2002.

[17] W. B. Frakes and R. Baeza-Yates, *Information Retrieval, Data Structures and Algorithms*, Prentice Hall, 1992.

[18] E. Garfield, *Citation indexing – its theory and application in science, technology, and humanities*, Wiely, New York, (1979). reprinted 1983 by ISI Press, Philadelphia.

[19] G. Golub and C. F. van Loan, *Matrix Computations*, Johns Hopkins, 3 ed., 1996.

[20] D. Harman, *Ranking algorithms*, in Information Retrieval, Data Structures and Algorithms, W. B. Frakes and R. Baeza-Yates, eds., Prentice Hall, 1992, pp. 363–392.

[21] ——, *The Eighth Text REtrieval Conference (TREC-8)*, NIST Special Publication 500–246. http://trec.nist.gov/pubs/trec8/t8_proceedings, (2000), p. A1 (Appendix).

[22] X. He, H. Zha, C. Ding, and H. Simon, *Web document clustering using hyperlink structures*, Tech. Rep. CSE-01-006, Department of Computer Science and Engineering, Pennsylvania State University, 2001.

[23] E. R. Jessup and J. H. Martin, *Taking a new look at the latent semantic analysis approach to information retrieval*, in Computational Information Retrieval, M. W. Berry, ed., SIAM, 2000, pp. 121–144.

[24] M. Kleinberg, *Authoritative sources in a hyperlinked environment*, Journal of the ACM, 46 (1999), pp. 604–632.

[25] T. G. Kolda, *Limited-memory matrix methods with applications*, PhD thesis, Applied Mathematics, University of Maryland, 1997.

[26] T. G. Kolda, *A semi-discrete decomposition for latent semantic indexing in information retrieval*, ACM-Trans Information Systems, 16(4) (1998), pp. 322–346.

[27] G. Kowalski, *Information Retrieval Systems, Theory and Implementation*, Kluwer Academic Publishers, 1997.

[28] 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.

[29] T. A. Letche and M. W. Berry, *Large-scale information retrieval with Latent Semantic Indexing*, Information Sciences - Applications, 100 (1997), pp. 105–137.

[30] K. E. Lochbaum and L. Steeter, *Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retreival*, Information Processing and Management, 25(6) (1989), pp. 665–676.

[31] 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.

[32] H. Park, M. Jeon, and J. B. Rosen, *Lower dimensional representation of text data in vector space based information retrieval*, in Computational Information Retrieval, M. Berry, ed., Proceedings in Applied Mathematics, SIAM, Philadelphia, 2001, pp. 7–27.

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

[34] A. Ruhe, *Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices*, Mathematics of Computation, 33 (1979), pp. 680–687.

[35] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, 1992.

[36] G. Salton and C. Buckley, *Term-weighting approaches in automatic text retrieval*, Information Processing & Management, 24 (1988), pp. 513–523.

[37] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[38] `http://www.google.com`.

[39] J. Xu and W. B. Croft, *Query expansion using local and global document analysis*, Proc. ACM SIGIR, (1996), pp. 4–11.

[40] H. Zha, *A subspace-based model for information retrieval with applications in latent semantic indexing*, Lecture Notes in Computer Science, 1475 (1998), pp. 29–42. Springer Verlag.

A Krylov subspace method for Information Retrieval.

Katarina Blom* Axel Ruhe†

April 14, 2004

Abstract

An new algorithm for information retrieval is described. It is a vector space method with automatic query expansion. The original user query is projected onto a Krylov subspace generated by the query and the term-document matrix. Each dimension of the Krylov space is generated by a simple vector space search, first using the user query and then new queries generated by the algorithm and orthognal to the previous query vectors.

The new algorithm is closely related to latent semantic indexing, LSI, but it is a local algorithm that works on a new subspace of very low dimension for each query. This makes it faster and more flexible than LSI. No preliminary computation of the singular value decomposition, SVD, is needed and changes in the data base cause no complication.

Numerical tests on both small (Cranfield) and larger (Financial Times data from the TREC collection) data sets are reported. The new algorithm gives better precision at given recall levels than simple vector space and LSI in those cases that have been compared.

keywords Information Retrieval, Vector space model, Query expansion, Latent Semantic indexing, Singular value decomposition, Lanczos algorithm, Krylov subspace.

*Department of Computing Science, Chalmers Institute of Technology and the University of Göteborg, SE-41296 Göteborg, Sweden `blom@cs.chalmers.se`. Financial support has been given by The Swedish Research Council, Vetenskapsrådet, contract 2002-4152

†Department of Numerical Analysis and Computer Science, NADA, Royal Institute of Technology, SE-10044 Stockholm, Sweden `ruhe@kth.se`

1 Introduction

The purpose of an information retrieval (IR) system is to seek through a large collection of information items, or *documents*, to retrieve those relevant to information requests, or *queries*, stated by a user. In the present contribution, we will show how computational tools from Numerical Linear Algebra can be helpful. We will use IR criteria to decide success or failure of the algorithms developed. How large part of the relevant documents are found, and how many of the retrieved documents are relevant to the user?

The documents may be books in a library, documents in a data base of news telegrams, scientific papers in journals or web pages on the world wide web (WWW). Each document contains *terms*, words that are significant in some way. The query is also formulated in terms of the same kind. We will look at the document collection as a huge matrix, where there is one row for each term that occurs anywhere in the collection and each column represents one document. This *term-document* matrix is denoted A throughout this paper. We let the element a_{ij} in row i and column j of A be nonzero if the i-th term is present in document number j, zero otherwise. The term document matrix will typically be very large and very sparse. The query will be expressed by the same terms as the documents, i.e. as a column vector q, where the i-th element q_i is nonzero if the i-th term is a part of the query, zero otherwise.

A very simple IR algorithm is to choose those documents that contain any of the terms in the query. This *Boolean search* can be expressed as a row vector $p^T = q^T A$, where each element p_j is the scalar product between the query vector q and a document column vector a_j of A, and choosing those documents for which p_j is nonzero. (We use the common linear algebra convention of letting a Latin letter stand for a column vector and ^{T} stand for transposing a column into a row. The matrix A has the columns $A = [a_1, a_2, \ldots, a_n]$.)

The *vector space model* is a refinement of Boolean search. The numerical values of the scalar products p_j are used to get angles between the query vector q and the document vectors a_j. The documents are scored, starting with those that make the smallest angle to the query vector.

In the present contribution we will study refinements of the vector space model. The main emphasis is on *subspace metods*, where we project the query and document vectors on a carefully chosen subspace, and use the angles between these projected vectors to determine closeness. We show that in

many cases subspace methods behave in a similar way to methods based on *query expansion*, another common class of refined vector space methods.

One subspace method is Latent Semantic Indexing [7], where the dominant principal component subspace computed by the singular value decomposition, SVD, is used. It is supposed to filter away noisy and particular information from the general and relevant information that we need to distinguish between documents on different subjects. Another subspace method is based on a known classification and uses *concept vectors* [5, 11]. Our main interest is a new subspace method based on *Krylov sequences* of subspaces reachable from the query vector. The first steps of the Krylov sequence correspond to a query expansion that is closely related to query expansion based on co-occurrences as introduced by Sparck Jones [12] and studied by Xu and Croft [13].

The advantage of our approach, compared to LSI, is that it works on the original term document matrix A, no SVD computation is needed in the outset, and it is trivial to add and delete terms and documents between queries. The main computational work is the same as a few applications of a naive vector space search, the rest is manipulation of small matrices.

1.1 Summary of contents

After some preliminary explanations of numerical linear algebra and information retrieval notations in this section, we describe subspace methods in section 2. We explain their common characteristics and show that some well known algorithms can be characterized as subspace methods, using different subspaces. We also discuss the relation between subspace methods and query expansion. In section 3 we describe the Krylov subspace algorithm we have used. It is simply the well known Golub Kahan bidiagonalization [8], applied to the term document matrix A, starting at the query q. It is used to find an expanded query $\hat{q}$, which is used to compute angles to score the document vectors a_j. We also give quantities that can be used to determine convergence. In our context the algorithm is stopped at a much earlier stage than for instance when solving least squares problems. Finally, in section 4, we show results of some numerical experiments, using both the small and well known Cranfield data and a larger test matrix coming from the Financial Times collection in the TREC material [10].

We have formulated our algorithm and got some preliminary results in the licentiate thesis of the first author [2]. Experiments on the small matrices

are reported in more detail in the conference contribution [4]. Details on term weighting, experiments on more data sets and the inclusion of relevance feed back is discussed in the thesis [3].

1.2 Notations

Matrices: Throughout this paper, A will denote the $m \times n$ term document matrix. The j:th column vector of the matrix A will be denoted a_j and the j:th column vector of the identity matrix I will be denoted e_j.

Singular Value Decomposition: Let

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

be the SVD of A, see [9]. The best rank s approximation to A in the Frobenius or sum of squares norm is

$$A^{(s)} = U_s \Sigma_{ss} V_s^T \tag{2}$$

where U_s and V_s are formed by the first s columns of U and V and the $s \times s$ diagonal matrix Σ_{ss} has the s largest singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_s$ in its diagonal.

Seen as a mapping, the $m \times n$ matrix A maps the n dimensional space $\mathcal{R}^n$ into its range space $\mathcal{R}(A)$, the subspace of $\mathcal{R}^m$ which is spanned by the columns of A. Its dimension is r the rank of A.

Krylov spaces: A Krylov subspace of a square matrix C, starting at the vector v, is a subspace of the form

$$\mathcal{K}_r(C, v) = \text{span}\{v, Cv, C^2v, \ldots C^{r-1}v\}. \tag{3}$$

Increasing the dimension r we finally get the entire reachable subspace of the pair (C, v). Its dimension is $r \leq n$, the dimension of v.

Measures: Two standard measures used by the information retrieval community are *Precision* and *Recall*. Precision is the ratio of the number of relevant documents retrieved for a given query over the total number of documents retrieved. Recall is the ratio of relevant documents retrieved over the total number of relevant documents for that query. Precision and recall

are usually inversely related (when precision goes up, recall goes down and vice versa). A recall level for a particular query can be arbitrarily chosen from $\frac{1}{t}, \frac{2}{t} \ldots 1$ where t is the number of relevant documents to this particular query.

In order to show precision at various recall levels graphically, interpolation may be used. The *interpolated precision* at a recall cutoff R for one query is defined to be the maximum precision at all recall levels greater than and equal to R.

The *average precision*, is a single valued measure that reflects performance over all relevant documents. Average precision is the average of the precision value obtained after each relevant document is retrieved. Average precision will reward systems that rank all relevant documents high, the last relevant document found is equally important as the first.

When reporting results for test sets with multiple queries, we will consider the *mean interpolated average precision* over all queries at a fixed sequence of recall cutoff values.

A way to compare performance when finding the first relevant documents is *document level average*, DLA(i), the precision when a certain number, i, of documents are retrieved. It mimics the use of a search engine where 10 documents are presented to the user each time. Then DLA(10) is the fraction of those that are relevant. For further details, see Harman [10].

Relevance is always judged by comparing the results of an algorithm to relevance judgments provided with the test sets. These have been compiled by a panel of human experts who have considered at least all those documents marked as relevant.

2 Subspace methods

In a general sense, the vector space method works in a space $\mathcal{D}$ of all documents that can be expressible as texts. This space of all possible documents has a countably infinite number of dimensions, and it is not simple to determine closeness between two documents. We therefore choose to see each document as a bag of terms, and represent it as a vector $a_j \in R^m$ in the m dimensional space of document vectors. This is already a rather severe restriction, we have reduced the dimension from infinity to m. We have also made a choice of which words we regard as significant, and used these words as terms.

When terms are chosen, we represent the query as a vector $q \in R^m$. We use angles between the query vector q and the document vectors a_j to determine which documents to retrieve in the naive vector space method.

In our information retrieval task, we have a finite collection of n documents to choose from, they build up a document collection space $\mathcal{A} = \mathcal{R}(A)$, the range space of the term document matrix A, which is of dimension at most n. Most often the number of terms m is larger than the number of documents, $m > n$, and the documents are linearly independent, making $\mathcal{A}$ into an n dimensional subspace $\mathcal{A} \subset R^m$. The query vector q is not in this subspace $\mathcal{A}$, but we may use the projected query vector $P_{\mathcal{A}}q$, and retrieve those documents a_j that are closest to that vector. If we use angles in the Euclidean space to decide closeness, this will yield the same ranking as when we use the angles between the document vectors and the original query vector.

A wide class of IR algorithms can now be classified as *subspace algorithms* where we restrict our view to a subspace $\mathcal{S} \subset \mathcal{A}$ and use angles between a projected query $\hat{q} = P_{\mathcal{S}}q$ and projected documents $\hat{a}_j = P_{\mathcal{S}}a_j$.

Let us look at some natural choices of subspaces $\mathcal{S}$:

2.1 Dominant subspace: Latent semantic indexing

Latent Semantic Indexing, LSI, [7] uses the singular value decomposition, SVD (1) of the term document matrix

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

and choose the space of the leading s singular vectors (2)

$$\mathcal{S} = \text{span}\,[U_s]$$

It separates the global and general structure, corresponding to the large singular vectors, from local or noisy information, which hides among the small. LSI has been reported to perform quite well on both rather large and small document collections. See for example Dumais [6]. It can handle synonymy (when two words mean the same) and polysemy (when one word has several distinct meanings depending on context) quite well. However LSI needs a substantial computational work to get the SVD, and there is no simple way to determine how many singular vectors s that are needed to span the leading subspace. Work on this has been done by M Berry [1] and H Zha et al [14].

2.2 Classification: Centroid vectors

The singular vectors make up a basis of the best rank s approximation to the given term document matrix A, and this can be considered as the best subspace if nothing else is known. On the other hand, if we know that the documents are taken from a set of subclasses, we may use a carefully selected set of centroid or *concept vectors*, as a basis of another subspace $\mathcal{S}$, see Dhillon and Modha [5]. Park et al [11] compare the use of singular and centroid vectors in a general formulation of low rank approximations of the term document matrix A.

2.3 Reachable subspaces: Krylov sequences

In the present contribution, we will try a third sequence of subspaces. We will let the subspaces be determined by the query vector q. We take it as the Krylov sequence of subspaces of vectors reached from q via a small number k of naive vector space searches.

In matrix language, this means that we take the query vector q, multiply it with the transposed term document matrix A to get a ranking or *scoring vector* $p = A^T q$. Each element p_j of p is a scalar product between the query vector q and the corresponding document vector a_j, so the elements of p give a ranking from the naive vector space method (if the columns of A are normalized). In this first step of the Krylov sequence, we find those documents that are directly related to the query, let us say its sisters.

In the second step, we multiply this scoring vector p with the term document matrix A to get a new vector $q_2 = Ap$, a new query that contains all the terms that were contained in the documents that p pointed to. If we apply this new query, we get $p_2 = A^T q_2$ which points to all documents that contain any of all the terms in q_2, i. e. those two links away from the query, let us say its cousins.

In later steps this continues in a chain letter fashion, and soon we will reach all documents in the collection that are *reachable* from the query, to borrow a term from Control Theory. In matrix language,

$$\mathcal{S} = \mathcal{K}_k(AA^T, q) \tag{4}$$

after k steps, see (3).

In our computation we do not just follow the Krylov sequence, we also make the vectors $q_1, q_2, \ldots, q_r$ and $p_1, p_2, \ldots, p_r$ into *orthogonal* bases. Intuitively this means that we remember what we asked for in the first query q_1, and make a totally different query next time, q_2. This is standard practice in numerical linear algebra.

2.4 Relevant subspaces

There is a fourth subspace that is of theoretical interest, and can be used for comparison purposes. That is the *relevant subspace* $\mathcal{Z}$ spanned by those documents that are relevant to the query q. This subspace is not possible to use in any practical algorithm, it supposes that all the relevant documents are already known. However, it is interesting to see if the query q is closer to the relevant subspace $\mathcal{Z}$, than to any other subspace spanned by a similar number of document vectors. Are there many irrelevant documents that are closer to the relevant subspace $\mathcal{Z}$ than the query q?

In a way, the properties of the relevant subspace determine if there is any hope for any algorithm, built up by tools from numerical linear algebra, to find the relevant documents to a given query.

2.5 Subspaces and query expansion

Subspace algorithms are closely related to another class of refined vector space IR methods built up around *query expansion*. Say that the subspace algorithm takes a subspace $\mathcal{S}$ in any of the manners described in the previous subsections, and uses the angles between the projected query $\hat{q} = P_{\mathcal{S}} q$ and the projected documents $\hat{a}_j = P_{\mathcal{S}} a_j$, to determine which documents a_j that are relevant to the query q. The cosine of this angle is

$$\hat{c}_j = \frac{\hat{q}^T \hat{a}_j}{\|\hat{q}\|_2 \|\hat{a}_j\|_2}$$

The scalar product in the numerator is

$$\hat{q}^T \hat{a}_j = (P_{\mathcal{S}} q)^T P_{\mathcal{S}} a_j = q^T P_{\mathcal{S}}^T P_{\mathcal{S}} a_j = q^T P_{\mathcal{S}} a_j = (P_{\mathcal{S}} q)^T a_j = \hat{q}^T a_j \,,$$

provided that the projection is orthogonal, $P^T = P$. We see the scalar product between the projected query vector $\hat{q}$ and the projected document vector $\hat{a}_j$ is the same as that between the projected query $\hat{q}$ and the *original* document vector a_j. Using scalar products to determine closeness, the subspace method based on $\mathcal{S}$ gives the same result as a straightforward vector space

method using the expanded query $\hat{q}$. The angles are not invariant however, since the norms in the denominator differ. We know that $\|\hat{a}_j\|_2 \leq \|a_j\|$ giving a larger cosine or smaller angle in the subspace than in the query expansion case.

Still, the result of a subspace method based on $\mathcal{S}$ is closely related to using the expanded query $\hat{q} = P_{\mathcal{S}} q$ in the original vector space method.

When we choose $\mathcal{S}$ as a Krylov subspace (4), our choice of query expansion is related to the technique of Sparck Jones [12]. The second vector in the the Krylov sequence (4), $\tilde{q}_2 = AA^T q$, weighs in components of all terms that are co-occurring with the terms in the original query. The weights give an emphasis to the co-occurrence in the documents that are ranked highest in the vector space search, $p = A^T q$, giving an effect similar to the local expansions of Xu and Croft [13].

3 The Krylov subspace algorithm

We use the Golub Kahan bidiagonalization algorithm [8] to compute the Krylov sequence of subspaces (4). It is a variant of the Lanczos tridiagonalization algorithm and is widely used in the numerical linear algebra community .

The Golub Kahan algorithm starts with the normalized query vector $q_1 = q/\|q\|$, and computes two orthonormal bases P and Q, adding one column for each step k, see [9] section 9.3.3.

Algorithm Bidiag
Start with $q_1 = q/\|q\|_2\,, \beta_1 = 0$
For $k = 1, 2, \ldots, r$ *do*

1. $\alpha_k p_k = A^T q_k - \beta_k p_{k-1}$
2. $\beta_{k+1} q_{k+1} = A p_k - \alpha_k q_k$

End

The scalars α_k and β_k are chosen to normalize the corresponding vectors.

Define

$$\begin{aligned} Q_{r+1} &= \begin{bmatrix} q_1 & q_2 & \cdots & q_{r+1} \end{bmatrix}, \\ P_r &= \begin{bmatrix} p_1 & p_2 & \cdots & p_r \end{bmatrix} \\ B_{r+1,r} &= \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \alpha_r & \\ & & \beta_{r+1} & \end{bmatrix}. \end{aligned} \tag{5}$$

After r steps we have the *basic recursion*,

$$\begin{aligned} A^T Q_r &= P_r B_{r,r}^T \\ A P_r &= Q_{r+1} B_{r+1,r} \end{aligned}$$

The columns of Q_r will be an orthonormal basis of the Krylov subspace (4),

$$\text{span}\,[Q_r] = \mathcal{K}_r(AA^T, q) \subseteq \mathcal{R}([A\,q]) \tag{6}$$

in the document space, spanned by the query q and the columns of A. The columns of P_r similarly span a basis of the Krylov subspace

$$\text{span}\,[P_r] = \mathcal{K}_r(A^T A, A^T q) \subseteq \mathcal{R}(A^T)\,, \tag{7}$$

in the term space spanned by the rows of A.

We see that $B_{r+1,r} = Q_{r+1}^T A P_r$ is the projection of A into these Krylov subspaces and the singular values of $B_{r+1,r}$ will be approximations to those of A.

If $\beta_k = 0$ for some $k \leq r$ we have exhausted the Krylov space (6), reachable from the query q. Then $Q_k B_{k,k} P_k^T$ is the restriction of A to this reachable subspace, and the singular values of $B_{k,k}$ are a subset of those of A.

The columns of AP_r span the *reached subspace* after r steps starting from q. It is the intersection between the Krylov subspace (6) and the column space of A,

$$\mathcal{R}(AP_r) = \text{span}\,[Q_{r+1} B_{r+1,r}] \subseteq \mathcal{R}(A) \tag{8}$$

The basic recursion (6) implies that it has the orthonormal basis W_r, where

$$W_r = Q_{r+1} H_{r+1,r}\,, \tag{9}$$

with $H_{r+1,r+1}$ the orthogonal factor in the QR factorization,

$$B_{r+1,r} = H_{r+1,r+1} R\,. \tag{10}$$

Note that since $B_{r+1,r}$ is bidiagonal, $H_{r+1,r+1}$ will be both orthogonal and Hessenberg and can be computed as a product of r elementary rotations.

The projected query vector It is now easy to use the basis W_r (9) to project the query and the documents into the reached subspace (8). The projected query $\hat{q}$ is

$$\hat{q} = P_{\mathcal{R}(AP_r)} q = W_r W_r^T q = W_r H_{r+1,r}^T e_1 = W_r \begin{pmatrix} h_{1,1} \\ h_{1,2} \\ \vdots \\ h_{1,r} \end{pmatrix} \tag{11}$$

and we see that the first row of H gives the coordinates of the query in the basis W. When we run several steps r of our algorithm, new columns are added to H, but when one column $r+1$ is added in step r, it is only the last r-th column that is modified.

We get the projected document $\hat{a}_j$ similarly as,

$$\hat{a}_j = W_r W_r^T a_j\,. \tag{12}$$

3.1 Scoring documents

We may regard our algorithm as a subspace method and choose the angles between the query and each of the document vectors, projected onto the reached subspace (8),

$$\mathrm{Css}_j^{(r)} = \frac{\hat{q}^T \hat{a}_j}{\|\hat{q}\|_2 \|\hat{a}_j\|_2} \quad j = 1 \ldots n. \tag{13}$$

Alternatively we may regard our algorithm as a query expansion method and use the angles between the projected query and the original documents,

$$\mathrm{Cqe}_j^{(r)} = \frac{\hat{q}^T a_j}{\|\hat{q}\|_2 \|a_j\|_2} \quad j = 1 \ldots n. \tag{14}$$

We compute these quantities using the basis W (9) and the small orthogonal Hessenberg $H_{r+1,r+1}$ (10). Apply an elementary orthogonal transformation S_r to make all elements but the first in the first row of $H_{r+1,r} S_r$ zero. Then $W_r S_r$ forms a new basis of the reached subspace (8). The first element $(y_j^{(r)})_1$ in the vector

$$y_j^{(r)} = S_r^T W_r^T a_j$$

will give the component of a_j along $\hat{q}$ and the rest of the projected $\hat{a}_j$ (12) as the norm of the remaining elements in $y_j^{(r)}$. Thus the subspace cosine (13) is

$$\mathrm{Css}_j^{(r)} = \frac{(y_j^{(r)})_1}{\|y_j^{(r)})\|_2}\,,$$

while the query expansion cosine (14) is slightly smaller at

$$\mathrm{Cqe}_j^{(r)} = \frac{(y_j^{(r)})_1}{\|a_j\|_2}\,.$$

Our experiments have shown that using the query expansion cosines Cqe_j (14) of the angles between projected query and original documents for scoring, often gives better performance than the subspace cosines Css_j (13), so we use query expansion, Cqe_j,as our standard. It gives a preference for documents whose vectors a_j are closer in angle to the reached subspace.

3.2 Following progress

In the Krylov method, a new bidiagonalization is performed for every query vector q. Thus the number of iterations must be small. The optimal number of iterations r is different for various queries. Choosing the optimal number r of iterations is an interesting and important problem. Figure 1 show performance for the Cranfield set using different numbers of iterations r. Performance is measured by average precision. It is clear from this figure that best average performance for all queries is reached when three iterations are performed. When more than three iterations are used, the performance rapidly converges towards the performance of the vector model. Note that some queries show optimal performance after two iterations and very few after one iteration. For one iteration, performance is worse than the performance for the vector model for most queries. This pattern of performance

(initial worse than the vector model, increasing performance and then a rapid convergence towards the vector model) was observed for most of the queries in all data sets we tested.

The convergence towards the vector model performance can easily be explained and estimated using quantities from the bidiagonalization algorithm presented.

Consider the least squares problem

$$\min_x \|Ax - q\|_2, \tag{15}$$

where A is the term document matrix and q is the query vector. It can be solved using the BIDIAG algorithm (see for example the textbook [9]). In step k the distance between the query vector and the projected query vector $\hat{q}^{(k)}$ is the residual

$$d^{(k)} = q - Ax^{(k)} = q - \hat{q}^{(k)}.$$

Here $x^{(k)}$ is the solution to problem (15) in step k. The distance decreases as we let k grow, but will not tend to zero unless the query is a linear combination of the documents in A [1].

The *normal equation residual* $A^T d^{(k)} A^T(q - \hat{q}^{(k)})$ to the problem (15) will tend to zero as k grows. If the normal equation residual converges monotonously to zero [2] then it is not surprising that the average precision for the Krylov method, using the query expansion scoring $\mathrm{Cqe}_j^{(k)}$ (14), tends to the scoring of the vector model. This is precisely what we see in figure 1. Note that, even if the convergence of $A^T d^{(k)}$ is monotonuous, the convergence for the average precisions does not have to be monotonuous. Looking closely into figure 1, a few such examples are visible.

Finally $d^{(k)}$, the distance between the query and its projection and the normal equation residual $A^T d^{(k)}$, can easily be computed for each step k in the bidiagonalization procedure.

In step k the distance between the query q and the projected query $\hat{q}^{(k)}$

[1] In our tests no query vector is completely in the range of A

[2] The convergence of the normal equation residual is not in general monotonuous. For all tests we made however, the convergence was monotonuous for at least the first 10 iterations.

is

$$\begin{aligned} d^{(k)} &= q - \hat{q}^{(k)} \\ &= Q_{k+1}e_1 - Q_{k+1}H_{k+1,k}H_{k+1,k}^T e_1 \\ &= Q_{k+1}(I - H_{k+1,k}H_{k+1,k}^T)e_1 \\ &= Q_{k+1}h_{k+1}^{(k)}h_{k+1}^{(k)T}e_1 \\ &= Q_{k+1}h_{k+1}^{(k)}h_{1,k+1}^{(k)} \end{aligned} \tag{16}$$

and its norm is just

$$\|d^{(k)}\| = |h_{1,k+1}^{(k)}| \tag{17}$$

The normal equation residual is

$$\begin{aligned} A^T d^{(k)} &= A^T Q_{k+1}h_{k+1}^{(k)}h_{1,k+1}^{(k)} \\ &= P_{k+1}B_{k+1,k+1}^T h_{k+1}^{(k)}h_{1,k+1}^{(k)} \\ &= P_{k+1}\begin{pmatrix} B_{k+1,k}^T \\ 0 \quad \alpha_{k+1} \end{pmatrix} h_{k+1}^{(k)}h_{1,k+1}^{(k)} \\ &= P_{k+1}\begin{pmatrix} 0 \\ \alpha_{k+1}h_{k+1,k+1}^{(k)} \end{pmatrix} h_{1,k+1}^{(k)}. \end{aligned} \tag{18}$$

Its norm is

$$\|A^T d^{(k)}\| = |\alpha_{k+1}h_{k+1,k+1}^{(k)}h_{1,k+1}^{(k)}|. \tag{19}$$

3.3 Complexity of the algorithm

In the BIDIAG algorithm, the matrix vector multiplications are performed between a sparse matrix and a dense vector. The number of operations needed is proportional to the number of nonzero elements in A. The rest of the algorithm consists of subtracting and normalizing vectors of length m. In exact arithmetic we will have $Q_{r+1}^T Q_{r+1} = I$ and $P_r^T P_r = I$ (5). In standard floating point arithmetic, fully accurate orthogonality of these vectors is only observed at the beginning of the process. In order to recover the orthogonality some type of reorthogonalization would be necessary. This would of course add operations to the complexity of the algorithm. Since we keep the number of iterations r very small, we believe that no reorthogonalization is needed. The main computational work for the document scoring (13) (14) again is in the size of multiplying a sparse matrix with a dense vector.

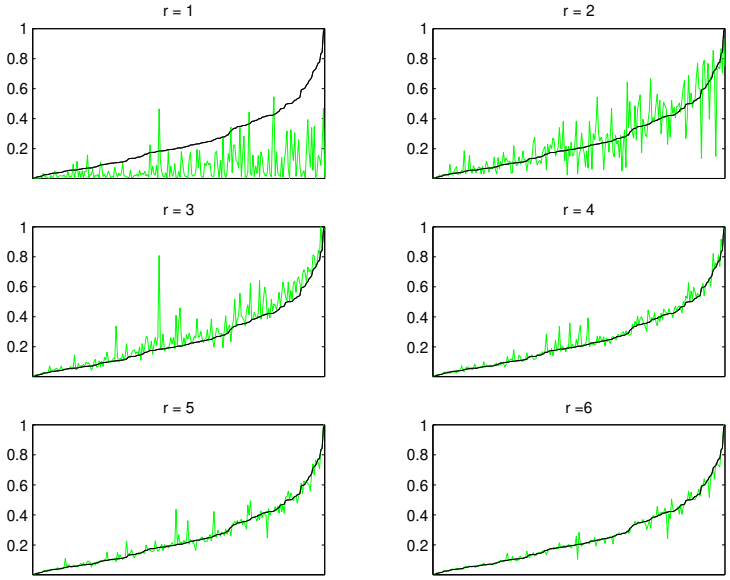

Figure 1: Average precision, for all 225 queries using the Cranfield set for $r = 1, 2 \ldots 6$ in the BIDIAG algorithm. The dark lines are the vector model and the light grey lines are the Krylov subspace model. Queries are sorted after increasing vector model apr.

4 Numerical experiments

Data sets: Each one of the test collections we have used consists of a document data base and a set of queries for which relevance judgments are available.

For illustration and comparison purposes, we have used the small and widely circulated data sets Medline, Cranfield, ADI and CICI.

We have also used larger test collections received from the Text Retrieval Conference (TREC) [10]. The TREC 4 disc contains 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.

Tests on data from the Cranfield collection and from the Financial Times collection will be reported here. Similar tests have been made for the Medline, ADI, CICI and Congressional Record collections.

Parsing the data sets: For both collections, any non-zero length string of characters, delimited by white space or return, was regarded as a term. All terms that occurred in more than 10% of the documents were removed. They were considered to be common words of no interest for the retrieval. Each element $a_{i,j}$ in the term document matrix was set to the number of occurrences of term number i in document j .

The size of the Cranfield matrix is $7,776$ terms$\times 1,400$ documents. Before starting the bidiagonalization process, first the rows and then the columns of the term document matrix were normalized. This tends to deemphasize common terms and long documents.

The Financial Times term document matrix is of size $m = 343,578$ terms by $n = 210,158$ documents with $26,790,949$ nonzero elements. The columns were normalized before the bidiagonalization algorithm BIDIAG was started.

Results for the Cranfield collection: There are 225 queries supplied with the test matrix, together with indices j of relevant documents for each each query. This gives between 2 and 40 relevant documents for each query, 476 documents were not relevant to any of the queries, 417 documents were relevant to just one, while the remaining 507 documents were relevant to more than one and at most 8 of the 225 queries. We compare our results to these correct answers.

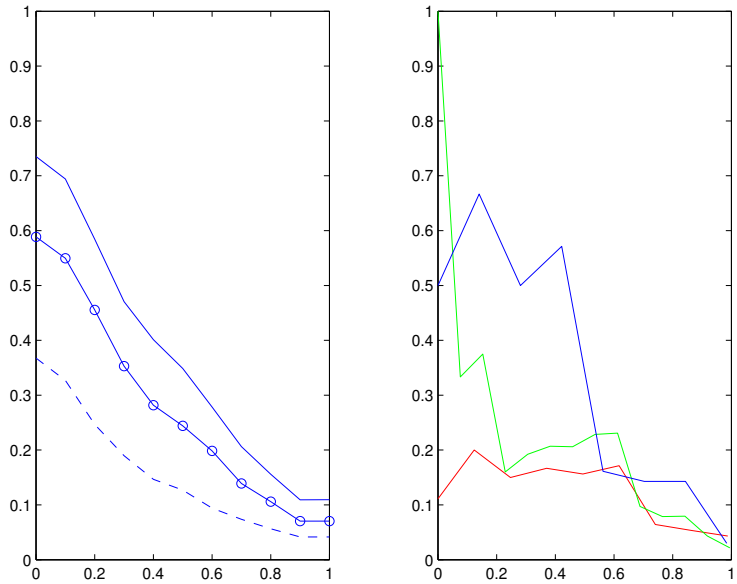

Figure 2: Precision as a function of recall for the Cranfield collection. Left: Interpolated and averaged over all queries (recall level precision average). Dashed (- -) is vector model, line with circle (-o) is LSI for rank $s = 296$, line (-) is our Krylov algorithm for r=3 steps. Right: Our Krylov algorithm to r=3 for 3 different queries, precision at actual recall levels.

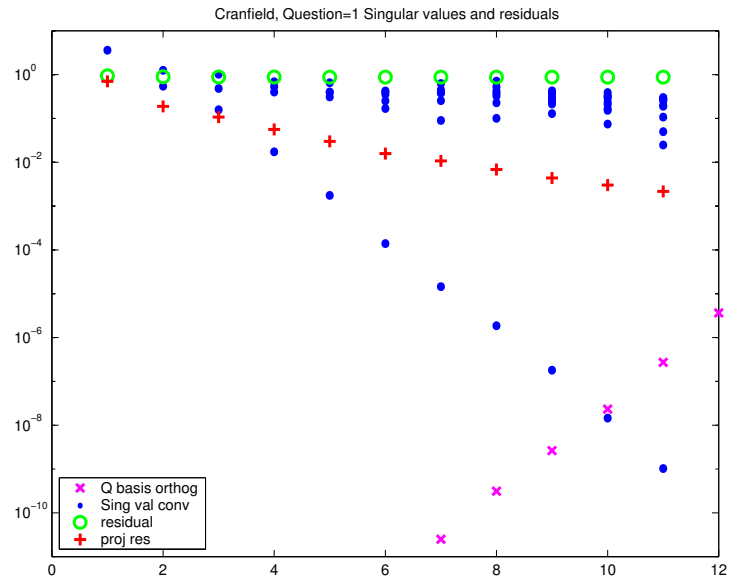

Figure 3: Cranfield matrix, follow convergence of bidiagonalization procedure starting at query q_1.

We first summarize the performance in an averaged precision-recall graph. In figure 2 the vector model is compared to LSI and our algorithm, as described in section 3, run for $r = 3$ steps. For the LSI method the optimal rank $s = 296$ in the low rank approximation of A (2) was obtained by computing the sum of the average precisions for each query and simply picking the s with the largest sum. It is clear that our Krylov algorithm gives the best averaged precision at all recall levels for these Cranfield data.

Let us look into the details and follow the Golub Kahan algorithm on one query. Take query 1, it has 29 relevant documents which is rather many for a Cranfield query. Our algorithm scores this query reasonably well. In figure 3 we follow the progress in linear algebra terms, as we execute the algorithm for steps $k = 1, \ldots, 12$. Circles are the residual norms $\|r^{(k)}\|$, (17), they decrease

unnoticeably slowly from 1 to 0.879. This means that the query q is at a rather large angle to the reached subspace (8), it has a projection of length 0.477. We plot the normal equation residuals $\|A^T r^{(k)}\|$, (19), as pluses, and note that they decrease fast enough at a linear rate. After 12 steps we have found the projection of the query into the document space spanned by A to nearly 3 decimals.

We were curious to see how the singular values converged and plotted estimates of their accuracies as points. Note that the leading singular value converged very fast, after 12 steps its vector is accurate to 9 decimals and the singular value to full machine precision. It is well known that the basis vectors Q_k keep orthogonal until one of the singular values converges. We plotted the orthogonality of each basis vector q_k to its predecessors Q_{k-1} as crosses and, true to theory, the crosses and points intersect at half the machine accuracy level 10^{-8} during step 10.

Let us now turn to a view of all the documents, and see how well we find the relevant documents for query 1. We plot them in a two dimensional coordinate system in figure 4. The x axis is along the projected query $\hat{q}$ (11). The y axis is used to plot the component of each a_j in the reached subspace (12) orthogonal to $\hat{q}$. This makes up two of the three components of each a_j vector. We can infer the length of the third component, which is orthogonal to the reached subspace, by remembering that all vectors a_j were normalized to unit length, so the distances of the points plotted to the origin indicate how close the vectors are to the reached subspace. Those shown close to the origin are far from the reached subspace. If we continue the bidiagonalization to full length $r = n$, most of the vectors will get unit length, because then the reached subspace is the whole span of A, except in the rare case when the query is totally unrelated to a part of the document collection.

If we use our standard query expansion based scoring method (14), taking angles between the original documents and the projected query, we would choose documents from right to left as plotted in figure 4, and we can check how well we find the relevant documents. We show this by giving the ranking beside each of the 10 highest scored relevant documents. Look at the lower part of figure 4 which shows the situation after $r = 2$ steps. First comes documents 1, 2, and 3 they are all relevant. Then the next relevant document is retrieved as number 6, we see two non relevant documents as points above and closely below the circle with number 6. Then the next relevant document is retrieved as number 9. Now our algorithm has given us 10 suggestions, of which we find that 5 are relevant. We say that DLA(10), the document level

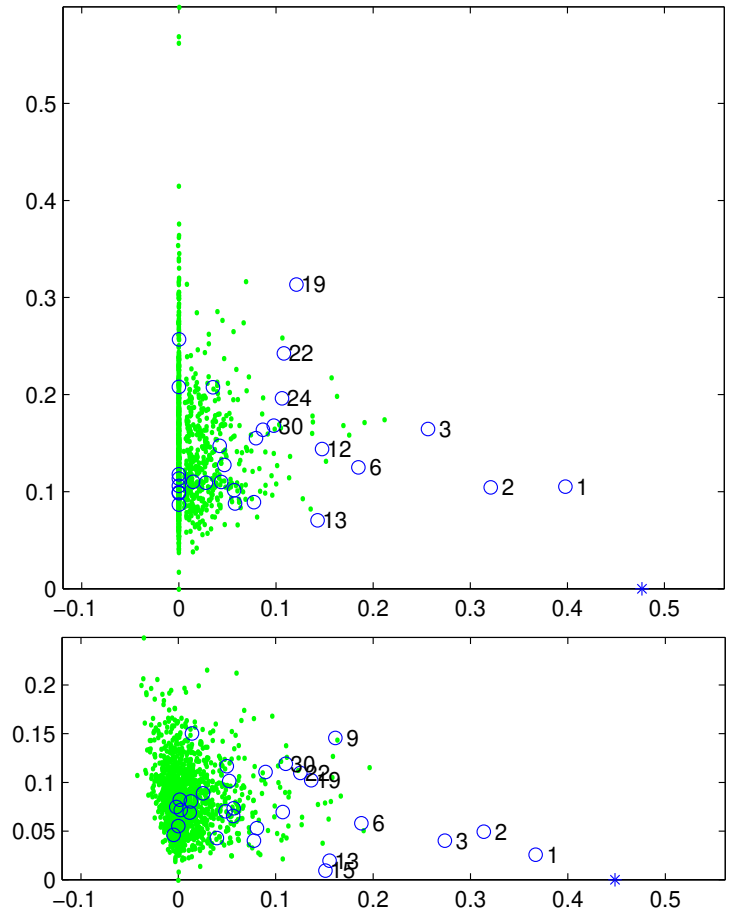

Figure 4: Cranfield matrix, Query 1, upper half step $r = 12$, lower half step $r = 2$. Numbers are rankings given by the algorithm to relevant documents. Circles mark relevant documents while points mark those not relevant. Asterix marks the projected query.

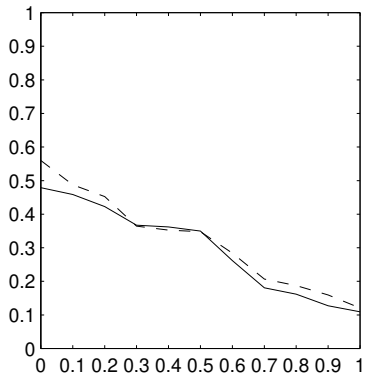

Figure 5: Interpolated precisions for recall levels $0, 0.1, \ldots 1$ for the Financial Times collection from the TREC data base. The vector model (–) is compared to our algorithm for $r = 3$ (- -) . The average of the 25 documents that are best ranked by the vector space method are included.

average precision after 10 documents is 0.5. The average precision over all relevant documents [10], is lower, 0.297, since the last relevant documents are found much later, we see that the 10:th relevant document scores as number 30 while the 29:th and last one does not appear until 1029.

Look at the upper half of figure 4, the final one after $r = 12$ steps. There are many points along the y axis, they denote documents that are orthogonal to the projected query, and will be the last ones scored. Actually 933 of the 1400 documents are orthogonal to the original query.

When scoring documents by angles in the reached plane (13), these can be seen as angles to the x axis in figure 4. It did not differ much from the standard query expansion scoring (14), for some queries it was better for others it was worse. For this Query 1, it gave about the same average precision at 0.296 and retrieved relevant documents ranked as 1,2,3,4,5,9, giving a DLA(10) = 0.6. The third scoring choice (angles to Krylov subspace) amounts to choosing those documents plotted far from the origin in figure 4, and gives about the same choices but with lower average precision, 0.180, and DLA(10) = 0.4.

Results for the Financial Times collection: There are several queries provided with the TREC collection. We have used query number 251 to 350. Nine of the queries do not have any relevant answers among the Financial Times documents, and for the rest of the queries there are between 1 and 280 relevant documents. Altogether 3,044 of the 210,158 documents are relevant to some query, 116 documents are relevant to two queries and 7 documents are relevant to three queries.

In figure 5 the vector model is compared to our algorithm run to $r = 3$. The experiments were made in the same way as for figure 2, but we did not have results for LSI for this large matrix. Documents were scored using the standard query expansion scores (14). We did choose $r = 3$ as dimension of the Krylov subspace, here the results were better for larger subspaces for some of the queries.

We choose such a query, number 344, to report in figure 6. As for figure 4, the x axis is along the projected query $\hat{q}$ (11) and the y axis is used to plot the component of each document vector in the reached subspace. The labels show the ranking of the relevant documents, there are only 3 relevant documents among all the 210,158, quite like seeking a needle in a haystack. Note that the relevant documents get better ranking for the larger subspace $r = 6$ than for $r = 3$. This question is not one of the 25 best questions included in figure 5.

Discussion: The experiments have shown good performance for the small data set (Cranield), but not that good performance for the larger Financial times (FT) set. Although we cannot notice any major differences in the structure of the term document matrices or the distribution of singular values, there are differences between the two sets. The FT set consists of news telegrams and Cranfield of scientific papers. For the Cranfield collection, most users will probably agree on the relevance judgements given for this set, while for the FT documents more subjectivity is involved in the relevance judgements. We believe the larger sets do reflect a more realistic case.

The construction of the FT matrix also plays a role in the performance of our algorithm. Perhaps more care has to be taken when deciding what terms to use for the matrix. It might not be enough to remove all terms occurring in more than 10% of the documents, maybe that figure should be 5% or something else.

Some type of row and column normalization is useful. In our Cranfield ex-

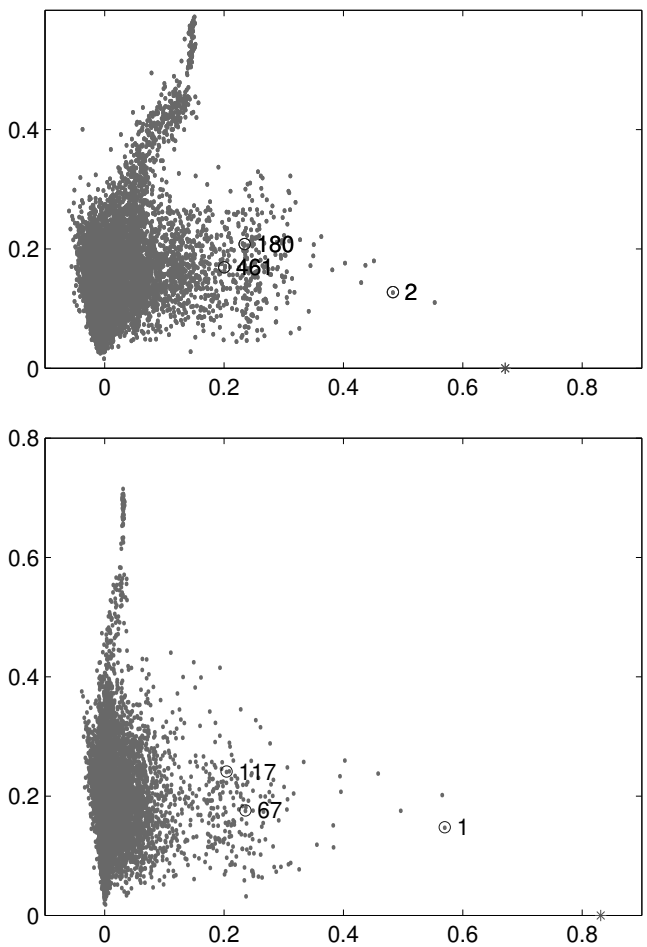

Figure 6: The TREC Financial Times matrix. Query no. 344, upper half step r = 3 and lower half step r = 6, numbers rankings of relevant documents. For the upper half 97 % of the documents are in the interval < 0.1 and for the lower half 99 % of the documents are in that interval, only a sample of those are shown. Asterix marks projected query

periments, we first normalized the row vectors, and then the column vectors. Even if the normalization of the column vectors destroys the row normalization, a smoothing effect remains. This had some effect for the performance for the Cranfield matrix. For the FT matrix only the columns were normalized.

The starting vector (the query) in our algorithm plays an important role, and it might also benefit our algorithm to pay more attention to how to construct the query vector. We have only tried our algorithm for at most $r = 12$ steps, since generating a larger subspace is too time consuming to be interesting in a realistic case. Moreover the starting vector looses its importance the longer we iterate. For our future work we will concentrate on improving the starting vector and we will investigate how to add relevance feedback to the algorithm.

References

[1] M. W. Berry, S. Dumais, and G. W. O'Brien, *Using linear algebra for intelligent information retrieval*, SIAM Review, 37 (1995), pp. 573–595.

[2] K. Blom, *Information retrieval using the singular value decomposition and Krylov subspaces*, Tech. Rep. 1999-5, Dept. Mathematics, Chalmers University of Technology, Göteborg, 1999. ISSN 0347-2809.

[3] K. Blom,*Information Retrieval using Krylov subspace methods*, PhD Thesis, Chalmers Universtiy of Technology, Göteborg, 2004 ISBN 91-7291-453-X.

[4] K. Blom and A. Ruhe, *Information retrieval using very short Krylov sequences*, in Computational Information Retrieval, M. Berry, ed., vol. 106 of Proceedings in Applied Mathematics, SIAM, Philadelphia, 2001, pp. 41–56.

[5] I. S. Dhillon and D. S. Modha, *Concept decompositions for large sparse text data using clustering*, Machine Learning, 42 (2001), pp. 143–175.

[6] S. T. Dumais, *Latent semantic indexing (LSI): TREC-3 report.*, in D K Harman Editor, The third Text REtrieval Conference (TREC-3), NIST Special Publication 500-225, 1995, pp. 219–230.

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

[8] G. H. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM Journal on Numerical Analysis, 2 (1965), pp. 205–224.

[9] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Maryland, 3 ed., 1996.

[10] D. Harman, *The Eighth Text REtrieval Conference (TREC-8)*, NIST Special Publication 500-246. http://trec.nist.gov/pubs/trec8, (2000), p. A1 (Appendix).

[11] H. Park, M. Jeon, and J. B. Rosen, *Lower dimensional representation of text data in vector space based information retrieval*, in Computational Information Retrieval, M. Berry, ed., vol. 106 of Proceedings in Applied Mathematics, SIAM, Philadelphia, 2001, pp. 7–27.

[12] K. Sparck Jones, *Automatic Keyword Classification for Information Retrieval*, Butterworths, London, 1971.

[13] J. Xu and W. B. Croft, *Query expansion using local and global document analysis*, in Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1996, pp. 4–11.

[14] H. Zha, O. Marques, and H. D. Simon, *Large-scale SVD and subspace-based methods for information retrieval*, in Solving Irregularly Structured Problems in Parallel, A. Ferreira, J. Rolim, H.Simon, and S. Teng, eds., Springer LNCS 1457, 1998, pp. 29–42.

Modified Krylov subspace methods for information retrieval.

Katarina Blom

April 12, 2004

Abstract

This paper describes how simple modifications of the Krylov subspace method for IR can be used to steer what documents to retrieve and thus improve retrieval performance.

In our experiments retrieval performance, measured in average precision, is in general better for those queries that are at a smaller angle to their subspaces of relevant documents.

Improved query vectors are used directly in the vector model to rank documents for relevancy, and also for explicit restart of the bidiagonalization procedure in the Krylov subspace method.

The bidiagonalization process used in the Krylov subspace method is rewritten so that only directions orthogonal to subspaces spanned by irrelevant documents will be taken into account.

Starting the bidagonalization procedure with subspaces spanned by the terms in the queries, or a block of relevant retrieved documents will further improve the ranking of relevant documents. We replace the Golub-Kahan bidiagonalization procedure in the Krylov subspace method for IR with a band Lanczos procedure.

The modifications we make are based on relevance feedback and quite naturally our experiments show a significant increase in retrieval performance for the modified methods compared to the original Krylov subspace method used for IR.

Keywords
Information retrieval, Relevance feedback, Lanczos algorithm, band Lanczos algorithm, Vector space model, Krylov subspace, SVD, Singular value decomposition, query expansion, numerical linear algebra.

Contents

1 Introduction **7**
1.1 Measures 9
1.2 Notation 10
1.3 Mathematical background 11

2 Krylov subspace methods for information retrieval **12**

3 The subspace spanned by relevant document vectors **14**
3.1 Query vector and the relevant subspace 15
3.2 Projected query vectors 15
3.3 Optimal scoring 16

4 Using relevance feedback **17**
4.1 Formulating improved query vectors 17
4.1.1 Ranking documents 20
4.2 Explicit restart 21
4.2.1 Simple explicit restart 21
4.2.2 Modifying the bidiagonalization procedure 22
4.3 Query subspaces and the union of Krylov subspaces 23
4.3.1 Query subspace 23
4.3.2 The band Lanczos algorithm 24
4.3.3 The band Lanczos procedure for IR 25
4.4 Starting at scoring vector 27

5 Numerical experiments **27**

A The Golub-Kahan bidiagonalization procedure **39**

B The-Golub Kahan bidiagonalization procedure modified **41**

1 Introduction

An information retrieval (IR) system matches user queries (formal statements of information needs) to documents stored in a database. For each query entered the IR system will rank all the documents in relevance order. In *vector space models* queries and documents are encoded as vectors in m-dimensional space, where m is the number of unique terms in the collection. Documents are sorted for relevancy measuring the angles between each document vector and the query vector. The vector model has some major drawbacks. The terms used in the query vectors are often not the same as those by which the information searched has been indexed in the document vectors. In the vector model all document vectors that have no term in common with the query vector will be orthogonal to the query vector (and there by be ranked as irrelevant).

In Blom and Ruhe [7, 6] we discussed how a few iterations in the Golub-Kahan bidiagonalizing procedure could (at least in theory) overcome this problem and further improve the document ranking. In this paper we describe how simple modifications of the methods used in [6] can be used to steer what documents to retrieve and thus improve retrieval performance.

In IR *query expansion* is used to change the terms in the query vector in order to formulate a new improved query vector Harman [13], Ide [15], Rochio [17], Xu and Croft [20]. The original query is replaced with the expanded query and the documents are ranked again. The expected effect is that the new query is moved towards the relevant documents and away from irrelevant documents and there by improve the ranking.

In our experiments there is a relationship between the query vectors closeness in angles to their relevant subspaces (the subspaces spanned by column vectors corresponding to relevant documents) and retrieval performance in vector models. For query vectors close in angles to to their relevant subspaces, often retrieval performance (measured in average precision) is better than for queries further away from their relevant subspaces.

We discuss how simple projections of the original query vector q can be used to formulate improved query vectors. We use the retrieved relevant documents from the vector model to create new query vectors that (hopefully) are closer in angles to the relevant subspace.

For Krylov subspace methods *explicit restart* means replacing the starting vector q with an improved starting vector q^+ and restart the bidiagonalization procedure with this new vector. (In eigenvalue computations explicit restart

is often used to limit the sizes of the basis set). We use the retrieved relevant documents from the Krylov subspace method to construct new starting vectors for the explicit restart.

The two alternating steps of matrix-vector multiplications, $A^T q$ and Ap, in the bidiagonalization procedure in the Krylov method can be roughly interpreted as finding all documents containing the terms in the query q and finding all the terms contained in the documents represented by p. It is easy to see that after several iterations this process will bring in some relevant documents as well as many irrelevant ones. Controlling this growth process will improve retrieval efficiency for the Krylov subspace method.

Technically it is easy to rewrite the bidiagonalizing procedure to exclude unwanted (irrelevant) search directions. We use the retrieved irrelevant documents to construct subspaces spanned by unwanted directions to avoid in the bidiagonalizing procedure.

Starting the bidiagonalizing procedure with a block of relevant documents (relevant directions) instead of only one vector will improve performance significantly. This is done by replacing the Golub-Kahan bidiagonalizing procedure used in the Krylov subspace method for IR with a band Lanczos algorithm.

This article is organized as follows:

In section 1.1 we present measures for retrieval efficiency used in this article. The measures are standard in the IR community. The notation and some symbols frequently used throughout the article are listed in section 1.2. In section 1.3 we give a short mathematical background for some concepts further used.

Section 2 gives a short presentation of the Krylov method used for IR [6]. (The Golub-Kahan bidagonalization procedure used in the Krylov method is shortly presented in appendix A).

In section 3 we introduce the *relevant subspace* (the subspace spanned by relevant document vectors) and it's complement. We discuss how the query vectors may be projected onto the relevant subspace or orthogonal to the complement. We also introduce an *optimal scoring*.

In section 4 we introduce techniques for how to approximate the projected query vectors from section 3 and how to approximate the optimal scoring.

In section 4.2 improved query vectors are used for explicit restart of the bidiagonalization procedure. We also discuss how the bidiagonalization procedure can be modified to search only in directions orthogonal to unwanted directions.

Section 4.3 introduce *query subspaces* (subspaces spanned by the terms in the queries). A short description of the band Lanczos procedure and how it can be used for IR is given.

In section 5 some numerical experiments are presented.

1.1 Measures

The retrieval efficiency of an information retrieval system depends on two main factors. The ability of the system to retrieve relevant information and the ability to dismiss irrelevant information. The ability to retrieve relevant information is measured by *recall*, the ratio of relevant documents retrieved over the total number of relevant documents for that query. A systems ability to reject irrelevant documents is measured by *precision*, the ratio of the number of relevant documents retrieved for a given query over the total number of documents retrieved. Precision and recall are usually inversely related (when precision goes up, recall goes down and vice versa).

When we evaluate a query q, all the documents are ranked and we receive an ordered list $\mathcal{L}$ of documents. Assume t documents are relevant to the query and let $\ell_i, \quad i = 1 \ldots t$ be the position for the ith relevant document in $\mathcal{L}$. The *average precision* (non interpolated) for a single query is defined as

$$\frac{1}{t}\sum_{i=1}^{t}\frac{i}{\ell_i}.$$

The *mean average precision* for multiple queries is defined as the mean of the average precisions for all queries.

Precision can be computed at any *actual recall level*

$$\frac{i}{t}, \quad i = 1, 2, \ldots, t$$

(where t is the number of relevant documents to the query).

Let r_j be the jth recall level from the 11 *standard recall levels* $0, 0.1, 0.2 \ldots 1$. The *interpolated average precision* for a query at standard recall level r_j is the maximum precision obtained for any actual recall level greater that or equal to r_j.

The *Recall level precision averages* for multiple queries are the means of the interpolated average precision values at each (standard) recall level for

the queries. Recall level precision averages are used as input for plotting the recall-precision graphs.

For further details, see Harman [14].

1.2 Notation

The notation used in this article is rather standard in the numerical linear algebra community. We use uppercase letters for matrices and lowercase letters for vectors. Lowercase Greek letters usually denotes scalars. Component indices are denoted by subscript. For example, a vector c and a matrix M might have entries c_i and m_{ij} respectively. On the occasions when both an iteration index and a component index are needed, the iteration is indicated by a parenthesised superscript, as in $c_j^{(r)}$ to indicate the jth component of the rth vector in a sequence. Otherwise c_j may denote either the jth component of a vector c or the jth column of a matrix C. The particular meaning will be clear from its context.

The range of a matrix M is the subspace spanned by the columns of M and is denoted $R(M)$.

Some symbols are frequently used throughout the article. They are listed below. For a detailed description please see each reference.

A: Term document matrix (section 2).

q: Query vector (section 2).

$\mathcal{Q}$: The query subspace (section 4.3.1).

$\mathcal{A}$: We use $\mathcal{A}$ for $R(A)$, the range of A. It is spanned by the columns of A and has the dimension r, the rank of A.

$\mathcal{R}$, $\mathcal{C}$: The relevant subspace and the complementary subspace (section 3).

$\tilde{\mathcal{R}}$, $\tilde{\mathcal{C}}$: The subspace spanned by the relevant retrieved documents and its complement in the residual collection (section 4.1).

Some of the notations not listed here that are used throughout the article are introduced in section 1.3.

1.3 Mathematical background

Orthogonal projections Let M be any $m \times n$ matrix. The *orthogonal projection* of the column vectors in M onto the space $\mathcal{S}$ is denoted $\mathcal{P}_S M$. The column vectors in M may also be projected orthogonal to $\mathcal{S}$, $M - \mathcal{P}_S M$.

Principal angles The relative orientation of two subspaces can be described by *principal angles*, the angles formed by *principal vectors* in the spaces. Let $\mathcal{F}, \mathcal{G} \in R^m$ be two subspaces whos dimensions satisfy $p = dim(\mathcal{F}) \geq dim(\mathcal{G}) = s \geq 1$. The first principal angle θ_1 between $\mathcal{F}$ and $\mathcal{G}$ is the smallest angle that can be formed by a vector $f_1 \in \mathcal{F}$ and a vector $g_1 \in \mathcal{G}$. Since the angle is minimized when the cosine is maximized the smallest angle satisfies

$$\begin{aligned} \cos\theta_1 &= \max_{f\in\mathcal{F}} \max_{g\in\mathcal{G}} f^T g = f_1^T g_1 \\ \text{subject to} \quad & \|f\| = \|g\| = 1. \end{aligned}$$

The vectors f_1 and g_1 are called principal vectors. The second principal angle θ_2 is defined to be the smallest angle that can be formed between a vector in $\mathcal{F}$ that is orthogonal to f_1 and a vector in $\mathcal{G}$ that is orthogonal to g_1. The principal angles $\theta_1, \ldots, \theta_s$ between $\mathcal{F}$ and $\mathcal{G}$ are defined recursively by

$$\begin{aligned} \cos\theta_k &= \max_{f\in\mathcal{F}} \max_{g\in\mathcal{G}} f^T g = f_k^T g_k \\ \text{subject to} & \\ \|f\| &= \|g\| = 1, \\ f^T f_i &= 0, \ i = 1 \ldots k-1, \\ g^T g_i &= 0, \ i = 1 \ldots k-1. \end{aligned}$$

For further reading on principal angles see Watkins [19].

Let F be an $m \times p$ matrix and let G be an $m \times s$ matrix. Assume that both F and G have linearly independent columns and let $F = Q_F R_F$ and $G = Q_G R_G$ be the QR-decompositions of F and G respectively. Using the SVD the principal angles and vectors for the ranges $R(F)$ and $R(G)$ can be computed [4]. Let

$$USV^T = Q_F^T Q_G \tag{1}$$

be the singular value decomposition of $Q_F^T Q_G$. The cosines of the principal angles are the singular values in S and the principal vectors for F and G are $Q_F U$ and $Q_G V$ respectively.

Krylov subspaces The *Krylov subspace* $\mathcal{K}_r(B, x)$ of the square matrix B and starting vector x is spanned by the r vectors

$$x, Bx, B^2x, \ldots, B^{r-1}x$$

where x is any nonzero starting vector. The *block Krylov subspace* $\mathcal{K}_r(B, X)$ is spanned by the pr vectors in the block Krylov sequence

$$X, BX, B^2X, \ldots, B^{r-1}X$$

where the columns in $X = \begin{bmatrix} x_1 & x_2 & \ldots & x_p \end{bmatrix}$ are linearly independent.

Theorem Let vectors $x_1, x_2, \ldots x_p$ be orthonormal and starting vectors for the p sequences spanning the Krylov subspaces $\mathcal{K}_r(B, x_1)$, $\mathcal{K}_r(B, x_2)$, $\ldots$, $\mathcal{K}_r(B, x_p)$ respectively.

Then the block Krylov subspace is the union of the p Krylov subspaces,

$$\mathcal{K}_r(B, X) = \bigcup_{j=1}^{p} \mathcal{K}_r(B, x_j)$$

proof The proof follows from a simple permutation of the vectors that span the block Krylov subspace, we have

$$\begin{aligned} \mathcal{K}_r(B, X) &= \text{span}\{x_1, x_2, \ldots, x_p, Bx_1, Bx_2, \ldots, Bx_p, \ldots, B^{r-1}x_1, B^{r-1}x_2, \ldots, B^{r-1}x_p\} \\ &= \text{span}\{x_1, Bx_1, \ldots, B^{r-1}x_1, x_2, Bx_2, \ldots, B^{r-1}x_2, \ldots, x_p, Bx_p, \ldots, B^{r-1}x_p\} \\ &= \bigcup_{j=1}^{p} \mathcal{K}_r(B, x_j). \end{aligned}$$

2 Krylov subspace methods for information retrieval

In vector space models both queries and documents are encoded as vectors in m-dimensional space, where m is the number of unique terms in the collection. The document vectors are stored as columns in an $m \times n$ term document matrix A. The query vectors are stored as $m \times 1$ vectors q and query matching can be viewed as a search in the column space of the term document matrix A.

In the *vector model* the documents are scored measuring the cosine of the angles between the query vector q and each document vector a_j in A,

$$c_j = \frac{q^T a_j}{\|q\|_2 \|a_j\|_2}, \quad j = 1, 2, \ldots, n. \tag{2}$$

The smaller the angle (i.e. the larger cosine value) the higher relevance score.

For the *Krylov subspace methods* we will use the Golub Kahan bidiagonalization procedure [12] applied to the term document matrix A starting at the query vector q to compute the two *basis matrices* Q_{r+1} and P_r and the $(r+1) \times r$ lower bidiagonal matrix $B_{r+1,r}$ satisfying

$$B_{r+1,r} = Q_{r+1}^T A P_r \tag{3}$$

The column vectors in the basis matrices Q_{r+1} and P_r span bases for the two Krylov subspaces $\mathcal{K}_{r+1}(AA^T, q)$, in the document space (spanned by the query q and the columns of A) and $\mathcal{K}_r(A^T A, A^T q)$, in the term space (spanned by the rows of A) respectively. The Golub-Kahan bidiagonalization procedure is further described in appendix A.

We let W form an orthonormal basis for the column vectors in the *reached subspace* $\text{span}(AP_r)$.

The reached subspace W, the basis matrices Q_{r+1}, P_r and the $B_{r+1,r}$ matrix are used to score the documents in relevance order to the query (see Blom Ruhe [6]). The similarity measures we use for this article are:

- In the *subspace projection measure* the documents in A are sorted according to their closeness measured in angles to the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$. The relevance is measured using

$$c_j = \|Q_{r+1}^T a_j\|_2, \quad j = 1, 2, \ldots, n. \tag{4}$$

- A projected query vector

$$\hat{q} = WW^T q \tag{5}$$

  is constructed using the reached subspace. In the *expanded query measure* the documents are scored using the angles between the expanded query vector $\hat{q}$ and each document vector in A,

$$c_j = \frac{\hat{q}^T a_j}{\|\hat{q}\|_2 \|a_j\|_2}, \quad j = 1, 2, \ldots, n. \tag{6}$$

Note that if the starting vector $q \in \mathcal{A}$, the range of A, then the projected query $\hat{q} = q$ and the cosines (6) are simply the vector model scoring (2).

3 The subspace spanned by relevant document vectors

When a user issues a search for information on a topic, the information retrieval system will start to return 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.

Let $\mathcal{A}$ be the subspace spanned by the column vectors in the term document matrix A and assume r documents are relevant to query q. The subspace

$$\mathcal{R} \subseteq \mathcal{A} \tag{7}$$

spanned by the document vectors in A that correspond to the r relevant documents we call the *relevant subspace*. We define the *complementary subspace* by the set difference between the range of A and the relevant subspace,

$$\mathcal{C} = \mathcal{A} - \mathcal{R}. \tag{8}$$

It is important to note that with this definition of the relevant subspace the relevant document vectors are completely in $\mathcal{R}$. The irrelevant document vectors are spanned by vectors both in the relevant subspace and in the complementary subspace.

The relevant subspace $\mathcal{R}$ for a query q is *discernible* if no irrelevant document vectors are completely in $\mathcal{R}$. For all sets we have studied[1] the relevant subspaces for all queries are discernible[2].

[1] The Adi, Cici, Cranfield and Medline data sets [9] (see also [2]) and the Financial Times and Congressional Records from the TREC data sets [1].

[2] How the document vectors in the term document matrices were set up is described in section 5.

3.1 Query vector and the relevant subspace

The query vector q can be divided into three orthogonal parts q_1, q_2 and q_3 where q_1 is in the relevant subspace (7) of q, q_2 is in $\mathcal{A}$, the range of A but not in the relevant subspace and q_3 is orthogonal to $\mathcal{A}$.

$$q = q_1 + q_2 + q_3 \tag{9}$$

and $\|q\|_2^2 = \|q_1\|_2^2 + \|q_2\|_2^2 + \|q_3\|_2^2$. In our experiments $\|q_3\|_2$ is in general larger than $\|q_1\|_2$ and $\|q_2\|_2$, and $\|q_2\|_2$ is in general larger than $\|q_1\|_2$.

It is not always the case that the query vector is closer to the subspace of relevant documents than the irrelevant document vectors are. Quite often we will find irrelevant document vectors making a smaller angle to the relevant subspace than the query vector itself.

This is clearly seen in figure 1 where cosines of the angles between the relevant subspace $\mathcal{R}$ and each document vector and the query vector are plotted. 15 irrelevant document vectors make slightly smaller angles to the relevant subspace than the query vector.

We have also found several query vectors being orthogonal to their relevant subspaces.

There is a tendency that average precision (using any of the ranking algorithms presented in this article) is better for queries (or expanded queries) close to their relevant subspaces (7) than for queries further away from their relevant subspaces.

When sorting the retrieval performance (measured in average precision for the vector model (2)) for each query vector q within a data set according to $\|q_1\|_2$ there is a relationship (see figure 2 (right plot))[3]. Average precisions for query vectors orthogonal to or with a small part in their relevant subspaces is very moderate. Average precisions for query vectors closer in angles to their relevant subspaces tend to be higher.

3.2 Projected query vectors

We can move the query vector q away from irrelevant documents (measured in angles) by projecting it orthogonal to the complementary subspace $\mathcal{C}$ (8)

$$q - \mathcal{P}_{\mathcal{C}} q = q_1 + q_3 \tag{10}$$

[3]Similar relationships seems to occur when sorting the retrieval performance for each query q according to $\frac{\|q_1\|_2}{\|q_2\|_2}$.

where q_1 and q_3 are the parts of q that are in the relevant subspace for q and orthogonal to the range of A respectively as defined in (9). We can move the query vector towards the relevant subspace by projecting it onto the relevant subspace $\mathcal{R}$ (7)

$$\mathcal{P}_{\mathcal{R}} q = q_1. \tag{11}$$

Clearly both the projected queries (10) (11) are orthogonal to the complementary subspace $\mathcal{C}$. Unless q is completely in the range of A the projected query vectors (10) (11) are not equal[4].

3.3 Optimal scoring

An *optimal scoring* will rank all relevant documents better than irrelevant documents.

Assume the relevant documents for a query span a discernible subspace. Since $\mathcal{P}_{\mathcal{R}} a_j = a_j$ for all relevant document vectors a_j, then optimality is obtained if all documents are sorted in descending order according to their angles to the relevant subspace

$$\frac{\|\mathcal{P}_{\mathcal{R}} a_j\|_2}{\|a_j\|_2} = \frac{\|a_j - \mathcal{P}_{\mathcal{C}} a_j\|_2}{\|a_j\|_2}, \quad j = 1, 2, \ldots, n. \tag{12}$$

The angles between the projected query vector $\mathcal{P}_{\mathcal{R}} q$ and each document vector a_j in the term document matrix

$$\frac{(\mathcal{P}_{\mathcal{R}} q)^T a_j}{\|\mathcal{P}_{\mathcal{R}} q\|_2 \|a_j\|_2} = \frac{(\mathcal{P}_{\mathcal{R}} a_j)^T q}{\|\mathcal{P}_{\mathcal{R}} q\|_2 \|a_j\|_2} = \frac{(a_j - \mathcal{P}_{\mathcal{C}} a_j)^T q}{\|\mathcal{P}_{\mathcal{R}} q\|_2 \|a_j\|_2}, \quad j = 1, 2, \ldots, n \tag{13}$$

does not necessarily give an optimal scoring.

This is easily verified. For relevant document vectors a_j, the angle between the projected document $\mathcal{P}_{\mathcal{R}} a_j$ and the query vector q is the same as the angle between the document vector itself and the query vector, [5]

$$\frac{(\mathcal{P}_{\mathcal{R}} a_j)^T q}{\|a_j\|_2} = \frac{a_j^T q}{\|a_j\|_2}, \quad j = 1, 2, \ldots r.$$

[4]For the data sets we have studied, (the Adi, Cici, Cranfield and Medline data sets [9], the FT and the CR sets from TREC [1]), q is never completely in $\mathcal{A}$.

[5]Since $\|\mathcal{P}_{\mathcal{R}} q\|_2$ in the denominator is constant for all $j = 1, 2, \ldots, n$ it will not affect the scoring and can be omitted.

We cannot gurantee that this quantity, $\frac{(\mathcal{P}_{\mathcal{R}} a_j)^T q}{\|a_j\|_2}$, is smaller than the angles between the projected irrelevant document vectors and the query vector. Some of the irrelevant documents may be ranked higher than relevant documents.

Clearly the scoring (13) is not optimal. In experiments this scoring (not surprisingly) performs very well measured in average precision.

However, the documents forming the relevant subspace for a query are not known in advance (in fact we are trying to find them). In reality we can only compute approximations to the projected query vectors (10), (11) and rankings (12), (13).

4 Using relevance feedback

Both the vector model and the Krylov subspace method have a limited recall. Usually some relevant documents are retrieved to a query, but almost never all the relevant documents. In this section we will discuss some techniques that may be used together with the Krylov method to further increase the recall (mostly the techniques will also increase precision).

To steer the process of what documents to retrieve, we will use relevance feedback. In a relevance feedback cycle, the user is presented a list of retrieved documents, and after examining them, marks those that are relevant. The main idea is to use the information provided by the user to make a new (hopefully) improved search[6].

4.1 Formulating improved query vectors

Assuming that relevant documents resemble each other it is natural to formulate an initial query and to incrementally compute vectors of the relevant and/or the complementary subspaces.

Assume that t retrieved documents have been returned back to the user at some point. Assume that s of these were identified as relevant and $t-s$ were irrelevant. Let the columns of A_t correspond to the t retrieved documents.

[6]Relevance feedback can also be performed without involving a user. In *pseudo relevance feedback* new queries are constructed using the top retrieved documents, see for example Xu and Croft [20].

Let

$$\tilde{\mathcal{R}} \subseteq R(A_t) \tag{14}$$

be the subspace spanning the s retrieved relevant document vectors and let

$$\tilde{\mathcal{C}} = R(A_t) - \tilde{\mathcal{R}} \tag{15}$$

be it's complement in the range of A_t.

Clearly the subspace spanned by the relevant retrieved document vectors is in the relevant subspace (7), $\tilde{\mathcal{R}} \subseteq \mathcal{R}$. The complement $\tilde{\mathcal{C}}$ to $\tilde{\mathcal{R}}$ in $R(A_t)$ may not be completely in the complementary subspace $\mathcal{C}$ (8)

We will mimic the two projected query vectors (10) and (11) with the two *expanded query vectors*

$$q^- = q - \mathcal{P}_{\tilde{\mathcal{C}}} q \quad \text{and} \quad q^+ = \mathcal{P}_{\tilde{\mathcal{R}}} q \tag{16}$$

respectively. Note that q^+ is in $\mathcal{A}$, the range of A, while q^- might not be completely in $\mathcal{A}$. It is possible to add tuning constants to the expanded query vector q^-. Letting

$$q^- = \alpha q + \beta \mathcal{P}_{\tilde{\mathcal{C}}} q \tag{17}$$

and with appropriate choices of α and β retrieval performance for the expanded query vector q^- may improve.

If no relevant documents were found among the t best scored documents then it is not possible to form the expanded query vector q^+. One option could then be to look further down the ranked list of retrieved documents in order to find some relevant documents. Often this is wasted effort, the vector model will pull in too many irrelevant documents and there is an upper limit to how many documents we can expect a human user to judge for relevance. Forming the other expanded query vector q^- is also likely to fail since the query vector q will be projected orthogonal to the subspace $\tilde{\mathcal{C}}$. The number of retrieved documents t is then much less than the number of document vectors n in A and the subspace $\tilde{\mathcal{C}}$ will be a very poor approximation of the complementary subspace $\mathcal{C}$ (8).

There are three classic (and rather similar) ways to calculate an improved query vector for vector models [10].

In *standard Rochio* [17] the new query vector is computed using the original query vector q and the retrieved relevant document vectors in A_s and the retrieved irrelevant document vectors in A_{t-s}

$$q_{\text{ROCHIO}} = \alpha q + \frac{\beta}{s} A_s e_s - \frac{\gamma}{t} A_{t-s} e_{t-s}$$

where e_s and e_{t-s} are the vectors of ones and α, β and γ are used as tuning constants. Sometimes γ is set to 0. For the two other methods, *Ide Regular* and *Ide dec hi* [15] other tuning constants are used and for the Ide dec hi method the highest ranked irrelevant document vector is subtracted instead of the sum of the irrelevant document vectors.

The last vector $A_{t-s}e_{t-s}$ in the standard Rochio expansion can be divided in two parts, one part that is in the subspace spanned by the relevant retrieved documents $\tilde{\mathcal{R}}$ (14) and one part that is in the complement $\tilde{\mathcal{C}}$ (15).

The standard Rochio expansion can be formulated

$$q_{\text{ROCHIO}} = \alpha q + \mathcal{P}_{\tilde{\mathcal{R}}}(\frac{\beta}{s}A_s e_s - \frac{\gamma}{t}A_{t-s}e_{t-s}) - \frac{\gamma}{t}\mathcal{P}_{\tilde{\mathcal{C}}}A_{t-s}e_{t-s}. \tag{18}$$

Let the *residual collection*

$$A_{n-t} \tag{19}$$

be the columns of A with the t document vectors that correspond to the retrieved documents removed.

We cannot formally prove that any of the three expanded query vectors (16) – (18) are closer in angles to their relevant subspaces in the residual collection than the query vectors q. In our experiments[7] some trends are clear though. In general q^+ and q_{ROCHIO} are moved towards their relevant subspaces in the residual collection. For $\alpha = \beta = 1$ the expanded query vectors q^- are moved away from the irrelevant document vectors but rarely towards their relevant subspaces in the residual collection. If an expanded query vector is moved towards its relevant subspace in the residual collection retrieval performance, using any of the ranking algorithms presented in this article, is in general better than performance for the original query vector q.

In figure 1 we used the ten best scored documents from the vector model (2) and constructed the expanded query vector q^+ (16). The expanded query vector q^+ is not completely in the subspace spanned by the relevant documents in the residual collection, however the expanded query vector is closest in angle to the relevant subspace in the residual collection compared to all document vectors in the residual collection.

[7] with the Adi, Cici, Cranfield and Medline data sets [9] (see also [2]) and the Financial Times and Congressional Records from the TREC data sets [1].

4.1.1 Ranking documents

When using relevance feedback only the document vectors in the residual collection A_{n-t} (19), where the t removed document vectors correspond to the t documents used for the feedback cycle, will be scored for relevance.

In order to mimic the optimal scoring (12) we have sorted the documents in relevance order using any of the measures

$$\frac{\|\mathcal{P}_{\tilde{\mathcal{R}}}a_j\|_2}{\|a_j\|_2}, \quad j = 1, 2, \ldots, n-t \tag{20}$$

and

$$\frac{\|a_j - \mathcal{P}_{\tilde{\mathcal{C}}}a_j\|_2}{\|a_j\|_2}, \quad j = 1, 2, \ldots, n-t \tag{21}$$

respectively, where a_j is in the residual collection A_{n-t}.

In the vector model (2) documents are scored measuring angles between the query vector q and each document vector in A. It is natural to score documents measuring angles between the expanded query vectors and each document vector in the residual collection.

$$\frac{(q^+)^T a_j}{\|a_j\|_2}, \quad j = 1, 2, \ldots, n-t, \tag{22}$$

$$\frac{(q^-)^T a_j}{\|a_j\|_2}, \quad j = 1, 2, \ldots, n-t \tag{23}$$

and

$$\frac{(q_{\text{ROCHIO}})^T a_j}{\|a_j\|_2}, \quad j = 1, 2, \ldots, n-t \tag{24}$$

respectively. The vectors (22) – (24) are sorted in descending order.

Note that the three scorings (22) – (24) in general are not equal.

In average, performance is better for all three rankings (22) – (24) using the expanded query vectors compared to the vector model (2).

In figure 1 forming the expanded query vector q^+ we are able to capture all four remaining relevant documents only by considering in total 30 irrelevant documents (compared to the vector model where we had to consider in total 107 irrelevant documents).

The scorings (22) – (24) does not improve average precision for all queries compared to the vector model. Sometimes it is better to use the original

query vector when scoring documents for relevance. In figure 4 the scorings (22) – (24) are compared with the vector model for documents in the residual collection. A few relationships can be noticed. Queries loosing in performance for q^+ also looses in q_{ROCHIO}. Queries where performance improvement is large for q^+ is also large for q_{ROCHIO}.

Retrieval performances for the scorings (20) – (24) and the vector model (2) are compared in figure 5. In average, performance is better for all three expanded query vectors compared to the vector model. Precision is much better but also a small improvement in recall can be seen. The expanded query q^+ gives largest improvement while the q^- score documents rather similar to the original vector model. With other tuning constants for the Rochio queries or the expanded query vector q^- performance might improve. The approximate optimal scoring (20) using the relevant retrieved subspace (14) is also good, while the approximated scoring (21) using the complement (15) is not very effective.

4.2 Explicit restart

Explicit restart means replacing the starting vector q with an improved starting vector and restart the bidiagonalization procedure with this new vector[8]. In eigenvalue computations explicit restart is often used to limit the sizes of the basis set. In the context of IR we want to restart the bidiagonalization procedure with a vector that better captures the connections between the groups of relevant documents. The basic idea is to start the bidiagonalization procedure with the original query vector q and rank the documents using any of the rankings in section 2. Based on relevance feedback information from the user, new improved starting vectors for the bidiagonalization procedure are constructed.

4.2.1 Simple explicit restart

Figure 6 is a recall-precision graph. We constructed expanded query vectors q^+, q^- and q_{ROCHIO} from section 4.1 using relevance feedback. For each query the user judged the 10 top ranked documents from the expanded query measure (6) (using the original query vector q as starting vector in the bidiagonalizing procedure). The bidiagonalization procedures were restarted and

[8]The bidiagonalization procedure is further described in [6]. A summary can be found in appendix A

the documents in the residual collections were ranked using the subspace projection measure (4). Retrieval performances for the different starting vectors are compared in the figure. In general performance is improved by explicit restart.

Even though we measure performance only for the documents in the residual collection, it is important to keep the relevant retrieved documents in the term document matrix when bidiagonalizing. Otherwise performance will decrease[9]. Quite often retrieval performance is further increased by removing the irrelevant retrieved documents from the term document matrix before bidiagonalizing.

4.2.2 Modifying the bidiagonalization procedure

The retrieval performance can be further increased by using the feedback information also when bidiagonalizing. Assume the subspace $\mathcal{E}$ span directions we want to avoid. Starting the bidiagonalizing procedure with any vector orthogonal to $\mathcal{E}$ will start a search orthogonal to the unwanted directions, but it is not enough to guarantee the orthogonality between the basis vectors in Q_{r+1} (3) and $\mathcal{E}$. Technically it is easy to rewrite the BIDIAG procedure to incrementally compute vectors $q_i \in Q_{r+1}$ orthogonal to $\mathcal{E}$, and thus completely avoid all directions in the subspace while bidagonalizing (further details are in appendix B).

Start with $q_1 = \frac{q - \mathcal{P}_{\mathcal{E}} q}{\|q - \mathcal{P}_{\mathcal{E}} q\|_2}$, $\beta_1 = 0$
for $k = 1, 2, \ldots r$ **do**
  $\alpha_k p_k = A^T q_k - \beta_k p_{k-1}$
  $y = A p_k - \alpha_k q_k$
  $\beta_{k+1} q_{k+1} = y - \mathcal{P}_{\mathcal{E}} y$
end.

The vectors spanning $\mathcal{E}$ need to be chosen with some care. In order not to increment the computational load too much the number of vectors spanning $\mathcal{E}$ needs to be rather moderate. There must be document vectors in A that are not completely in $\mathcal{E}$ otherwise the procedure will vanish.

[9]In order to keep the effect of relevant documents resembling each other in the bidiagonalization procedure it seems to be important that all the relevant document vectors are kept otherwise the resembling effect will be to weak.

$\mathcal{E}$ could for example be chosen to span irrelevant retrieved directions, $\mathcal{E} = \tilde{\mathcal{C}}$ (15). Another option is to let $\mathcal{E}$ be spanned by one vector containing all terms that are in the irrelevant retrieved documents but not in the retrieved relevant. This procedure could also be useful for boolean queries.

4.3 Query subspaces and the union of Krylov subspaces

In this section we will consider query subspaces instead of query vectors. Instead of using the Golub Kahan bidiagonalizing procedure (3) we will use the band Lanczos procedure starting with the query subspace (or subspaces spanned by document vectors corresponding to relevant retrieved documents).

4.3.1 Query subspace

One way to broaden the query is to use query expansion, where more terms are added to the query vector to make it broader. Another way to broaden the query is to let the terms in the query span a subspace $\mathcal{Q}$.

Assume the query vector q consist of s terms, then we let the *query subspace* $\mathcal{Q}$ be spanned by s $m \times 1$ vectors, each vector with one nonzero element corresponding to a term in the query.

Any vector that can be expressed as a linear combination of terms in the query belongs to the query subspace $\mathcal{Q}$, in particular the query vector $q \in \mathcal{Q}$.

As for the query vector (9), the query subspace can be divided into three orthogonal subspaces, $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{Q}_3$ where $\mathcal{Q}_1$ is in the relevant subspace (7), $\mathcal{Q}_2$ is in the range of A, $\mathcal{A}$ but not in the relevant subspace and $\mathcal{Q}_3$ is orthogonal to $\mathcal{A}$.

The query subspace can be projected onto the subspace spanned by relevant retrieved document vectors (14) $\mathcal{Q}^+ = \mathcal{P}_{\bar{\mathcal{R}}}\mathcal{Q}$ or orthogonal to the complement (15) $\mathcal{Q}^- = \mathcal{Q} - \mathcal{P}_{\tilde{\mathcal{C}}}\mathcal{Q}$

We may score the documents for relevancy measuring the angles between the query subspace and each vector in the term document matrix. The cosines of the angle between the query subspace and each document vector in A

$$\|\mathcal{P}_{\mathcal{Q}} a_j\|_2$$

are sorted in decreasing order, and the documents corresponding to the larger cosines are ranked high.

We may also mimic the scorings (22) (23) from section 4.1.1 and score the documents according to the closeness in angles to the projected subspaces respectively

$$\|\mathcal{P}_{\mathcal{Q}^+} a_j\|_2$$

and

$$\|\mathcal{P}_{\mathcal{Q}^-} a_j\|_2.$$

4.3.2 The band Lanczos algorithm

The band Lanczos algorithm [18] (see also [3]) is based on block Krylov subspaces induced by a square matrix B and a block of s linearly independent starting vectors

$$y_1, y_2, \ldots, y_s. \tag{25}$$

The band Lanczos algorithm constructs orthonormal vectors that form a basis for the subspace spanned by the first linearly independent vectors of the block Krylov sequence

$$y_1, y_2, \ldots, y_s, By_1, By_2, \ldots, By_s, B^2y_1, B^2y_2 \ldots$$

If we apply the band Lanczos algorithm to the matrix

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

where A is the $m \times n$ term document matrix, with the starting block

$$Q_s = \begin{bmatrix} q_1 & q_2 & \ldots & q_s \\ 0 & 0 & \ldots & 0 \end{bmatrix} \tag{26}$$

with orthonormal columns that is a basis of the subspace spanned by $y_1, y_2, \ldots, y_s$, it reduces to the BANDL procedure below.

Define $h_{ji} = 0$ when $i < 1$. The following procedure compute the basis matrices Q_{r+s} and P_r and the $(r+s) \times r$ lower $(s+1)$ lower diagonal matrix $H_{r+s,r}$ satisfying

$$H_{r+s,r} = Q_{r+s}^T A P_r. \tag{27}$$

ALGORITHM BANDL(A, Q_s, r)
Start with s orthonormal vectors forming $Q_s = \begin{bmatrix} q_1 & q_2 & \dots & q_s \end{bmatrix}$.
for $j = 1$ **to** r **do**
  $h_{jj}p_j = A^T q_j - p_{j-s}h_{j,j-s} - p_{j-s+1}h_{j,j-s+1} - \dots - p_{j-s}h_{j,j-s}$
  $w = Ap_j - q_i h_{jj}$
  for $i = j+1$ **to** $j+s-1$ **do**
    $h_{ij} = q_i^T w; w = w - q_i h_{ij}$
  end
  $h_{j+s,j}q_{j+s} = w$
end

With $H_{r+s,r} = \begin{bmatrix} h_{ij} \end{bmatrix}$, the matrix $H_{r+s,r}$ is of size $r+s \times r$ and lower (s+1)-diagonal. In the first part, where computing p_j, previously computed subdiagonal elements or H are used while h_{jj} is computed to give p_j unit norm. In the second part, where computing q_{j+s}, the subdiagonal elements of H are computed as Gram Schmidt orthogonalization coefficients.

Define $Q_{r+s} = \begin{bmatrix} q_1 & q_2 & \dots & q_{r+s} \end{bmatrix}$ and $P_r = \begin{bmatrix} p_1 & p_2 & \dots & p_r \end{bmatrix}$. In exact arithmetic we will have $Q_{r+s}^T Q_{r+s} = I$ and $P_r^T P_r = I$. After $r+s$ iterations the basic relations

$$\begin{aligned} A^T Q_r &= P_r H_{rr}^T \\ AP_r &= Q_{r+s} H_{r+s,r} \end{aligned}$$

will hold. The columns of Q_{r+s} will be an orthonormal basis of the block Krylov subspace $\mathcal{K}_{r+1}(AA^T, Q_s)$ in the document space, spanned by the starting block Q_s and the columns of A.

The columns of P_r similarly span a basis of the block Krylov subspace $\mathcal{K}_r(A^T A, A^T Q_s)$ in the term space spanned by the rows of A. The singular values of $H_{r+s,r}$ will be approximations to those of A.

As for the original bidiagonalizing procedure from section 2 it is easy to rewrite the BANDL procedure to incrementally compute vectors $q_i \in Q_{r+s}$ orthogonal to a subspace $\mathcal{E}$ (section 4.2.2)

4.3.3 The band Lanczos procedure for IR

The BANDL applied to the term document matrix A gives us an opportunity to start with a block of (orthonormal) vectors spanning relevant information.

Let columns in the $m \times s$ matrix Q_s be orthonormal forming a starting block. We apply the BANDL algorithm to the term document matrix A starting at Q_s to receive the two basis matrices Q_{r+s} and P_r and the $r+s \times r$ lower (s+1)-diagonal matrix $H_{r+s,r}$. We let W form an orthonormal basis for the reached subspace spanned by the column vectors in AP_r.

As for the bidiagonalization procedure (3) the reached subspace W, the basis matrices Q_{r+s}, P_r and the $H_{r+s,r}$ matrix are used to score documents in relevance order to the query. A few examples were presented in section 2.

- In the *block subspace projection measure* the documents are sorted according to their closeness measured in angles to the block Krylov subspace $\mathcal{K}_{r+1}(AA^T, Q_s)$. The closer the document is, the more relevant.

$$c_j = \|Q_{r+s}^T a_j\|, \quad j = 1 \dots n. \tag{28}$$

  Sorting the documents according to Q_{r+s} in general give a better scoring than sorting the documents according to Q_{r+1} in the subspace projection measure (4) from section 2.

- For the *block expanded query measure* a projected query vector $\hat{q} = WW^T q$ is constructed using the reached subspace. The documents are sorted using the cosine scoring between $\hat{q}$ and each document vector in A.

$$c_j = \frac{\hat{q}^T a_j}{\|a_j\|}, \quad j = 1 \dots n. \tag{29}$$

  The larger the cosine value the more relevant document.

A few relations should be observed:

If we let the s vectors in the starting block Q_s span be the relevant retrieved subspace $\tilde{\mathcal{R}}$ (14) and stop the iterations in the BANDL procedure when $r = s$, then the block subspace projection measure (28) is the approximate optimal scoring (20) and the block cosine scoring (29) is the cosine of the angle between the expanded query q^+ from section 4.1 and each document vector (22).

If the starting block only consists of the query vector q then the BANDL procedure reduces to the BIDIAG procedure in section 2, otherwise letting $Q_s = \begin{bmatrix} q_1 & q_2 & \dots & q_s \end{bmatrix}$ and using theorem in section 1.3 column vectors in Q_{r+s} and P_r from the BANDL procedure form the union of the s Krylov subspaces $\mathcal{K}_r(AA^T, q_j)$ and $\mathcal{K}_r(A^T A, A^T q_j)$, $j = 1 \dots s$ respectively.

Let the column vectors in Q_s span the relevant retrieved subspace $\tilde{\mathcal{R}}$ (14) The Krylov subspace $\mathcal{K}_r(AA^T, q^+)$ received when using the bidiagonalization procedure from section 2 with A and the expanded starting vector q^+ (16) is a subspace of the block Krylov sequence $\mathcal{K}_{r+s}(AA^T, Q_s)$ received when using the BANDL process with A and the relevant retrieved directions spanned by Q_s, thus we will have $Q_{r+1} \subset Q_{r+s}$, where Q_{r+1} is the basis matrix (3) from the BIDIAG procedure and the Q_{r+s} is the basis matrix (27) from the BANDL procedure.

Used properly the BANDL procedure performs very well. Figure 7 is a recall-precision graph for the block expanded query measure.

4.4 Starting at scoring vector

Sometimes we start the bidiagonalization procedure with a scoring vector p, a weighted combination of documents. Then it is natural to reduce the matrix A to *upper* bidiagonal form by computing orthonormal bases for the Krylov subspaces $\mathcal{K}_r(A^TA, p)$ and $\mathcal{K}_{r+1}(AA^T, Ap)$, using the bidiagonalization procedure

$$\begin{aligned} \rho_k u_k &= Ap_k - \theta_k u_{k-1} \\ \theta_{k+1} p_{k+1} &= A^T u_k - \rho_k p_k \end{aligned}$$

with $k = 1, 2 \ldots r$, $p_1 = p/\|p\|_2$ and $\theta_1 = 0$. If we start the iteration with $p = A^T q$ this bidiagonalizing procedure can be derived from the BIDIAG procedure discussed in section 2. The relationships between the bidiagonalizations are discussed by Paige and Saunders [16] and also by Golub [11].

5 Numerical experiments

We present our experiments using the Cranfield collection, however the results are general and valid for other sets as well[10]. The overall retrieval performance varies between the sets. For the Medline set retrieval performance is very good while performance for the FT set is more moderate.

[10]Similar experiments were performed using the Adi, Cici, Cranfield and Medline data sets [9] (see also [2]) and the Financial Times and Congressional Records from the TREC data sets [1].

Cranfield is a small collection (1400 documents) with a large number of queries (225 queries). The data set consist of document abstracts in aerodymancis originally used for tests at the Cranfield Institute of Technology in Bedford, England.

We choose to report our experiments with a sequence of figures with appropriate captions.

Preparing the term document matrix We have used a simple term frequency weighting to construct the term document matrix

$$A = [a_{ij}] = \begin{cases} 0 & \text{if term } i \text{ not present in document } j \\ t_{ij} & \text{if term } i \text{ is present in document } j. \end{cases} \tag{30}$$

where t_{ij} is the number of times term i appears in document j. We use one row normalization followed by one column normalization in order to deemphasize common terms and long documents[11]. All rows corresponding to terms appearing in more than 10% of the documents were removed. For a further discussion about weightings for the Krylov subspace method please see [5].

In experiments where relevance feedback is used one initial run is made and the user is shown the top 10 documents. These documents are then used for relevance feedback purposes. Queries where no relevant documents were found among the top 10 or all relevant documents were among the top 10 were removed.

For evaluation measures the *residual collection method* is used. The evaluation of the results compares only to the residual collection A_{n-10}, that is all documents except the ten previously shown to the user are ranked and evaluated. The residual collection method provides an unbiased and realistic evaluation of feedback. However, because highly ranked relevant documents have been removed from the residual collection, there is a risk that the recall-precision figures will be lower than those for standard evaluation methods, and cannot directly be compared.

Relevance is always judged by comparing the results of an algorithm to relevance judgments provided with the test sets. These have been compiled by a panel of human experts who have considered at least all those documents marked as relevant.

[11]The column normalization will destroy the previous row normalization but not completely. Some deemphasizing effect of common terms still remain.

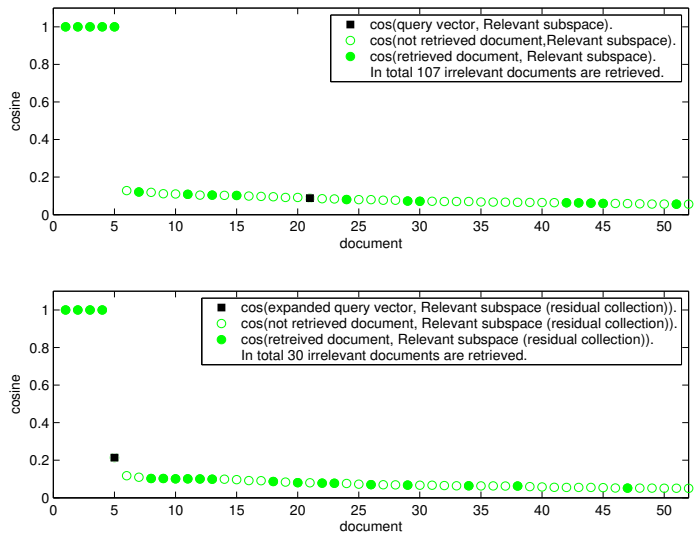

Figure 1: **Upper plot.** Cosines of the angles between the relevant subspace (7) and each document vector and the query vector for query no. 5 from the Cranfield set (only the 50 largest cosines are shown). In order to retrieve all five relevant documents using the vector model(2) 107 irrelevant documents were returned. **Lower plot.** Cosines of the angles between the relevant subspaces in the residual collection (19) and each document vector and the expanded query vector q^+ (16) for query no. 5 from the Cranfield set. (Only the 50 largest are shown). The expanded query vector q^+ was formed using the ten best scored documents from the vector model (2). In order to retrieve all four relevant documents in the residual collection measuring the angles between the expanded query and each document vector in the residual collection 30 irrelevant documents were returned. **In each plot** the cosine for the query vector is marked with a black square. The cosines corresponding to the documents are marked with circles and the documents retrieved in order to capture all the relevant documents are marked with filled circles. Query no. 5 has five relevant documents. Using the vector model one relevant document is scored among the top 10. The query is typical in the sense that in order to retrieve all the relevant documents using the vector model many irrelevant documents are retrieved (making precision rather moderate)

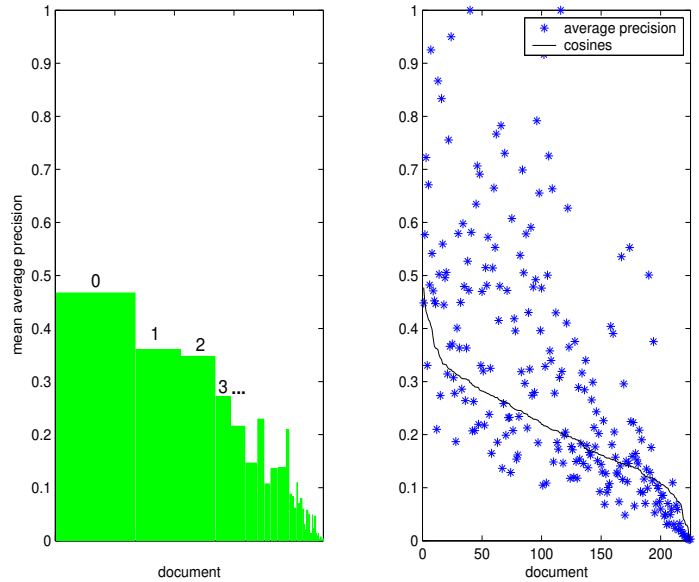

Figure 2: **Left.** Mean average precision for the Cranfield queries when the vector model (2) is used. Each bar is mean average precision when $0, 1, 2 \ldots$ irrelevant documents has smaller angles to its relevant subspace (7) than the query vector. The breadth of each bar is proportional to the number of queries used to compute the mean average precisions. **Right.** Average precision for the 225 Cranfield queries (the stars) when the vector model scoring (2) is used. The queries are sorted according to their closeness to the relevant subspace (7) (the black line). **In each plot** In general average precisions for query vectors orthogonal or with a small part in their relevant subspaces are very moderate. Average precisions for queries closer in angles to their relevant subspaces tend to be better.

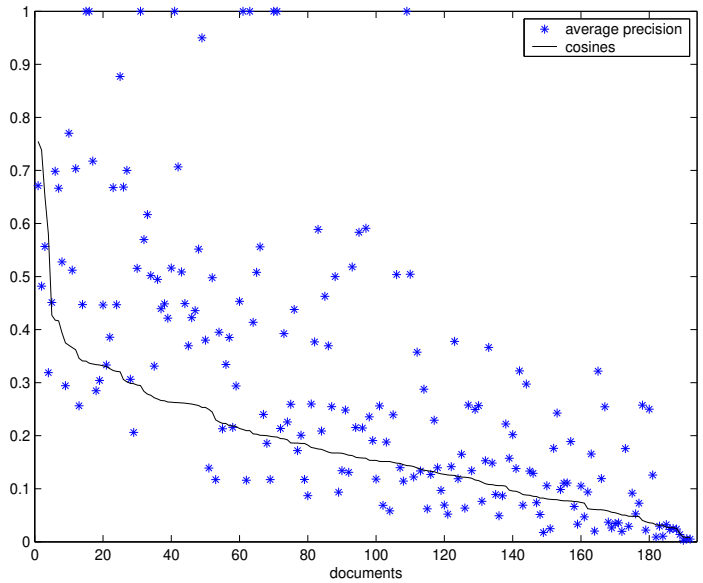

Figure 3: Mean average precisions for the Cranfield queries when the block subspace projection measure (28) is used to score the documents. The queries are sorted according to first principal angle between the subspace $\tilde{\mathcal{R}}$ spanned by relevant retrieved document vectors (14) and the subspace spanned by relevant not retrieved document vectors for each query. Some correlation seems to appear.

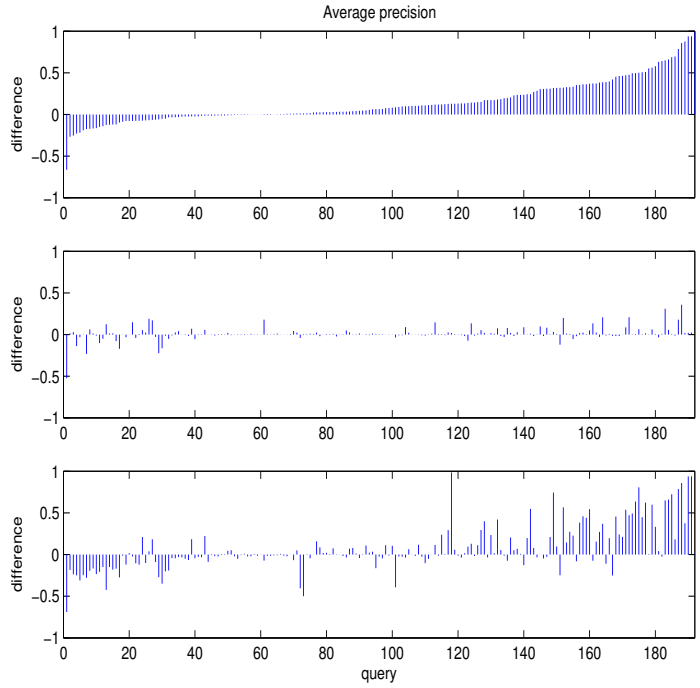

Figure 4: Differences between average precisions for the ranking algorithms (22) (23) (24) using the expanded query vectors from section 4.1 and the vector model (2). Plots from top to bottom are q^+, q^- and q_{ROCHIO}. In all plots the queries are sorted according to the differences in the top plot. Only documents from the residual collection are used when computing average precisions.

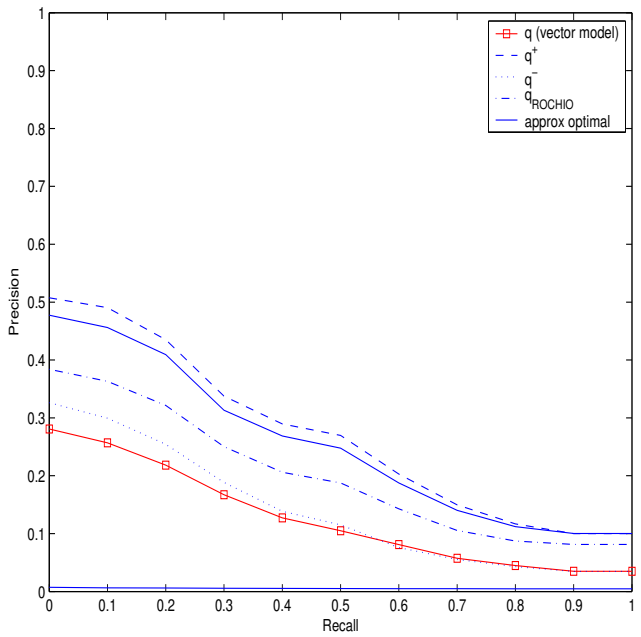

Figure 5: Recall-precision graph for the Cranfield collection comparing the rankings presented in section 4.1.1. The approximate optimal scoring using the relevant retrieved subspace (20) correspond to the upper solid and the approximate optimal scoring using the complement (21) correspond to the lower solid. For all test collections we have tried the q^+ ranking (22) and the approximate optimal scoring (20) are the best. Performance of the q^- (23) and q_{ROCHIO} (24) scorings depend on what tuning constants are used, but performance is rarely above the approximate optimal and the q^+ ranking. All rankings except the approximate optimal scoring (21) give better retrieval performance than the vector model (2).

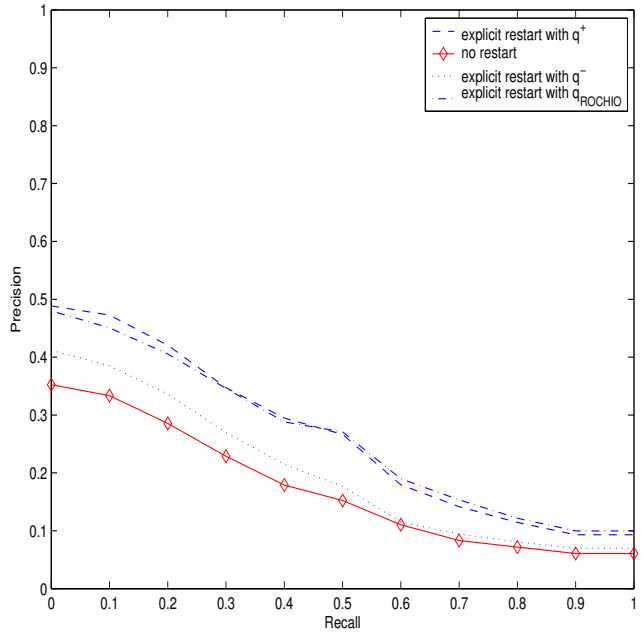

Figure 6: Recall-precision graph for the Cranfield collection. The bidiagonalizing procedure was restarted with the expanded vectors q^+, q^- and q_{ROCHIO} from section 4.1 respectively and the subspace projection measure (4) was used for ranking the documents in the residual collection. For all test collections we have tried explicit restart improves the retrieval performance compared to no explicit restart.

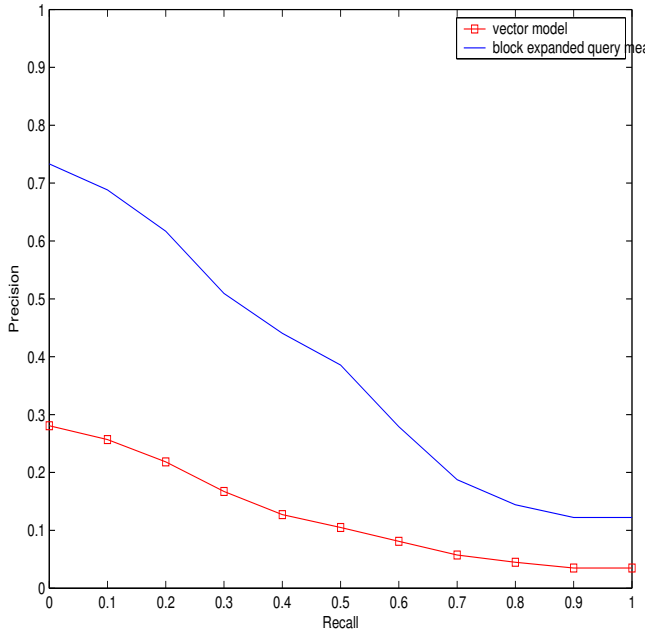

Figure 7: Recall-precision graph for the Cranfield collection. The block expanded query measure (29) is compared with the vector model (2). For all test collections we have tried any of the block measures, block expanded query measure (29) and block subspace projection measure (28), are the best rankings compared to all the other rankings presented in this article.

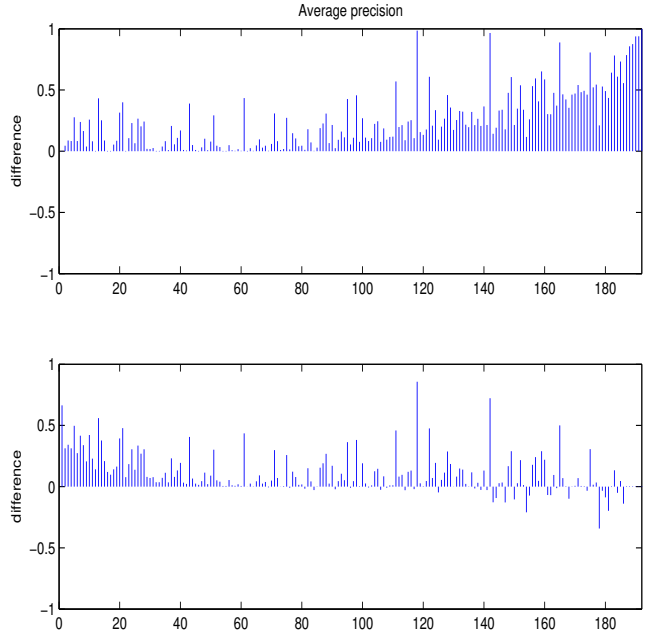

Figure 8: **Upper plot.** Differences in average precisions for block expanded query measure (29) and the vector model (2). **Lower plot** Differences in average precisions for block cosine measure (29) and the q^+ ranking (22) from section 4.1.1. The queries are sorted as in figure (4).

References

[1] *Text REtrieval Conference (TREC).* `http://trec.nist.gov/`.

[2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley, 1999.

[3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[4] A. Björck and G. H. Golub, *Numerical methods for computing angles between linear subspaces*, Math Comp, 27 (1973), pp. 579–594.

[5] K. Blom, *Experimenting with different weighting schemes for the Krylov Subspace method used for IR*, tech. rep., Dept. of Mathematics, Chalmers university of Technology, 2003.

[6] K. Blom and A. Ruhe, *Information Retrieval using a Krylov Subspace method*, Submitted for publication 2003.

[7] ——, *Information Retrieval using very short Krylov sequences*, in Computational Information Retrieval, M. W. Berry, ed., SIAM, 2000, pp. 39–52.

[8] L. Eldén, *Partial Least Squares vs. Lanczos Bidiagonalization I: Analysis of a Projection Method for Multiple Regression*, Tech. Rep. LiTH-MAT-R-2002-24, University of Linköping, Dept. of Mathematics, 2002.

[9] E. A. Fox, *Characterization of two new experimental collections in computer and information science containing textual and bibliographical concepts*, Tech. Rep. 83-561, `http://www.ncstrl.org`, 1983.

[10] W. B. Frakes and R. Baeza-Yates, *Information Retrieval, Data Structures and Algorithms*, Prentice Hall, 1992.

[11] G. Golub and C. F. van Loan, *Matrix Computations 3 ed.*, Johns Hopkins, 1996.

[12] H. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM Journal on Numerical Analysis, 2 (1965), pp. 205–221.

[13] D. Harman, *Relevance feedback and other query modifcation techniques*, in Information Retrieval, Data Structures and Algorithms, W. B. Frakes and R. Baeza-Yates, eds., Prentice Hall, 1992, pp. 241–263.

[14] ——, *A1 (appendix)*, in The Eighth Text REtrieval Conference (TREC-8), D. Harman, ed., NIST Special Publication, 2000, pp. 500–546.

[15] E. Ide, *New experiments in relevance feedback*, in The SMART Retrieval System, G. Salton, ed., Prentice Hall, 1971, pp. 337–354.

[16] 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.

[17] J. J. Rochio, *Relevance feedback in information retrieval*, in The SMART Retrieval System – Experiments in Automatic Document Processing, G. Salton, ed., Prentice Hall, 1971.

[18] A. Ruhe, *Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices*, Mathematics of Computation, 33(146) (1979), pp. 680–687.

[19] D. S. Watkins, *Fundamentals of Matrix Computations*, John Wiley & Sons, 1991.

[20] J. Xu and W. B. Croft, *Query expansion using local and global document analysis*, Proc. ACM SIGIR, (1996), pp. 4–11.

A The Golub-Kahan bidiagonalization procedure

The Golub-Kahan bidiagonalization procedure is a variant of the Lanczos tridiagonalization algorithm and it is widely used in the numerical linear algebra community.

We start the Golub Kahan algorithm with the normalized query vector $q_1 = q/\|q\|$ and use the term document matrix A, and computes two orthonormal bases P and Q, adding one column for each step k, see [11] in section 9.3.3.

ALOGRITHM BIDIAG(A,q,r):
Start with $q_1 = q/\|q\|$, $\beta_1 = 0$
for $k = 1, 2, \ldots r$ **do**
  $\alpha_k p_k = A^T q_k - \beta_k p_{k-1}$
  $\beta_{k+1} q_{k+1} = A p_k - \alpha_k q_k$
end.

The scalars α_k and β_k are chosen to normalize the corresponding vectors.

Define

$$\begin{aligned} Q_{r+1} &= \begin{bmatrix} q_1 & q_2 & \ldots & q_{r+1} \end{bmatrix}, \\ P_r &= \begin{bmatrix} p_1 & p_2 & \ldots & p_r \end{bmatrix}, \end{aligned} \tag{31}$$

$$B_{r+1,r} = \begin{bmatrix} \alpha_1 & & \\ \beta_2 & \alpha_2 & \\ & \ddots & \alpha_r \\ & & \beta_{r+1} \end{bmatrix}.$$

After r steps k we have the basic recursion

$$\begin{aligned} A^T Q_r &= P_r B_r^T \\ A P_r &= Q_{r+1} B_{r+1,r}. \end{aligned}$$

The columns of Q_r will be an orthonormal basis of the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$ and the columns of P_r forms an orthonormal basis for the Krylov subspace $\mathcal{K}_r(A^T A, A^T q)$. The lower bidiagonal matrix $B_{r+1,r} = Q_{r+1}^T A P_r$

is the projection of A into these Krylov subspaces and the singular values of $B_{r+1,r}$ will be approximations of those of A.

It is well known [8] that if the query vector q has large components along some singular vectors that do not correspond to the largest singular values of the term document matrix A then the first few basis vectors in Q_{r+1} (31) will contain large components along these singular vectors. If the components in q are not large enough or if the components correspond to the largest singular values then the first basis vectors in Q_{r+1} will be dominated by components from the singular vectors corresponding to the largest singular values.

B The-Golub Kahan bidiagonalization procedure modified

Assume the subspace $\mathcal{E}$ span directions we want to avoid and let the columns of E span the subspace $\mathcal{E}$.

Using the matrix $(I - EE^T)A$ instead of A in the BIDIAG procedure together with a query vector orthogonal to C we get

Start with $q_1 = \frac{(I-EE^T)q}{\|(I-EE^T)q\|_2}$, $\beta_1 = 0$
for $k = 1, 2, \ldots r$ **do**
  $\alpha_k p_k = A^T(I - EE^T)q_k - \beta_k p_{k-1}$
  $\beta_{k+1} q_{k+1} = (I - EE^T)Ap_k - \alpha_k q_k$
end.

Noting that $E^T q_k = 0$ for all k, the first row in the loop becomes

$$\alpha_k p_k = A^T q_k - \beta_k p_{k-1}.$$

Since we have $EE^T Ap_k = EE^T(Ap_k - \alpha_k q_k)$ the seco nd row in the loop is equal to the two rows

$$\begin{aligned} y &= Ap_k - \alpha_k q_k \\ \beta_{k+1} q_{k+1} &= y - EE^T y. \end{aligned}$$

Thus it is enough to keep the q_k vectors orthogonal to $\mathcal{E}$. The BIDIAG algorithm can be rewritten to

Start with $q_1 = \frac{(I-EE^T)q}{\|(I-EE^T)q\|_2}$, $\beta_1 = 0$
for $k = 1, 2, \ldots r$ **do**
  $\alpha_k p_k = A^T q_k - \beta_k p_{k-1}$
  $y = Ap_k - \alpha_k q_k$
  $\beta_{k+1} q_{k+1} = y - EE^T y$
end.

Experimenting with different weighting schemes for the Krylov subspace method used for Information Retrieval

Katarina Blom

April 12, 2004

Abstract

In a previous report we have described how simple Krylov subspace methods can be used for information retrieval. We used the Golub Kahan bidiagonalization procedure to generate an approximation to a low rank representation of the documents. The process is query based and a new approximation is made for every new query. The Krylov method often shows better retrieval performance than the raw vector model (where documents are scored measuring angles between the query and the original documents).

In this report we explore the effects of 107 different combinations of term weighting schemes for the term document matrix together with 27 different weighting schemes for the queries in each of four test collections. Also, for each weighting performance of three similarity measures for the Krylov method are compared to performance of the vector model, and for the best and worst performing weighting combination for each set also with the LSI.

Our results are rather consistent with results from similar experiments carried out previously.

There is a large difference in performance between the best performing weighting scheme and the worst performing weighting scheme.

There is no overall best weighting, but in general using a term weighting based on the distribution of a term within the whole collection improved performance.

A weighting that is bad for the Krylov subspace method is also bad for the vector model and the LSI.

keywords Krylov subspace, Information Retrieval, weighting experiment, LSI.

1 Introduction and Summary

In a previous report [5] we have described how simple Krylov subspace methods can be used for information retrieval. We used the Golub Kahan bidiagonalization procedure to generate an approximation to a low rank representation of the documents. The process is query based and a new approximation is made for every new query. The Krylov method often shows better retrieval performance than the raw vector model (where documents are scored measuring angles between the query and the original documents).

This report investigates the effect on retrieval performance when different term weightings are used. Simple weighting schemes are constructed using the one-norm, euclidean norm and max-norm. These simple weightings are compared to more sophisticated weighting schemes such as inverse document frequency and the entropy weighting. The weighting schemes used for this report are presented in section 2.

For the Krylov subspace method three similarity measures for scoring documents, the LSI-like measure ($c^{(1)}$), the expanded query measure ($c^{(2)}$) and the subspace projection measure($c^{(3)}$) are compared to the vector model (c). For the best and worst performing weighting combination for each set (performance is measured in average precision) we compare the LSI [4],[9] with the expanded query measure and the vector model. A short summary of the similarity measures and of the Krylov subspace method is given in section 3.

In order to make our experiments comparable to several other similar weighting experiments in the past, we use the four data sets Adi, Cisi, Cran and Med. These sets are old and rather small. The data sets are presented in Appendix B.

In section 4 the numerical results are presented. We explore the effects of 2889 different weighting combinations for the term document matrix and the query vectors in each of four test collections.

Our results are in general consistent with similar experiments carried out previously by Dumais [8], Salton et al [18], Kolda et al. [16],[15], see also Harman [13]. A few trends can be observed.

1. There is a large difference in performance between the best performing weighting scheme and the worst performing weighting scheme. Performance is measured in average precision.

2. There is no overall best weighting for all similarity measures and all four test sets, but in general using a term weighting based on the distribution of a term within the whole collection improves performance.

3. It seems important which query weighting is chosen (or at least the combination of term weighting and query weighting seems to be important).

4. In general the more sophisticated weighting schemes give better performance compared to the simpler vector norm weightings. But the euclidean norm is not far behind. The one-norm weighting in general decreases performance and should not be used.

5. A weighting that is bad for any of the Krylov subspace similarity measures (the LSI-like measure, the expanded query measure and the subspace projection measure) is also bad for the vector model and the LSI.

Notations The notations used in this report are rather standard in the Numerical Linear Algebra community. We use upper case letters for matrices and lower case letters for vectors. Lower case Greek letters usually denotes scalars. Component indices are denoted by subscript. For example, a vector c and a matrix M might have entries c_i and m_{ij} respectively. On the occasions when both an iteration index and a component index are needed, the iteration is indicated by a parenthesised superscript, as in $c_j^{(r)}$ to indicate the jth component of the rth vector in a sequence. Otherwise c_j may denote either the jth component of a vector c or the jth column of a matrix C. The particular meaning will be clear from its context.

The pseudo inverse of a matrix B is denoted B^+.

All of the vector norms we will use are instances of p-norms, which for a real $p \geq 1$ and a vector x of dimension n are defined by

$$\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}.$$

The special cases we use are
one-norm:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

euclidean norm:

$$\|x\|_2 = (\sum_{i=1}^{n} |x_i|^2)^{1/2}$$

and *max-norm*

$$\|x\|_\infty = \max_i |x_i|.$$

All norms on $\mathcal{R}^n$ are equivalent, i.e. if $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ are p-norms on $\mathcal{R}^n$, then there exist positive constants c_1 and c_2 such that

$$c_1\|x\|_\alpha \leq \|x\|_\beta \leq c_2\|x\|_\alpha \tag{1}$$

A *Krylov subspace* of a square matrix M, starting at the vector v, is a subspace of the form

$$\mathcal{K}_r(M, v) = \text{span}\{v, Mv, M^2v, \ldots M^{r-1}v\}.$$

Measures The retrieval efficiency of an information retrieval system depends on two main factors. The ability of the system to retrieve relevant information and the ability to dismiss irrelevant information. The ability to retrieve relevant information is measured by *recall*, the ratio of relevant documents retrieved over the total number of relevant documents for that query. A systems ability to reject irrelevant documents is measured by *precision*, the ratio of the number of relevant documents retrieved for a given query over the total number of documents retrieved. Precision and recall are usually inversely related (when precision goes up, recall goes down and vice versa).

When we evaluate a query q all documents in the set are ranked and we receive an ordered list $\mathcal{L}$ of documents. Assume t documents are relevant to the query and let ℓ_i, $i = 1 \ldots t$ be the position for the ith relevant document in $\mathcal{L}$. The *average precision* (non interpolated) for a single query is defined as

$$\frac{1}{t}\sum_{i=1}^{t}\frac{i}{\ell_i}.$$

The *mean average precision* for multiple queries is defined as the mean of the average precisions for all queries.

Precision can be computed at any *actual recall level* $\frac{i}{t}$, $i = 1 \ldots t$ (where t is the number of relevant documents to the query). Let r_j be the jth

recall level from the 11 *standard recall levels* $0, 0.1, 0.2 \ldots 1$. The *interpolated average precision* for a query at standard recall level r_j is the maximum precision obtained for any actual recall level greater that or equal to r_j.

The *Recall level precision averages* for multiple queries are the means of the interpolated average precision values at each (standard) recall level for the queries. Recall level precision averages are used as input for plotting the recall-precision graphs.

For further details, see Harman [14].

2 The term document matrix

In vector space models both queries and documents are encoded as vectors in m-dimensional space. The choice m is the number of unique terms in the collection. The documents are stored as columns in a $m \times n$ *term document matrix* A. The elements in A are the occurrences of each word in a particular document, i.e.

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

where a_{ij} is nonzero if term i occurs in document j, zero otherwise.

A term weight has three components; local, global and normalization [18]. *Local weights* are used to transform the term's frequency within the document. Each term in the collection is assigned a *global weight* to indicate its importance as an indexing term. A *normalization factor* is used to normalize the documents. We let

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

where l_{ij} is the local weight for term i in document j, g_i is the global weight for term i and d_j is the document normalization factor.

Specifically we can write

$$[a_{ij}] = A = GLD \tag{2}$$

where the elements in $L = [l_{ij}]$ are the local weights. G and D are diagonal matrices and g_{ii} in G is the global weight for term i and d_{jj} in D is the normalization factor for document j. The global weighting correspond to a *row scaling* of the term document matrix and the normalization corresponds to a *column scaling*.

There are several local and global weightings that can be used. For nice summaries see for example Frakes and Baeza-Yates [11], Salton and McGill [19] or Kolda [15].

The queries are stored the same way as the document vectors, that is

$$q = [q_i]$$

where q_i is nonzero if term_i appears in the query. As for the elements in the term document matrix local and global weightings are used. The local weights are computed using the term frequency within the query vector and the global weights are computed from the frequency counts in the documents. Normalizing the query makes no difference when ranking the documents and is not used [1].

For convenience [16], let

$$\chi(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{ir } x = 0 \end{cases}$$

Various combinations of weights are used for the documents in the term document matrix and for the queries. Each term weight combination is described using two three letter strings, representing the weightings for the term document matrix (first triple) and the query terms (second triple). The letters in each string represent the local, global and normalization component respectively.

Formulas and symbols for the weightings used for this report are shown in tables 1 - 3.

For example the classical idf weight [18] is described by the string

$$\texttt{bfx} \cdot \texttt{bfx},$$

which implies the local, global and normalization components

$$\begin{aligned} l_{ij} &= \chi(\text{tf}_{ij}) \\ g_i &= \log_2(\frac{n}{\text{df}_i}) \\ d_j &= 1 \end{aligned}$$

for the elements in the term document matrix. The *term frequency* tf_{ij} is the the number of times term i appears in document j, and the *document*

[1] In the Krylov subspace method used in this report (section 3 gives a short summary) the query vector is always normalized using euclidean norm before the bidiagonalization procedure is started.

frequency df_i is the number of documents to which term i is assigned. The local global and normalization components for the query vector elements are

$$\begin{aligned} l_i &= \chi(\text{tf}_i) \\ g_i &= \log_2(\frac{n}{\text{df}_i}) \\ d_j &= 1. \end{aligned}$$

Here tf_i is the term frequency for the terms in the query (i.e. the number of times term i appears in the query) and df_i is the document frequency for term i in the collection.

The binary local weighting (`b`) and the local frequency weighting (`t`) listed in table 1 are simple but with some major drawbacks. The binary weighting gives every word that appear in a document equal relevance. (This might be useful when the number of times a word appears is not considered important.)

The local frequency weighting give more credit to words that appear more frequently which might serve the recall function. For example a term such as melon appearing with reasonable frequency in some documents indicates that they deal with melons. The assignment of the term melon with high weight will then help to retrieve these documents in response to appropriate queries.

On the other hand, high precision implies high ability to distinguish individual documents from each other (to be able to prevent unwanted retrievals), therefore when common terms are not concentrated in a few documents but instead are spread out in the whole collection, precision is likely to drop.

More concretely, if the whole document collection deals with melons, almost all documents will contain the term melon many times, giving high credit to melon, will not help to identify the wanted subset of documents.

For a more detailed discussion see for example Salton [17] (or Salton and Buckley [18]).

The (local) augumented normalized term frequency (`n`) will give basic credit (0.5) to any word that appears and then give additional credit to words that appear more frequently.

The logarithmic weight (`l`) will deemphasize the effect of high frequency.

The choice for local weightings depends on the vocabulary used for the collection. Some general recommendations can be made [3]. Local binary term weighting schemes are recommended for sets where the term list (the number of rows in the term document matrix) is short. The local frequency

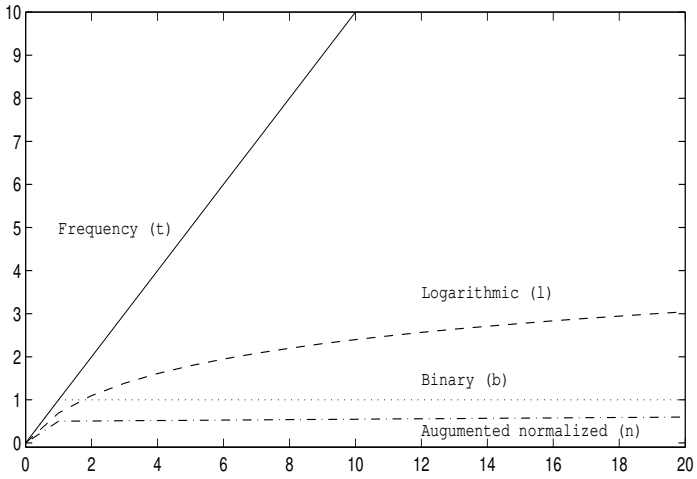

Figure 1: Comparison of local term weighting schemes. The term frequency range from 0 to 20 (x-axis) and the weights (y-axis) range from 0 to 10 in the figure.

weighting is recommended for varied vocabularies, eg. popular magazines, and the augumented normalized term frequency is recommended for technical or scientific vocabularies.

The four local weightings are compared in figure 1. The term frequency range from 0 to 20. The raw frequency grows very quickly compared to the other local weightings grow more slowly.

As mentioned above, precision might be better served by using very specific terms that will match the most relevant documents in the collection, because such terms are able to distinguish the few documents in which they appear from the many from which they are absent. All of the global weighting schemes in table 2 (except `x`) give less weight to frequent terms. So in order to fulfill both the requirements of high recall and of high precision, i.e. to credit those terms that occur frequently in individual documents but rarely in the remainder of the collection, the combination of local term frequency weighting and any of the global weightings may be used.

The global weightings `n`, $\mathtt{n}_1$ and $\mathtt{n}_\infty$ are based on simple vector norms and will normalize the length of each row in the term document matrix in some norm. This has the effect of giving high weight to infrequent terms. If a few rare terms appear frequently in only a few documents the max-norm is giving the most credit to these terms, followed by the euclidean norm and then the one-norm.

The entropy global weighting (`e`) uses concepts from information theory. In information theory the least predictable terms in a running text, those exhibiting the smallest probabilities, carry the greatest information value. The weighting assign weights between zero and one. Zero for a term appearing with the same frequency in every document and one for a term that appears only once.

The weights given by the different global schemes to two different terms in a collection are compared in figure 2. For both terms the local term frequency (`t`) was used. The term in the upper plot appears once in one document and three times in another. The term is rare in the set and all of the global weighting schemes give high credit to the term in the two documents where it appears. In the lower plot a term appears in all but one document. This term is common in the set. The global schemes will not emphasize the appearance of the term as they did for the rare term. All the weighting schemes give slightly more credit to the term in the document where it appears three times.

The normalizing factors will normalize the length of each column in the term document matrix. This has the effect of giving higher weights to all terms in short documents and giving lower weights to all terms in long documents. If using the angles between the query and the document vectors in the therm document matrix when ranking documents for relevancy there is a tendency that shorter documents will be ranked more relevant than longer documents. In order to retrieve documents of a certain length with the same probabilities the pivoted cosine normalization scheme has been proposed for indexing the TREC collection [6], [20].

In this report we always apply first the local weighting, then the global weighting and at last the normalization factor. For example, the matrix weighting `tnc` corresponds to first normalizing the rows in the term document matrix using euclidean norm, then normalizing the columns (using euclidean norm). Note that the column normalization might destroy the previous row normalization, but not completely. Some deemphasizing effect on common terms still remain.

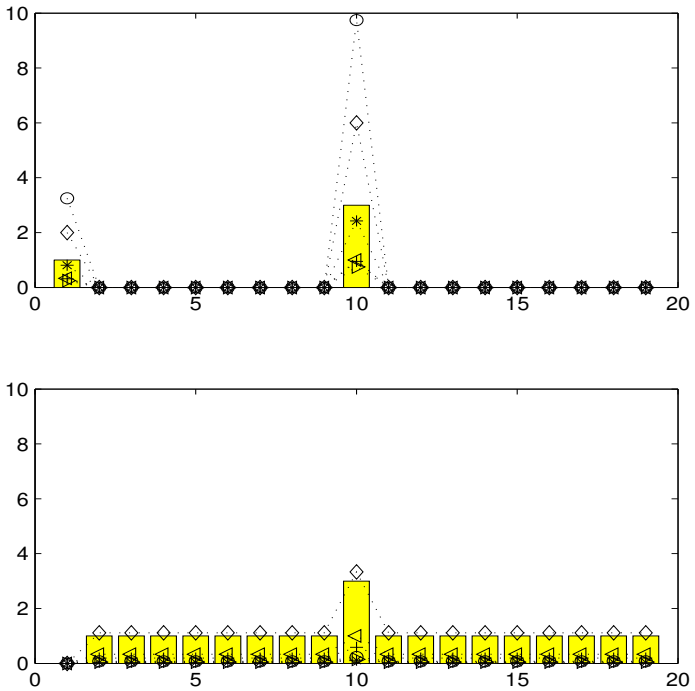

Figure 2: Comparison of global term weighting schemes when the local term frequency weighting (`t`) is used. The bars are the term frequencies. The global weighting schemes are: no weighting `x` (the bars), inverse document frequency `f` ($\circ$), GfIdf `g` ($\diamond$), entropy `e` ($*$), normal `n` ($+$), one-norm $\mathtt{n}_1$ ($\triangleright$) and max-norm $\mathtt{n}_\infty$ ($\triangleleft$).

Local Weighting		Description
`b`	$\chi(\mathrm{tf}_{ij})$	*Binary weight* [18] equal 1 for terms present in vector, zero otherwise. The term frequency tf_{ij} is the number of times term i appears in document j.
`t`	tf_{ij}	*Raw frequency weight* [18] is number of times a term appears in a document or a query.
`l`	$\log_2(1+\mathrm{tf}_{ij})$	*Logarithmic weight* [8][13] takes the log of the term frequency, thus dampening effects of large differences in frequencies.
`n`	$\frac{1}{2}\left(\chi(\mathrm{tf}_{ij}) + \frac{\mathrm{tf}_{ij}}{\max_k(\mathrm{tf}_{kj})}\right)$	*Augumented normalized term frequency* [18][13]. The term frequency tf_{ij} is normalized by maximum appearance of term in document j and further normalized to lie between 0.5 and 1.0[2].

Table 1:

[2] A more general formula was proposed by Croft [7]. The formula was parameterized by a value K (a sliding importance factor), $t_{ij} = \chi(\mathrm{tf}_{ij})\mathrm{K} + (1-\mathrm{K})\frac{\mathrm{tf}_{ij}}{\max_k(\mathrm{tf}_{kj})}$. It is suggested that K be low for large documents and high for short documents.

[3] In [8] $l_{ij} = \mathrm{tf}_{ij}$

[4] In [18] $l_{ij} = \mathrm{tf}_{ij}$.

Global Weighting		Description
x	1	No change in weight [18].
f	$\log_2(\frac{n}{\mathrm{df}_i})$	*Inverse document frequency* (Idf) [18] where n is number of documents in collection and df_i is the document frequency (the number of documents to which term i is assigned).
g	$\frac{\mathrm{gf}_i}{\mathrm{df}_i}$	*GfIdf* [8]. gf_i is the global frequency (the total number of times term_i appears in the whole collection). df_i is the document frequency.
e	$1-\sum_{j=1}^{n}\frac{p_{ij}\log(\frac{1}{p_{ij}})}{\log(n)}$	*Entropy* [8][13]. n is number of documents in collection and $p_{ij}=\frac{\mathrm{tf}_{ij}}{\mathrm{gf}_i}$ where tf_{ij} is the raw term frequency and gf_i is the global frequency.
n	$\frac{1}{\sqrt{\sum_j l_{ij}^2}}$	*Normal* [8], where l_{ij} is received after applying any of the local weightings presented in table 1 [3].
n_1	$\frac{1}{\sum_j l_{ij}}$	where l_{ij} is received after applying any of the local weightings presented in table 1.
n_∞	$\frac{1}{\max_j l_{ij}}$	where l_{ij} is received after applying any of the local weightings presented in table 1.

Table 2: .

Normalization factor		Description
x	1	No normalization factor is used [18].
c	$\frac{1}{\sqrt{\sum_i (g_i l_{ij})^2}}$	*Cosine normalization* [18] [4].
n_1	$\frac{1}{\sum_i g_i l_{ij}}$	
n_∞	$\frac{1}{\max_i g_i l_{ij}}$	

Table 3: The local weightings l_{ij} and global weightings g_i are received after applying any of the local and global weightings respectively presented in tables 1 and 2

3 The Krylov subspace method for Information retrieval

Query matching can be viewed as a search in the column space of the term document matrix A. One of the most common similarity measures used for query matching is to measure the angle between the query vector and the document vectors in A. The smaller the angle is the more relevant the document is. In the *vector model* the cosines between the query vector q and document vectors a_j are used to score the documents in relevance order,

$$c_j = \frac{q^T a_j}{\|q\|_2 \|a_j\|_2}, \quad j = 1, \ldots, n. \tag{3}$$

For the Krylov subspace methods we will use the Golub Kahan bidiagonalization procedure [12] applied to the term document matrix A starting with the query vector q to receive the two *basis matrices* Q_{r+1} and P_r and the $r+1 \times r$ lower bidiagonal matrix B_{r+1}:

$$[Q_{r+1}, B_{r+1}, P_r] = \textsc{Bidiag}(A, q, r) \tag{4}$$

The column vectors in the basis matrices Q_{r+1} and P_r span bases for the two Krylov subspaces $\mathcal{K}_{r+1}(AA^T, q)$, in the document space (spanned by the query q and the columns of A) and $\mathcal{K}_r(A^TA, A^Tq)$, in the term space (spanned by the rows of A) respectively. We let the *reached subspace* W form an orthonormal basis for the column vectors in AP_r.

The Bidiag procedure is further described in section 3.1.

The reached subspace W, the basis matrices Q_{r+1}, P_r and the B_{r+1} matrix are used to score the documents in relevance order to the query (see Blom Ruhe [5]). A few examples of similarity measures are:

- For the *subspace projection measure* the documents in A are sorted according to their closeness measured in angles to the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$. The closer the document is the more relevant the document is. The cosine of the angle between the basis matrix Q_{r+1} for the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$ and each document vector in A

  $$c_j^{(3)} = \|Q_{r+1}^T a_j\|, \; j = 1, \ldots, n, \tag{5}$$

  is used to sort the documents. Note that for $r = 0$ in the Bidiag procedure the subspace projection measure is simply the vector model scoring (3).

- A projected query vector

$$\hat{q} = WW^T q \tag{6}$$

is constructed using the reached subspace. In the *expanded query measure* the documents are sorted measuring the angle between $\hat{q}$ and each document vector in A,

$$c_j^{(2)} = \frac{\hat{q}^T a_j}{\|a_j\|}, \ j = 1, \ldots, n. \tag{7}$$

In the *LSI-like measure* we mimic the LSI [5] and the documents are scored measuring the angle between $\hat{q}$ and each projected document vector in A

$$c_j^{(1)} = \frac{\hat{q}^T a_j}{\|W^T a_j\|}, \ j = 1, \ldots, n. \tag{8}$$

The smaller the angle the more relevant the document is. Note that if the starting vector $q \in \mathcal{R}(A)$ then the projected query $\hat{q} = q$ and the cosines (7) is simply the vector model scoring (3).

3.1 The Golub Kahan bidiagonalization procedure

The Golub Kahan bidiagonalization procedure is a variant of the Lanczos tridiagonalization algorithm and it is widely used in the numerical linear algebra community.

The Golub Kahan algorithm starts with the normalized query vector $q_1 = q/\|q\|$, and computes two orthonormal bases P and Q, adding one column for each step k, see [12] in section 9.3.3.

ALGORITHM BIDIAG(A,q,r):
Start with $q_1 = q/\|q\|$, $\beta_1 = 0$
for $k = 1, 2, \ldots r$ **do**
  $\alpha_k p_k = A^T q_k - \beta_k p_{k-1}$
  $\beta_{k+1} q_{k+1} = A p_k - \alpha_k q_k$
end.

[5] In LSI, Berry et al [4], Dumais et al [9], see also Berry and Brown [3], the $m \times n$ term document matrix is represented using a rank-k approximation, $k < \min(m, n)$, from the singular value decomposition of A. Documents are scored measuring the angles between the query and the column vectors in the approximation.

The scalars α_k and β_k are chosen to normalize the corresponding vectors.

Define

$$\begin{aligned} Q_{r+1} &= \begin{bmatrix} q_1 & q_2 & \ldots & q_{r+1} \end{bmatrix}, \\ P_r &= \begin{bmatrix} p_1 & p_2 & \ldots & p_r \end{bmatrix}, \end{aligned}$$

$$B_{r+1} = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & & \alpha_r \\ & & & \beta_{r+1} \end{bmatrix}.$$

After r steps k we have the basic recursions

$$\begin{aligned} A^T Q_r &= P_r B_r^T \\ A P_r &= Q_{r+1} B_{r+1}. \end{aligned}$$

The columns of Q_r will be an orthonormal basis of the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$ and the columns of P_r forms an orthonormal basis for the Krylov subspace
$\mathcal{K}_r(A^T A, A^T q)$. The lower bidiagonal matrix $B_{r+1} = Q_{r+1}^T A P_r$ is the projection of A onto these Krylov subspaces and some of the singular values of B_{r+1} will be approximations of those of A.

With r large enough the bidiagonalization procedure BIDIAG(A,q,r) can be used to compute a solution $x_L = P_r B_{r+1}^+ e_1$ for the least squares problem

$$\min_x \|Ax - q\|_2.$$

Let $k < r$. The projected query vector (6) $\hat{q} = Ax^{(k)}$ where $x^{(k)} = P_k B_{k+1}^+ e_1$ is an approximation to x_L received after k iterations in the BIDIAG procedure.

3.2 Numerical aspects of using weighting schemes

Let $A = GLD$ be the term document matrix defined in (2). If no global weighting or normalization factor is used (i.e. global weighting and normalization factor $\mathbf{x}$ respectively is used) then $A = L$. Consider the least squares problem

$$\min_x \|Lx - q\|_2 \tag{9}$$

A solution x_L to this problem can be obtained by using the BIDIAG procedure with L and starting at q (see for example [12] or [2]).

If no global weighting (i.e. global weighting **x** from table 2) is used for the term document matrix then $A = LD$ where D is the $n \times n$ diagonal matrix defined in (2). Assume D is nonsingular (i.e. assume all documents has at least one term) and consider the least squares problem

$$\min_y \|LDy - q\|_2. \tag{10}$$

Multiplying L by a diagonal matrix from the right corresponds to a column scaling of L and the solution x_L to problem (9) can be obtained by finding the minimum 2-norm solution y_L to problem (10). If rank(L)= n then $x_L = Dy_L$ otherwise Dy_L is the minimum D-norm solution [6] to (9).

However it is well known that column scaling affects singular values and that the number of iterations needed in the BIDIAG procedure before the solution is reached heavily depend on the distribution of singular values in the matrix that is used. When we use the BIDIAG procedure for IR purposes we stop iterating after $r \leq 10$ steps, that is long before a solution to any of the least squares problems (9) and (10) is reached. This means that we cannot use the relations between x_L and y_L directly when computing scorings $c^{(1)}$and $c^{(2)}$.

Assume only global weighting is used (i.e. normalization factor **x**) and consider the weighted least squares problem

$$\min_s \|G(Ls - q)\|_2 \tag{11}$$

where G is the $m \times m$ diagonal matrix with global weights defined in (2). (In equation (11) we have assumed that the terms in the query vector are weighted using the same global weighting as for the term document matrix [7]). Multiplying L by a diagonal matrix G from the left correspond to a row scaling of the matrix L (and query vector q). It is well known that row scaling affects the solution to a least squares problem [8] and there is no simple relation between the solutions to (9) and (11).

[6] D-norm is defined by $\|z\|_D = \|D^{-1}z\|_2$.

[7] In the experiments performed (see section 4) we have also tried combinations when L and q have different global weights.

[8] An exception occurs when $q \in \mathcal{R}(L)$. In this case the solution to (9) and (11) are equal. In all test sets we tried the query vector q is never completely in $\mathcal{R}(\mathcal{A})$.

4 Experiments

For our experiments four test sets were used, Adi, Cisi, Cran and Med. The sets are further described in appendix B .

We have tried all possible combinations of local, global and normalization factors from tables 1, 2 and 3 for the term document matrix. In tables 5 and 6 all weighting combinations we used are listed. For each weighting on the term document matrix the queries were weighted using all combinations of local and global weightings listed in table 6. (The document frequencies and the global frequencies are taken from the term document matrix.). In total we explored the effect of $105 * 27 = 2889$ different weighting combinations.

Using these data sets and weighting schemes makes our experiments comparable for example with the LSI experiments made by Dumais in [8], some of the experiments made by Salton et al [18] and with the LSI and LDD experiments made by Kolda et al [16].

For each weighting combination four similarity measures were used to score the documents, the vector model c (3), the LSI-like measure $c^{(1)}$(8), the expanded query measure $c^{(2)}$ (7) and the subspace projection measure $c^{(3)}$ (5). For the Krylov subspace methods the iterations in the BIDIAG procedure were stopped when maximum average precision before the number of steps $r = 10$ for each query was reached.

For the best and worst weighting combination for the expanded query measure we computed recall level average precisions for the LSI [4],[9]. For the LSI we need to chose a rank k (the number of singular vectors to use) for the low rank approximation of the term document matrix. We chose $k \leq 100$ (for Adi we let $k \leq 60$) to be the rank where where maximum mean average precision was found.

Computational Results For each weighting the number of times each of the four similarity measures gave best mean average precision is shown in figure 3. The expanded query measure $c^{(2)}$ generally give best average precision in Cran and Med. In Adi and Cisi the LSI-like measure $c^{(1)}$ gave best average precision in a little more than half of the weighting combinations. The vector model c is never the best one. Observe that since BIDIAG is stopped when best average precision before the number of steps $r = 10$ is reached the subspace projection measure $c^{(3)}$ never score worse than the vector model.

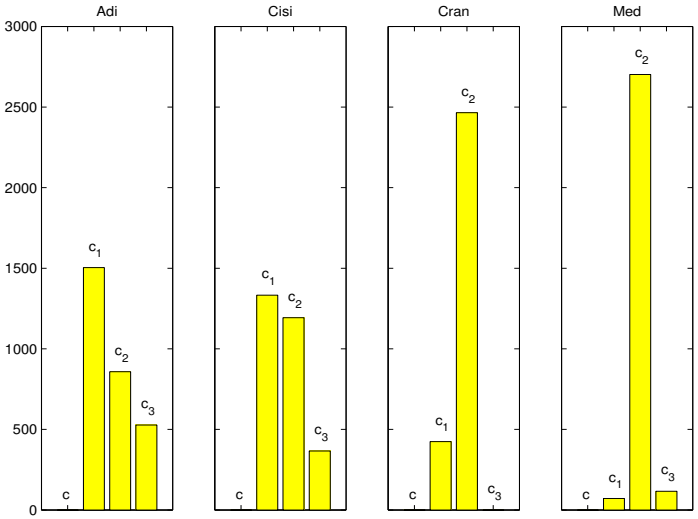

Figure 3: For each weighting the number of times each of the four similarity measures $c(3)$ $c^{(1)}$ (8), $c^{(2)}$ (7), and $c^{(3)}$ (5) gave best mean average precision.

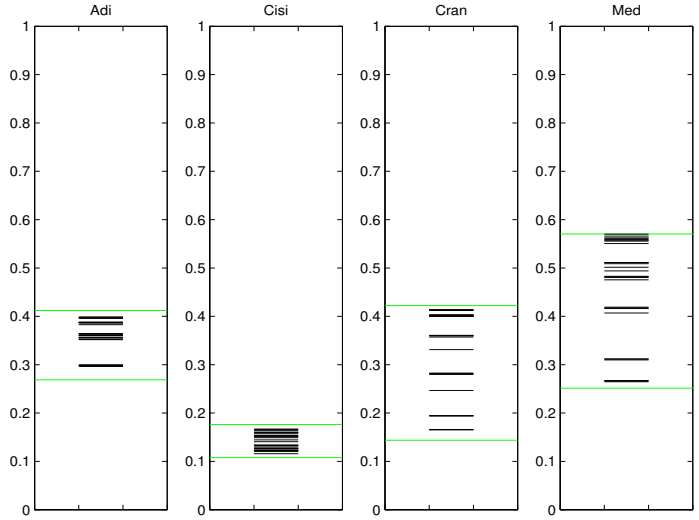

Figure 4: Comparison of mean average precisions for the vector model c (3). The weighting scheme `nxx` was used for the term document matrix. Mean average precision for each query weighting from table 6 is marked (the black lines). The grey lines are maximum and minimum mean average precision respectively for the vector model scoring in each set.

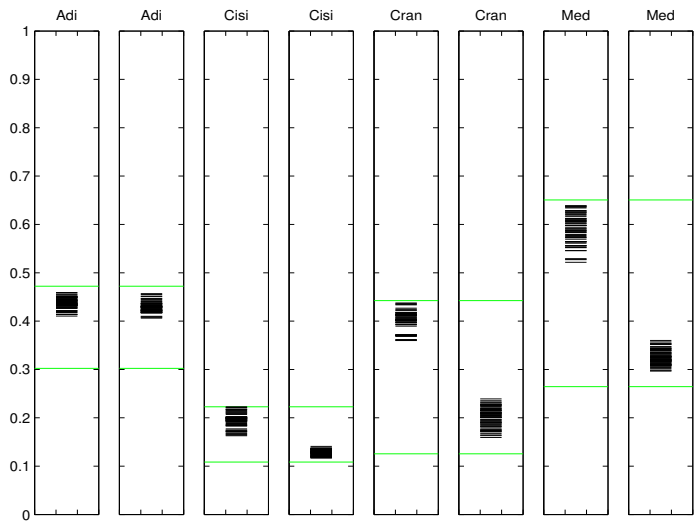

Figure 5: Comparision of mean average precisions for the LSI-like measure $c^{(1)}$ (8). In the left plot of each pair mean average precisions for weighting schemes where global entropy weighting (e) are used for both the term document matrix and the queries are marked (the black lines). In the right plot of each pair mean average precisions for weighting schemes where global one-norm weighting (n_1) is used for both the term document matrix and the queries are marked (the black lines). The grey lines are maximum and minimum mean average precision respectively for the LSI-like measure in each set.

Adi							
c		$c^{(1)}$		$c^{(2)}$		$c^{(3)}$	
ngx·l1x	0.41	bgc·t1x	0.47	bgc·t1x	0.49	lfc·lnx	0.48
lxx·l1x	0.41	te1·n1x	0.47	bgn_∞·l1x	0.48	tec·t1x	0.48
lex·lnx	0.41	bgc·l1x	0.47	ngn_∞·t1x	0.48	tfc·t1x	0.48
lgx·l1x	0.41	bgc·tnx	0.47	len_∞·nnx	0.47	lec·lnx	0.48
nxx·n1x	0.41	txx·tfx	0.47	bgn_∞·t1x	0.47	nfc·n1x	0.47
lfx·lnx	0.41	te1·b1x	0.47	bgx·t1x	0.47	nec·l1x	0.47
nxx·l1x	0.41	nxx·n1x	0.47	nfn_∞·bnx	0.47	nfc·l1x	0.47
lgx·t1x	0.41	nxn_∞·n1x	0.47	tx1·t1x	0.47	lfc·l1x	0.47
tex·t1x	0.41	bgn_∞·lnx	0.46	bg1·t1x	0.47	lec·l1x	0.47
Cisi							
c		$c^{(1)}$		$c^{(2)}$		$c^{(3)}$	
ngx·tfx	0.17	lex·tex	0.22	tgx·l1x	0.23	nfx·tgx	0.20
ngx·lfx	0.16	lex·lex	0.22	tgx·n1x	0.23	nfx·lgx	0.20
ngx·tex	0.16	lfx·tex	0.22	tgx·b1x	0.23	nex·tgx	0.20
lgx·tfx	0.16	lex·tfx	0.22	lex·tfx	0.23	nex·lgx	0.20
ngx·nfx	0.16	lex·nex	0.22	lex·tex	0.23	nfx·ngx	0.20
lgx·lfx	0.16	lfx·lex	0.22	lex·lfx	0.23	nnx·tgx	0.19
ngx·lex	0.16	lex·lfx	0.22	lfx·tex	0.23	nex·ngx	0.19
lgx·nfx	0.16	lfx·tfx	0.22	lex·lex	0.23	nnx·lgx	0.19
ngx·nex	0.16	lfx·nex	0.22	lex·nfx	0.23	lgx·lfx	0.19
Cran							
c		$c^{(1)}$		$c^{(2)}$		$c^{(3)}$	
ngx·lfx	0.42	lfc·bgx	0.44	ngx·lnx	0.51	ngc·lfx	0.43
ngx·bex	0.42	lfc·ngx	0.44	ngx·nnx	0.51	ngc·bex	0.43
ngx·nfx	0.42	lfc·lgx	0.44	ngn_∞·nnx	0.51	ngc·lex	0.43
ngx·lex	0.42	lec·bgx	0.44	ngn_∞·lnx	0.50	ngc·nfx	0.43
ngx·bfx	0.42	lec·ngx	0.44	ngc·nnx	0.50	ngc·bfx	0.43
ngx·nex	0.42	lfc·tgx	0.44	lgx·lnx	0.50	ngc·nex	0.43
lgx·bfx	0.42	lec·tex	0.44	ngc·lnx	0.50	ngc·tex	0.43
lgx·nfx	0.42	lec·lgx	0.44	lgc·lnx	0.50	ngc·tfx	0.43
ngx·tex	0.42	lec·lex	0.44	ngc·bnx	0.50	lgc·nfx	0.43
Med							
c		$c^{(1)}$		$c^{(2)}$		$c^{(3)}$	
ngx·bex	0.57	lec·bgx	0.65	ngn_∞·bnx	0.68	lfc·bgx	0.62
ngx·bfx	0.57	lfc·bgx	0.65	ngc·bnx	0.68	ngc·bnx	0.61
ngx·nex	0.57	lec·ngx	0.65	ngx·bfx	0.68	lec·bgx	0.61
ngx·nfx	0.57	lfc·ngx	0.65	lgn_∞·bnx	0.68	ngc·bfx	0.61
lgx·bex	0.56	lec·lgx	0.64	ngx·lnx	0.68	lfc·ngx	0.61
lgx·bfx	0.56	lfc·lgx	0.64	lgc·nnx	0.68	lfc·bfx	0.61
ngx·lfx	0.56	tfc·bfx	0.64	ngc·nnx	0.68	lfc·bex	0.61
ngx·lex	0.56	tfc·bex	0.64	ngx·bnx	0.68	lec·ngx	0.61
nex·bgx	0.56	tfc·nfx	0.64	ngx·nnx	0.68	lec·bfx	0.61

Table 4: The nine best performing weighting schemes for each set and each similarity measure. Performance is measured in mean average precision. Since the vector norms used as normalization factors are equivalent (1) they have no effect for the vector model (3) and are not listed in the table.

Tables 4 show numerical results for the Adi, Cisi, Cran and Med data sets. For each test we report the mean average precision for all queries in the set.

As we can see there is no overall best weighting for all similarity measures, however a few trends can be seen. We observe that for the vector model c (3) the matrix weightings `ngx` and `lgx` give the best results for all test sets.

The binary matrix weighting `bxx` and the raw term frequency weighting `txx` combined with no global weighting for the query vector tend to be ranked towards the bottom.

For the LSI-like measure $c^{(1)}$ (8) the global entropy weighting (`e`) is good. The matrix weightings $\mathtt{ten}_1$ and `lex` are good for Adi and Cisi respectively and the weighting `lec` is good for Cran and Med. But also the global inverse frequency weighting (`f`) and the GfIdf weighting (`g`) are good. The weightings `bgc` and $\mathtt{bgn}_\infty$ are good for Adi. Weighting `lfc` is good for Med and Cran and weighting `lfx` is good for Cisi.

Also for the expanded query measure $c^{(2)}$ (7) the global inverse frequency weighting (`f`) and the GfIdf weighting (`g`) work well. Matrix weightings `bgc` and $\mathtt{bgn}_\infty$ are good for Adi and weighting `tgx` is good for Cisi. For Cran and Med weightings `ngx` and `ngc` are good. But also the global entropy weighting (`e`) seems to work well.

Among the poor performing weighting combinations the global one-norm weighting ($\mathtt{n}_1$) is frequent. And for Adi also the global max-norm weighting ($\mathtt{n}_\infty$) is bad.

For the subspace projection measure $c^{(3)}$ (5) the global inverse frequency weighting (`f`) and the GfIdf weighting (`g`), but also the entropy global weighting (`e`) works well.

One trend found in weighting experiments is that the use of global weights improves performance (or at least does not hurt performance) [13]. In our experiments in general the use of global weights improves performance except when the global one-norm weighting ($\mathtt{n}_1$) is used. The global one-norm weighting is bad for all sets but the Adi. In figure 5 the mean average precision for the global entropy weightings and the global one-norm weightings are compared.

For the local weightings in general the binary weighting (`b`) appear among the poor performers, however the weighting works well for the Adi. This might be due to the small size of the term document matrix

It seemed to be important which query weighting was chosen. In figure 4 we plotted mean average precision for the matrix weighting `nxx` and all

27 different query weightings listed in table 6 for all test sets. As we can see the differences in mean average precisions are large. In general using a global weight for the query vector, preferably any of entropy (`e`), inverse document frequency (`f`), GfIdf (`g`) or normal (`n`), seems to improve retrieval performance.

Our results for the vector space model are quite consistent with those reported by Kolda [15]. Somewhat surprisingly she found that it makes little difference which query weighting is chosen.

In [8] Dumais report good performance for the `lec` matrix weighting on the matrix and Salton's *best weighting* reported in [18] was `tfc·tfx`. In general these weightings also work well in our experiments.

The weighting `tnc·txx` used by Blom Ruhe in [5] is among the average (sometimes above average) performing weighting schemes.

Figure 6 are recall-precision graphs for the best and worst performing weighting combinations for the expanded query measure $c^{(2)}$ (7) in each set. For each set interpolated average precision for the vector model c (3), the expanded query measure $c^{(2)}$ and the LSI are compared. We observe that a weighting combination that is bad for the expanded query measure $c^{(2)}$ also performs poorly for the c and the LSI.

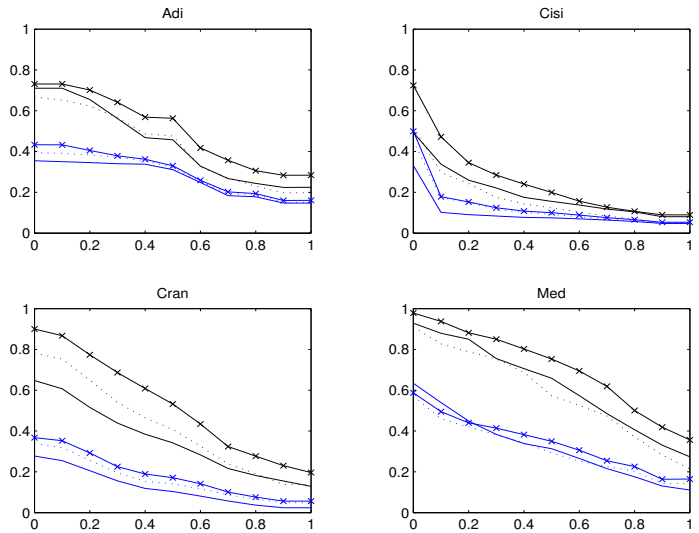

Figure 6: Recall-precision graphs for the best and worst performing weighting combinations for the expanded query measure $c^{(2)}$ (7) in each set. In each plot interpolated average precision for the vector model c (3) (...), the expanded query measure (x-) and the LSI [4],[9] (–) are shown.

A Weighting combinations

Local and global weigthing combinations						
bxx	bfx	bgx	bex	bnx	bn_1x	$(bn_\infty x)$
txx	tfx	tgx	tex	tnx	tn_1x	$tn_\infty x$
lxx	lfx	lgx	lex	lnx	ln_1x	$ln_\infty x$
nxx	nfx	ngx	nex	nnx	nn_1x	$nn_\infty x$
Local and normalization factor combinations						
bxc	bxn_1	(bxn_∞)	txc	txn_1	txn_∞	
lxc	lxn_1	lxn_∞	nxc	nxn_1	nxn_∞	
Local, global and normalization factor combinations						
bfc	bfn_1	bfn_∞	bgc	bgn_1	bgn_∞	
bec	ben_1	ben_∞	bnc	bnn_1	bnn_∞	
bn_1c	bn_1n_1	bn_1n_∞	$(bn_\infty c)$	$(bn_\infty n_1)$	$(bn_\infty n_\infty)$	
tfc	tfn_1	tfn_∞	tgc	tgn_1	tgn_∞	
tec	ten_1	ten_∞	tnc	tnn_1	tnn_∞	
tn_1c	tn_1n_1	tn_1n_∞	$tn_\infty c$	$tn_\infty n_1$	$tn_\infty n_\infty$	
lfc	lfn_1	lfn_∞	lgc	lgn_1	lgn_∞	
lec	len_1	len_∞	lnc	lnn_1	lnn_∞	
ln_1c	ln_1n_1	ln_1n_∞	$ln_\infty c$	$ln_\infty n_1$	$ln_\infty n_\infty$	
nfc	nfn_1	nfn_∞	ngc	ngn_1	ngn_∞	
nec	nen_1	nen_∞	nnc	nnn_1	nnn_∞	
nn_1c	nn_1n_1	nn_1n_∞	$nn_\infty c$	$nn_\infty n_1$	$nn_\infty n_\infty$	

Table 5: Weighting combinations used for the term document matrices. The weightings surronded by parentheses have no effect and are not used.

Local and global weighting combinations						
bxx	bfx	bgx	bex	bnx	bn_1x	$(bn_\infty x)$
txx	tfx	tgx	tex	tnx	tn_1x	$tn_\infty x$
lxx	lfx	lgx	lex	lnx	ln_1x	$ln_\infty x$
nxx	nfx	ngx	nex	nnx	nn_1x	$nn_\infty x$

Table 6: Weighting combinations used for the queriy vectors. The weighting surrounded by parentheses has no effect and is not used.

B Data sets

E.A. Fox at the Virginia Polytechnic Institute and State University has assembled nine small test collections in a CD-ROM. These test collections have been used heavily throughout the years for evaluation of information retrieval systems and they provide a good setting for preliminary testing. Among these nine sets we used four for our evaluation.

Adi Adi is a very small test collection of document abstracts from library science and related areas.

Cisi The data set consist of document abstracts in library science and related areas extracted from Social Science Citation Index by the Institute for Scientific Information.

Cran The Cranfield collection is a small collection with a large number of queries. The data set consist of document abstracts in aerodynamics originally used for tests at the Cranfield Institute of Technology in Bedford, England.

Med The Medline set is a small collection with a small number of queries. It has been extensively used in the past. The documents are abstracts in biomedicine received from the National Library of Medicine.

For a further summary on test sets see [1]. See also Fox [10].

Documents and queries are represented as vectors. Before the representation can be constructed a list of index terms must be compiled for each set. A list of all words (non-zero length strings of characters(A-Z,a-z) delimited by white space) found in the documents was constructed. Each word occurring on the SMART [19] stop list was removed. The remaining words form the set of index terms.

Table 7 summarizes some characteristics of the data sets and queries. All of these sets are rather small in size. For all the sets a large portion of the documents are relevant to some query. For all but the Medline set there are documents that are relevant to more than one query. All sets have more terms than documents and in general there are more terms per document than documents per term. All document vectors are longer than the query vectors.

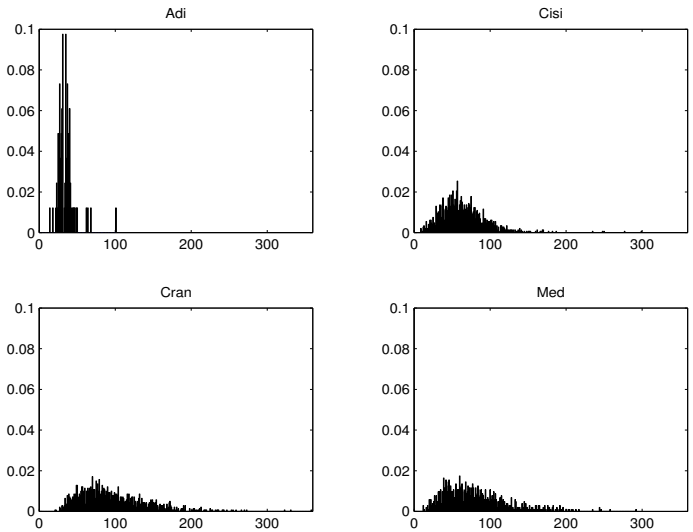

Figure 7: Portion of documents (y-axis) versus length of documents (x-axis) for the data sets.

In Cisi, Cran and Med the length of the documents (length is measured by number of terms) are more spread out than for the Adi set (see figure 7). This is probably due to the small size of the set. A few zero length documents appear in the Cranfield set.

	Adi	Cisi	Cran	Med
no of docs	82	1460	1400	1033
no of indexing terms	1311	10325	7776	12194
no of queries	35	35	225	30
no of relevant documents	72	467	924	696
no of <query,relevant doc> pairs	170	1742	1838	696
max/min/avr no of terms in docs	101/14/35	299/9/65	358/0/95	292/12/80
max/min/avr no of docs per term	44/1/2	644/1/7	703/0/11	262/1/5
max/min/avr no of terms in queries	13/3/7	18/3/8	21/4/9	23/2/11
nonzero elements in matrix (%)	2.1	0.48	0.79	0.46

Table 7: Some characteristics of the data sets Adi, Cisi, Cran and Med.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley, 1999.

[2] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.

[3] M. W. Berry and M. Browne, *Understanding Search Engines, Mathematical modeling and text retrieval*, SIAM, 1999.

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

[5] K. Blom and A. Ruhe, *Information Retrieval using a Krylov Subspace method*, submitted for publication, (2003).

[6] C. Buckley, A. Singhal, M. Mitra, and G. Salton, *New retrieval approaches using SMART: TREC 4*, in Proceedings of the Fourth Text Retrieval Conference (TREC-4), D. Harman, ed., Department of Commerce, National Institute of Standard and Technology. NIST special Publication, 1996, pp. 500–236.

[7] W. B. Croft, *Experiments with representation in a document retrieval system*, Information Technology: Research and Development, 2(1) (1983), pp. 1–21.

[8] S. T. Dumais, *Improving the retrieval of information from external sources*, Behavior Research Methods, Instruments, & Computers, 23 (1991), pp. 229–236.

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

[10] E. A. Fox, *Characterization of two new experimental collections in computer and information science containing textual and bibliographical concepts*, Tech. Rep. 83-561, `http://www.ncstrl.org`, 1983.

[11] W. B. Frakes and R. Baeza-Yates, *Information Retrieval, Data Structures and Algorithms*, Prentice Hall, 1992.

[12] G. Golub and C. F. van Loan, *Matrix Computations 3rd edition*, Johns Hopkins, 1996.

[13] D. Harman, *Ranking algorithms*, in Information Retrieval, data structures and algorithms, W. B. Frakes and R. Baeza-Yates, eds., Prentice Hall, 1992, pp. 363–392.

[14] ——, *Appendix*, in The Eighth Text REtrieval Conference (TREC-8), D. Harman, ed., Department of Commerce, National Institute of Standard and Technology. NIST special Publication, 2000, p. A1.

[15] T. G. Kolda, *Limited-memory matrix methods with applications*, PhD thesis, Applied Mathematics University of Maryland, 1997.

[16] T. G. Kolda and D. P. O'Leary, *A semi-discrete matrix decomposition for latent semantic indexing in information retrieval*, ACM Transactions on Information Systems, 16 (1998), pp. 322–348.

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

[18] G. Salton and C. Buckley, *Term-weighting approaches in automatic text retrieval*, Information Processing & Management, 24 (1988), pp. 513–523.

[19] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[20] A. Singhal, G. Salton, M. Mitra, and C. Buckley, *Document length normalization*, Tech. Rep. TR95-1529, Department of Computer Science, Cornell University, Ithaca, NY, 1995.

a Krylov Subspace method meets TREC

Katarina Blom

April 12, 2004

Abstract

We expect a lot from our search engines. We ask them vague questions about topics that we are unfamiliar with ourselves and we anticipate an organized response. In this report we will follow one topic from the TREC collection. The topic is interesting in this way. The user asks for documents relevant to three terms with high search value, and expects the search engine to give back documents from two groups of relevant documents, some documents where these terms appear and some where they do not appear. We show how the Krylov method used for IR is able to indicate a (weak) connection between the groups of relevant documents. We also show how simple modifications of the method can be used to decrease the scoring for irrelevant documents.

keywords TREC, Krylov subspace, Information Retrieval

1 Introduction and Contence

An information retrieval (IR) system matches user queries (formal statements of information needs) to documents stored in a database.

We look at the document collection as a huge term document matrix A, where there is one row for each term that occurs anywhere in the collection and each column represents one document. The value stored in each matrix element defines a nonzero weight if a term occurs in a document. If a term is not present in the document the corresponding value is zero. The queries will be expressed by the same terms as the documents, i.e. as a column vector q with a nonzero value for each term appearing in the query. There

are of course several ways to set up the term document matrix (choice of stop words[1], choice of weights for the nonzero elements in the matrix etc.).

Using a term document matrix A, query matching can be viewed as a search in the column space of A, and one of the most common similarity measures for scoring the documents is to measure the angles between the query vector and each document vector in A. In section 2.1 we discuss how we choose to set up the term document matrix for the TREC FT ([6]) set used in this report. We also discuss some properties of this matrix.

The *Krylov method* we use for IR is a subspace method based on Krylov sequences of subspaces reachable from the query vector [3]. The Krylov method is briefly presented in section 3 and some more details about the method can be found in appendix A.

In section 4 we follow one query (topic) from the TREC collection. The topic asks for documents relevant to polygamy, polygyny and polyandry, three terms with high search value. The documents that are relevant to the topic fall into two groups, those where any of the terms appear, and those where none of the terms appear. Clearly scoring the documents only by measuring the angles between the document vectors and the query vector will not capture all relevant documents.

Moreover the document vectors from one group are almost orthogonal to document vectors from the other group, but there is a weak connection between three documents in one of the two groups and one document in the other group. In experiments we show that the Krylov method is able to spot this relation. With the help of relevance feedback we are able to retrieve the relevant documents from a group where all vectors were orthogonal to the query vector.

The topic we follow not only describes what documents are relevant to the topic, it also describes what documents are irrelevant. In section 4.1.2 we show how a modified Krylov method [2] can be used in order to decrease the scoring for such (irrelevant) documents.

1.1 Notation

The notation used in this report is rather standard in numerical linear algebra We use uppercase letters for matrices and lowercase letters for vectors. Low-

[1] A stop word is a term whose frequency and/or semantic use makes it of no value as a searchable word.

ercase Greek letters usually denotes scalars. Component indices are denoted by subscript. For example, a vector c and a matrix M might have entries c_i and m_{ij} respectively. On the occasions when both an iteration index and a component index are needed, the iteration is indicated by a parenthesised superscript, as in $c_j^{(r)}$ to indicate the jth component of the rth vector in a sequence. Otherwise c_j may denote either the jth component of a vector c or the jth column of a matrix C. The particular meaning will be clear from its context.

1.2 Measures

The retrieval efficiency of an IR system depends on two main factors. The ability of the system to retrieve relevant information and the ability to dismiss irrelevant information. The ability to retrieve relevant information is measured by *recall*, the ratio of relevant documents retrieved over the total number of relevant documents for that query. A systems ability to reject irrelevant documents is measured by *precision*, the ratio of the number of relevant documents retrieved for a given query over the total number of documents retrieved. Precision and recall are usually inversely related (when precision goes up, recall goes down and vice versa).

When we evaluate a query q all documents are ranked and we recieve an ordered list ℓ of documents. Assume t documents are relevant to the query and let $l_i, i = 1 \ldots t$ be the position for the ith relevant document in ℓ. The *average precision* (non interpolated) for a single query is defined as

$$\frac{1}{t}\sum_{i=1}^{t}\frac{i}{l_i}.$$

The *mean average precision* for multiple queries is defined as the mean of the average precisions for all queries.

For further details, see Harman [6].

2 Data sets

The Text Retrieval Conferences (TREC) were created by the Defence Advance Research Projects Agency (DARPA) and the National Institute of Standards and Technology (NIST). The goal was to overcome the problems

of not having a common base for experimentation and also to provide test sets of a reasonable large size. TREC provides large, diverse test data sets available to anyone interested in using it as a basis for their testing. Since 1992 they also provide a yearly conference to share results between different researchers.

The TREC 4 disc, which we have been using, contains 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 consist of 210,158, 55,630 and 27,922 documents respectively. For our experiments we have used the FT collection with the 150 (ad hoc) queries no 301–450 making our experiments comparable with TREC conferences 7&8 [6].

TREC relevance judgments are made through a process known as pooling. The top 100 documents from runs submitted to TREC each year are combined into a single "pool". The group who created the topic then judges each document in the pool for relevance.

2.1 Term document matrix

The elements of a *term document matrix* A are the occurrences of each word in a particular document, i.e.

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

where a_{ij} is nonzero if term i occurs in document j, zero otherwise. Global, local weightings and normalization factors are applied to increase/decrease the importance of terms within and among documents. Often $a_{ij} = g_i l_{ij} d_j$ where l_{ij} is the local weighting for term i in document j, g_i is the global weighting for term i and d_j is the document normalization factor for document j.

Since every term does not normally appear in each document, the term document matrix is (very) sparse. A few terms, however, appear in all (or almost all) documents. These words have no discrimination value during a search and are called stop words. A *stop list* consists of terms whose frequency and/or semantic use make them of no value as searchable words. Eliminating the words appearing on the stop list usually decreases the total amount of words used in the database dramatically.

Preparing the matrix Minimal preprocessing on the raw text of the FT documents were done. All control sequences were removed (i.e. any text within < > delimiters). Upper case characters were replaced by lower case and white space were used to delimit terms. All non-zero length character sequences from (a-zA-z) were used as terms[2]. Defined in this way we found $230,173$ unique terms in the document collection.

A stop list consisting of all terms that appear in more than 10% of all documents from the FT set will have 299 terms and seems to be a good choice. The stop list will in general consist of common terms with no search value such as the and and. Removing the stop words will not decrease the size of the matrix much, but the number of nonzero elements will decrease significantly by 37%[3]. Clearly eliminating 37% of the nonzero elements will improve storage efficiency.

The most expensive operations in the Krylov subspace methods that we use for IR[4] are the matrix vector multiplications involving the term document matrix performed in the BIDIAG procedure. Since the time required for a matrix vector multiplication between a sparse matrix and a vector heavily depends on the number of nonzero elements in the matrix, we will also gain execution efficiency by eliminating the stop words from the term document matrix.

There is little debate on eliminating common words, but there is some discussion on what to do about singletons (words that only appear once or very infrequently in a document or a collection). For the FT collection almost 42%[5] of all terms in the database occur in only one document, thus eliminating these will decrease the size of the term document matrix significantly. For the BIDIAG procedure eliminating the singletons will decrease the length of the basis vectors, thus reducing both storage and time complexity for the procedure.

In this group of terms a lot of misspellings are found, but also a large amount of foregin words and rare names. The misspellings of course have no value as searchable words – but foregin words and names on the other hand

[2]Since we chose to remove digits as well as special characters queries addressing for example telephone numbers, years, dates, decades etc will have no meaning.

[3]from $40,687,915$ to $25,535,648$.

[4]A short description of the methods used are given in section 3. For a more detailed description please see [3].

[5]$96,549$ out of $230,173$ terms

might have high search values[6] even if it is unlikely that such a query is ever made.

Let $A(t,p)$ be an $m \times n$ term document matrix where the rows of $A(t,p)$ correspond to all terms found in a data set and the columns corresponds to the documents. We will partition $A(t,p)$ such that

$$A(t,p) = \begin{bmatrix} & S(t) & \\ A_1 & & A_2 \\ 0 & & D(p) \end{bmatrix} \tag{1}$$

where the rows of $S(t)$ correspond to the t terms in the stop list. The rows in $D(p)$ correspond to the terms in $A(t,p)$ that appear in at most p documents (and the columns are the document vectors corresponding to the documents where these terms appear). The rows of A_1 and A_2 correspond to the terms in $A(t,p)$ that appear in more than p documents but are not on the stop list. The columns of A_1 correspond to the document vectors from $A(t,p)$ that are not in $D(p)$ and the columns of A_2 correspond to those that are. The 0 is the zero matrix.

In figure 1 the partitions (1) of $A(299,1)$ for the FT set are plotted. The matrix $D(1)$ is large in size (left figure) but the portion of nonzero elements is low (right figure). The $S(299)$ matrix on the other hand is small in size but a large amount of elements are found here.

In the term document matrix used for experiments in this report all 299 stop words are removed and all singletons (in $D(1)$ (1)) are kept. We let the entries l_{ij} in the term document matrix be 1 if term i is present in document j, 0 otherwise. In order to deemphasize common words and long documents first the rows and then the columns of the term document matrix are normalized using the Euclidean norm[7]. The row normalization g_i is a global weighting. Elements in the term document matrix corresponding to rare terms in the data set are given higher values than elements corresponding to common terms. The column normalization d_j will give high weights to rare terms in a particular document. Terms that are common in a particular document will get low weight.

Our way of constructing the term document matrix has several draw

[6]A term has high search value if it is rare among the documents and if a query asking for documents with this term appearing is likely to address this term.

[7]The column normalization will destroy the previous row normalization but not completely. Some deemphazising effect of common terms still remains.

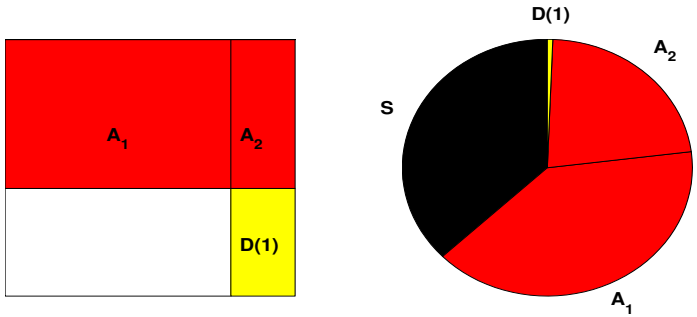

Figure 1: Partitioning of A(299,1) for the FT set. **Left figure** shows sizes of the zero matrix (white), D(1) (light grey), $[A_1 A_2]$ (dark grey) and on top $S(299)$ (black). The black field is small compared to the other fields so it is not visible. **Right figure** shows how the nonzero elements are distributed between $S(299)$ (black), $[A_1 A_2]$ (dark grey) and $D(1)$ (light grey).

backs. We do not use language specific techniques such as stemming[8], phrases, syntactic or semantic parsing, spell checking or correction, proper noun identification, a controlled vocabulary, a thesaurus[9] or any manual indexing. Our term weighting algorithm[10] is simply a row- followed by one column normalization of the 0/1 term document matrix.

We are aware of that retrieval efficiency for the methods used in this report might increase if the term document matrix is constructed with greater care. For a further discussion on weightings used with the Krylov method please see [1].

[8] In stemming the terms are represented by their stems. For example the stem comput could associate computable, computability, computation, computational, computed, computing, computer etc.

[9] A thesaurus is typically a one level or two level expansion of a term to other terms that are similar in meaning. For example the word computer may be linked to computer hardware and computer software.

[10] A nice summary of different term weighting methods can be found in the book by Frakes and Baeza-Yates [4]

2.2 The Queries

The queries for the TREC collections are called topics and have three parts. The *title*, a *description* that summarizes the topic and a *narrative* that further describes the query.

The titles and the description fields for the topics are often short. For example topic no. 316 used for this report has the title

> Polygamy Polyandry Polygyny

and the description field

> A look at the roots and prevalence of polygamy in the world today.

Titles and descriptions typically give information about which documents are relevant. We refer to such information as *positive information*. A query, consisting of terms from these fields, has a chance of bringing in terms that also would be found in relevant documents.

The narrative part of the queries often specifies what documents will be considered relevant and what documents are irrelevant. For example the narrative field from the same topic

> Polygamy is a form of marriage which permits a person to have more than one husband or wife. Polyandry refers to one woman sharing two or more husbands at the same time. Polygyny refers to one man sharing two or more wives at the same time. Primary focus of the search will be the prevalence of these practices in the world today and societal attitudes towards these practices. Also relevant would be discussions of the roots and practical sources of these customs. A modern development in this area is serial polygamy, a phrase coined to label the practice of men who take a series of wives in sequence as a solution to practical welfare, considerations of child care, housing, etc. Documents discussing serial polygamy will not be considered relevant.

Of course constructing a query vector by taking into account the terms in all three fields would bring in terms such as documents and relevant which are not relevant search terms for this query, it will also bring in terms matching irrelevant documents such as serial polygamy. Information on what documents are irrelevant we will refer to as *negative information*.

For this report we have constructed query vectors in two ways. The first approach is to assume that all information is positive and form *positive query vectors*. The positive query vectors were constructed using all the terms from the title only, using all terms from the title and the description, and using all terms from both the title, description and the narrative respectively. We will refer to these three query vectors as *short*, *middle* and *long* query vectors respectively. The query vectors are constructed as 0/1 vectors, the same way as the documents in the term document matrix, i.e. by letting a 1 in position j denote the presence of term j and 0 in position j denote absence of term j. The query vectors are normalized using euclidean norm (no global weighting is used). We let the terms represented in the term document matrix determine the terms in the collection.

The second approach is to construct *negative query vectors* that take into account the negative information i.e. which documents to leave out. The terms for the query vector with the negative information are picked by hand using information from the narrative field. Again a 1 in position i indicates presence of term i and a 0 absence. In the BIDIAG procedure we will avoid irrelevant documents by orthogonalizing against negative query vectors.

2.3 FT Term document matrix and query vectors

In table 1 the maximum, minimum and mean number of terms in document vectors and query vectors are listed. Although the topics are in general much shorter than the documents there exist topics that are longer than a few documents. However a large majority of the documents are longer than the topics. Common terms tend to appear more frequently in topics than in documents. When removing all terms appearing in more than 10% of the documents[11] the mean number of terms for the documents shrinks by roughly 37%[12]. The mean number of terms in the query vectors will shrink by 56% / 50% / 50% [13] for long/middle/short query vectors.

Rare terms are more common in the documents than in the topics. 22%[14] of the documents have at least one term that appears in only one document. At least one of these terms appears in 7% / 3% / 3% [15] of the

[11] 299 terms appears in more than 10% of the documents
[12] From $40,687,916$ to $25,535,649$.
[13] From 44/16/4 to 25/8/2 terms.
[14] $46,560$ out of $210,158$ documents.
[15] 10/5/4 (out of 150/150/150).

	max/min/mean number of terms	
	10% most common terms removed	no stop list
document vector	3227/3/122	3266/8/194
long query vector	61/8/25	98/18/44
middle query vector	27/1/8	48/6/16
short query vector	19/1/2	34/2/4

Table 1: Maximum, minimum and mean number of terms in document vectors and long, middle and short query vectors.

short/middle/long queries.

Due to the row and column normalizations of the term document matrix these terms are weighted high, and thus the presence of such term in the query vector will rank the corresponding document high. This is exactly what we want if the document is relevant to the topic and exactly what we would like to avoid if the document is irrelevant.

3 The Krylov subspace method for Information retrieval

Query matching can be viewed as a search in the column space of the term document matrix A. One of the most common similarity measures used for query matching is to measure the angle between the query vector and the document vectors in A. The smaller the angle is the more relevant the document is. In the *vector model* the cosines between the query vector q and document vectors a_j are used to score the documents in relevance order,

$$c_j = \frac{q^T a_j}{\|q\|_2 \|a_j\|_2}, \quad j = 1, \ldots, n. \tag{2}$$

For the Krylov subspace methods we will use the Golub Kahan bidiagonalization procedure [5] applied to the term document matrix A starting with the query vector q to receive the two *basis matrices* Q_{r+1} and P_r and the $(r+1) \times r$ lower bidiagonal matrix B_{r+1} satisfying $B_{r+1} = Q_{r+1}^T A P_r$:

$$[Q_{r+1}, B_{r+1}, P_r] = \text{BIDIAG}(A, q, r) \tag{3}$$

The column vectors in the basis matrices Q_{r+1} and P_r span bases for the two Krylov subspaces $\mathcal{K}_{r+1}(AA^T, q)$, in the document space (spanned by

the query q and the columns of A) and $\mathcal{K}_r(A^TA, A^Tq)$, in the term space (spanned by the rows of A) respectively. The *reached subspace* W forms an orthonormal basis for the column vectors in AP_r. The BIDIAG procedure is further described in appendix A.

Sometimes we want to avoid irrelevant information by making all document vectors in Q_{r+1} orthogonal to some vector $q^{(-)}$. Technically it is simple to rewrite the BIDIAG procedure to incrementally compute vectors q_i in Q_{r+1} orthogonal to $q^{(-)}$. The procedure

$$[Q'_{r+1}, B'_{r+1}, P'_r] = \text{BIDIAG}_o(A, q, q^{(-)}, r) \tag{4}$$

will compute two basis matrices Q'_{r+1} and P'_r and an $r+1 \times r$ lower bidiagonal matrix B_{r+1}. Both the procedures BIDIAG and BIDIAG$_o$ are further described in appendix A.

3.1 The expanded query measure for document ranking

The reached subspace W, the basis matrices Q_{r+1}, P_r and the B_{r+1} matrix from the BIDIAG are used to score the documents in relevance order to the query (see Blom Ruhe [3]). In this report the *expanded query measure* is used for ranking the documents. An expanded query is

$$\hat{q} = WW^Tq \tag{5}$$

the projection onto the reached subspace. In the *expanded query measure* the documents are sorted measuring the cosine of the angle between the projected query $\hat{q}$ and each document vector in A,

$$c_j^{(2)} = \frac{\hat{q}^Ta_j}{\|\hat{q}\|_2\|a_j\|_2}, \; j = 1, \ldots, n. \tag{6}$$

With the expanded query measure we are able to find document vectors that are orthogonal to the query vector.

Let the $m \times n$ term document matrix $A = \begin{bmatrix} M & X \end{bmatrix}$ where the columns of $M = \begin{bmatrix} m_j \end{bmatrix}$ correspond to d relevant documents and the columns of $X = \begin{bmatrix} x_j \end{bmatrix}$ correspond to the rest of the document vectors in A. Assume q is the query vector and that q is orthogonal to all document vectors in M, thus $q^TM = 0$. (This is precisely the situation we have with the topic we follow in section 4).

Let $\hat{q} = WW^Tq$ be the projected query vector (5). With $r = 1$ in the BIDIAG procedure (appendix A) the reached subspace

$$W = \text{span}(AP_1) = \text{span}\{AA^Tq\} = \frac{XX^Tq}{\|X^Tq\|_2},$$

where the last equality follows from the orthogonality between q and the columns in M. Let $c = X^Tq$ and $\gamma = \dfrac{1}{\|c\|_2}$, the expanded query measure (6) then becomes

$$c_j^{(2)} = \frac{\hat{q}^Ta_j}{\|\hat{q}\|_2\|a_j\|_2} = \begin{cases} \gamma\dfrac{c^TX^Tm_j}{\|m_j\|_2} & \text{if } 0 < j \leq d \\[2ex] \gamma\dfrac{c^TX^Tx_{j-d}}{\|x_{j-d}\|_2} & \text{if } d < j \leq n \end{cases}$$

In order to score the relevant document vectors in M high we want the first d elements in $c^{(2)}$ to be large (and the last $n-d$ elements in $c^{(2)}$ to be small). This is true if there are document vectors in X that are close both to the query vector and to the relevant document vectors in M. So at least in theory document vectors from M could be scored high.

If the relevant document vectors in M are orthogonal to both the query vector and the rest of the document vectors in X (i.e. $M^Tq = 0$ and $M^TX = 0$) the two basis matrices Q_{r+1} and P_r (3) from the BIDIAG procedure will span bases for the two Krylov subspaces

$$\mathcal{K}_{r+1}(AA^T, q) = \{q, XX^Tq, \ldots, (XX^T)^rq\}$$

and

$$\mathcal{K}_r(A^TA, A^Tq) = \{\begin{bmatrix} 0 \\ X^Tq \end{bmatrix}, \ldots, \begin{bmatrix} 0 \\ (X^TX)^{r-1}X^Tq \end{bmatrix}\}.$$

respectively. No directions from the misses documents in M will be in these subspaces, thus we will not find the misses using the bidiagonalization procedure in this case.

4 Experiments

Topic number 316 deals with polygamy, polyandry and polygyny and was presented in section 2.2. According to the relevance judgements 19 docu-

	q	doc_1*	doc_2	doc_3	doc_4*	doc_5*	doc_6*	doc_7*	doc_8
polygamy	0.58	0	0	0	0.22	0.22	0.20	0.17	0.16
polyandry	0.58	0.77	0	0	0	0	0	0	0
polygyny	0.58	0	0.62	0.46	0	0	0	0	0

Table 2: Weights for the terms in the short query vector and for the 8 top scored documents using the vector model (2). The documents marked with a star are relevant to the topic doc_1, doc_4, doc_5, doc_6 and doc_7 correspond to document FT932–4228, FT934–6885, FT943–10242, FT943–5141 and FT942–4193 respectively.

ments from the FT set are relevant to this topic[16]. Two of the relevant documents are identical[17].

The three terms polygamy, polyandry and polygyny appear in the short query vector. These are rare terms of high search value. polygamy appears in 14 documents and 11 of these are relevant. polyandry is in 1 (relevant) document and polygyny is in 2 (irrelevant) documents. (Clearly removing singletons will soon make the short query vector empty and terms with high search value will be removed from the document vectors.).

If using the vector model (2) the short query vector addresses 17 document vectors (corresponding to columns of A_2 (1)) and is orthogonal to the rest of the documents in the set. 12 of the documents addressed by the query vector are relevant to the topic and average precision for the vector model becomes 0.45.

The weights for the terms polygamy, polyandry and polygyny, appearing in the short query vector, for the 8 top scored documents are listed in table 2. The effect of the row normalization of the term document matrix is clearly seen. The highest weight (0.77) corresponds to the most rare of the three terms (polyandry). The weights for the second most rare term (polygyny) are 0.62 and 0.46 respectively. For the least rare term (polygamy) weights are smaller. The effect of the column normalization is not that clear. The weight for the term polygyny in doc_3 is lower because more rare terms appear in doc_3 than in doc_2. doc_3 is shorter than doc_2.

[16]Documents FT922–11381, FT922–12843, FT931–5366, FT931–6791, FT932–1167, FT932–3422, FT932–3656, FT932–4228, FT932–5625, FT933–7689, FT934–6885, FT942–4193, FT943–10242, FT943–2362, FT943–5141, FT944–1831, FT944–2037, FT944–2863 and FT944–8467 are relevant to topic 316.

[17]FT944–1831 and FT944–2037 are identical.

Since only one of the three terms in the short query vector appear in each document, the scoring follows the weights of the terms directly.

The relevant documents can be divided into two distinct groups, *retrieved relevant* and *misses*[18]. In the former group (retrieved relevant) one of the terms polygamy or polyandry appear. All the documents in the later group (misses) are orthogonal to the short query.

The 12 retrieved relevant documents are scored

$$[1, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16]. \tag{7}$$

and the 7 misses were scored[19]

$$[\infty, \infty, \infty, \infty, \infty, \infty, \infty].$$

The challenge is to see whether we can locate the 7 misses[20] using the Krylov subspace method (3).

4.1 Using the Krylov subspace method

Using the short query vector The short query vector is orthogonal to the misses, and using it as a starting vector for the BIDIAG procedure (3) will start a search orthogonal to the documents we want to locate. Unless the document vectors from the two groups of relevant documents are close in angles, iterating in the BIDIAG procedure will not bring in relevant documents from the misses group.

In table 3 terms and weights for the 10 top weighted terms in the projected query vector (5) are listed[21]. The weights for terms in the 8 top scored documents are also shown in the table.

[18]The retrieved relevant documents are the documents that are scored well enough for a human user to judge and the misses are the relevant documents scored so low that they will not be shown to the user (i.e. one cannot expect a user to read through all previous scored documents). In [] two kinds of retrieval failures were used, *false alarms* and misses. False alarms are highly scored irrelevant documents.

[19]We mark the orthogonality by using the scoring ∞.

[20]All relevant documents carry one of the terms polygamy, polygamist, polygamists and polyandry respectively. A search vector addressing all these four terms will score all relevant documents below 23. Thus if stemming had been used for the term document matrix or if the query vector had been expanded with polygamist and polygamists all the relevant documents had been captured using the vector model.

[21]We used the short query vector and constructed projected query vectors $\hat{q}$ (5) for $r = 1, \ldots, 6$ in BIDIAG. The top 10 weighted terms in the projected query vector that were brought in by the procedure dominated all 6 query vectors computed.

The terms polyandry, polygyny and polygamy still show high weights. Naturally all three terms have lower weights in $\hat{q}$ than in q. Also note that the most rare of the three terms (polyandry) has higher weight than polygyny, the second most rare term. The most common of the three terms polygamy has least weight.

The new terms that were brought in by the bidiagonalization scheme (gamy, supergrass, annemarie, telltale, bigamists, spinsters and matings) do not lead us to the wanted documents. doc_1 (relevant) lists different TV-shows and the words supergrass, annemarie and telltale comes from listed TV-shows that have nothing to do with polyandry. None of these terms appear in any other relevant documents. The words gamy, spinsters and matings appear in the non relevant documents doc_2 and doc_3. These terms do not appear in any relevant document.

Relevant documents from the two groups retrieved relevant and misses were scored (sorted as in 7)

$$[2, 7, 4, 5, 6, 9, 10, 21, 27, 31, 20, 33]$$

and

$$[579, 2981, 2965, 5113, 15071, 24559, 24560] \tag{8}$$

respectively. Even though average precision decreased from 0.45 to 0.33 all documents in the first group are scored below 33. The misses are not orthogonal to the projected query vector – so it is possible to score them. However they still do not belong to the group of retrieved relevant documents. The two document from the misses group that were scored 24559 and 24560 are identical.

Document vectors from the two groups retrieved relevant and misses are almost orthogonal. Let the first 12 columns of A_r be the retrieved relevant document vectors from the term document matrix and let the last columns be the document vectors corresponding to the 7 misses. The orthogonality is clearly seen in figure 2 where all scalar products[22] in $A_r^T A_r$ greater than 0.01 are plotted. Three documents from the retrieved relevant group share some terms with one of the misses. Although the three scalar products are small (between 0.01 and 0.05) the connection between the two groups can be seen in the scoring (8) where this document is scored 579.

[22]Since all document vectors are normalized $A_r^T A_r$ is the cosines of the angles between the relevant document vectors.

	$\hat{q}$	doc_2	doc_1*	doc_3	doc_5*	doc_6*	doc_7*	doc_4*	doc_8
polygyny	0.44	0.62	0	0.46	0	0	0	0	0
polyandry	0.44	0	0.77	0	0	0	0	0	0
monogamy	0.22	0.36	0	0	0	0	0.25	0	0.24
polygamy	0.20	0	0	0	0.22	0.20	0.17	0.22	0.16
supergrass	0.18	0	0.31	0	0	0	0	0	0
annemarie	0.17	0	0.29	0	0	0	0	0	0
telltale	0.16	0	0.27	0	0	0	0	0	0
skews	0.15	0.33	0	0	0	0	0	0	0
bigamists	0.13	0	0	0	0	0	0	0.82	0
matings	0.13	0	0	0.38	0	0	0	0	0

Table 3: The 10 top weighted terms in the projected query vector $\hat{q}$ (5) and their corresponding weights in $\hat{q}$ and in the 8 top scored documents. The documents were scored using the cosine of the angle between the projected query $\hat{q}$ and each document vector in the term document matrix (6), with the short query vector and $r = 3$ in BIDIAG (3). The documents marked with a star are relevant to the topic. The document numbers refer to the numbers in table 2

Also note (see figure 2) that the group of retrieved relevant documents can be divided into two groups having the first document orthogonal to the rest of the documents in the group. In this document the term polyandry appears. Since polyandry appears only once in the whole set the corresponding element is weighted high, and since the term also appears in the query vector the document is scored high (both in the vector model and using the Krylov method). Removing all rows corresponding to terms only appearing once in the set from the term document matrix will make this document vector orthogonal to both the query vector and the rest of the relevant documents. Thus the document will not be captured by the vector model nor by the Krylov subspace method.

False alarms are highly scored irrelevant documents[23]. In some of the false alarms some interesting relations appear.

- A modern phrase serial polygyny labels the practice of women who take

[23]The false alarms are scored 1, 3, 8, 11 – 19, 22 – 26, 28 – 30 and 32.

a series of husbands in sequence as a solution to practical welfare[24]. The two terms serial and polygyny appears in this context in the two irrelevant documents scored 1 and 3. Using the technique discussed in section 4.1.2 (by orthogonalizing against negative query vectors in the BIDIAG procedure) these two irrelevant documents may be moved further down in the ranking list. Let for example the elements in the negative query vector address the terms serial and polygyny.

- The irrelevant document scored 8 lists the today's television. One of the programs listed examines polygamy versus monogamy. The document were brought in because of polygamy appearing in it.

- In the two relevant documents scored 4 and 9 the term polygamy appears. The documents describes a Malaysian sect that uses polygamy. The false alarms scored 11, 12, 13 and 17 are short news telegrams about the Malaysian sect, clearly brought in because of its closeness to the relevant documents scored 4 and 9. In the false alarm scored 29 the sect is mentioned. (None of the terms in the short query vector appear in the irrelevant documents scored 11, 12, 13, 17 and 29.)

- The false alarms scored 14, 15 and 18 are about a Malaysian politician, brought in because of it closeness to the relevant documents scored 4 and 9 and the irrelevant documents scored 11, 12, 13 and 17 above.

- The false alarm scored 26 is about a family running an illegal bomb factory and among the TV shows listed in the (relevant) document scored 2 there is one program about this family.

- The terms skews and polygyny appear in the false alarm scored no. 1. The document were scored high because of the term polygyny appearing in it and it brings in the rare and high weighted term skews[25] to the projected query vector. skews also appears in the false alarms scored 19 and 30.

[24] In the narrative field of the topic it is stated that documents discussing the male equivalence serial polygamy will not be considered relevant.

[25] skews appears in 7 documents.

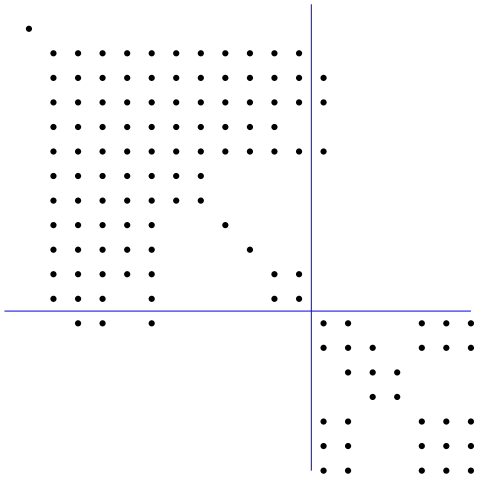

Figure 2: Nonzero elements greater than 0.01 in $A_r^T A_r$ where the 12 first columns of A_r are the retrieved relevant document vectors and the 7 last columns correspond to the misses. Columns in A_r are sorted as (7) and (8).

Using the middle and long query vectors None of the middle or long query vectors are orthogonal to all the misses (but the angles are large). The query vectors are still rather close to at least a few document vectors in the relevant retrieved group. Since the queries are longer the terms polygamy, polyandry and polygyny get lower weights, thus relevant documents from the relevant retrieved group above are scored worse compared to the scoring when the short query vector was used.

Relevant documents from the group retrieved relevant for the middle query vector using the vector model were scored (sorted as in 7)

$$[1, 14, 12, 16, 24, 28, 51, 90, 98, 113, 157, 205]$$

and for the long query vector

$$[3, 385, 169, 416, 430, 887, 728, 3143, 3199, 3989, 5139, 11607]$$

respectively. Iterating a few steps in the BIDIAG procedure will not improve the scoring for the relevant documents. In table 4 the top 10 weighted terms that are brought in after three iterations with the BIDIAG to the projected middle and long query vectors are listed. These differ from the terms that were brought in by the short query vector (listed in table 3). The important search term polygamy is no longer among the top 10 weighted terms. Due to the naive way of constructing the long query vector terms such as discussing and discussions are brought in.

Terms in the intersection In table 5 all terms that appear at least in 6 of the relevant documents and at least once in both groups, retrieved relevant and misses, of relevant documents. The terms appearing in the intersection are weighted low in the relevant documents and thus will not be useful as search terms in the query vector. Also note that the terms appearing in both groups differ completely from the terms appearing in the projected queries.

Summary All three query vectors, short, middle and long, are either orthogonal or almost orthogonal to all documents in the misses group and close in angles to the documents in the relevant retrieved group. The documents from the misses group are almost orthogonal to the documents in the relevant retrieved group. Using the BIDIAG procedure we are able to spot the weak relationship between the two groups but in order to capture documents from

$\hat{q}$(middle)		$\hat{q}$(long)	
look	0.51	housing	0.14
prevalence	0.29	discussions	0.13
roots	0.25	customs	0.11
polygyny	0.11	sources	0.10
polyandry	0.11	practices	0.09
column	0.09	series	0.09
lex	0.07	practice	0.09
elixir	0.07	discussing	0.09
grass	0.06	considered	0.09
monogamy	0.06	welfare	0.09

Table 4: Terms and weights for the 10 top weighted projected query vectors for the middle and long query vectors.

party french old home called clear senior once role family man opposition society century men history children books

Table 5: Terms that appear in at least 6 of the relevant documents and at least once in both groups retrieved relevant and misses.

the misses group we need either the starting vector (the query) to address document vectors from both groups or the document vectors from the two groups need to be closer in angles. Clearly in order to find the 7 misses we need to choose the starting vector and use the BIDIAG procedure with greater care.

4.1.1 Relevance feedback

In a relevance feedback cycle, the user is presented a list of retrieved documents, and after examining them, marks those that are relevant. The main idea is to use the information provided by the user to make a new (hopefully) improved search.

Assume a user has given judgements on the documents in the retrieved relevant group (7) from the vector model. Using the term weights from the relevant document vectors we constructed a new (improved) starting vector for the BIDIAG. We let the new query vector be the vector sum of the relevant

	q
polygamy	0.58
bigamists	0.33
polyandry	0.31
taso	0.30
rakai	0.30
magesi	0.25
deviationist	0.24
neziha	0.23
mezhoud	0.23
ashaari	0.21

Table 6: Using the term weights from the relevant document vectors a new query vector was constructed. We let the new query vector be the sum of the row vectors in the document vectors corresponding to the relevant retrieved documents. Terms and weights for the 10 top weighted terms are shown.

retrieved documents.

The terms and the weights (after normalizing) for the top 10 weighted terms in the improved query vector are listed in table 6. These words are rare in the term document matrix and a few differ from the terms for the projected query vector listed in table 3. The five terms taso, magesi, neziha, mezhoud and ashaari comes from the three names Mrs Marble Magesi of Taso, Mrs Neziha Meshoud and Ashaari Muhammad appearing in some of the relevant documents. The names are rare and thus high weighted. Rakai is a southern district in Uganda.

The important search terms polygamy and polyandry appear in the query vector. polygyny appeared in the two irrelevant documents scored 2 and 3 in (7) and is not present. Due to the orthogonality between the two groups misses and retrieved relevant none of the 10 top weighted terms in the improved query vector appear in any of the misses. Also there is no match between the terms in the intersection between the two groups relevant retrieved and misses, listed in table 5, and the most common terms in the improved query vector.

The scorings for the vector model and the Krylov method are listed below

Retrived relevant documents (sorted as in 7)

$$\begin{array}{rl} \text{vector model} & [8, 2, 4, 7, 1, 3, 5, 16, 18, 20, 21, 22] \\ \text{Krylov method} & [16, 9, 4, 5, 7, 12, 10, 31, 33, 36, 24, 37] \end{array}$$

and misses (sorted as in 8)

$$\begin{array}{rl} \text{vector model} & [\infty, \infty, \infty, \infty, \infty, \infty, \infty] \\ \text{Krylov method} & [57, 43346, 116025, 90478, 28075, 46251, 46252] \end{array}$$

The improved query vector manages to capture the document vector in the missing group that is closest to the intersection between the two groups of relevant documents (see figure 2 and scoring (8)) and it is scored 57. If we consider this document to be retrieved we may repeat the process and construct yet an improved query vector by summing the row vectors in the relevant retrieved group and the retrieved document vector from the misses. This will bring in the term polygamist to the query vector.

A query vector with the three terms polygamy, polyandry and polygamist highly weighted will retrieve all but two relevant documents using the vector model. Since the two not retrieved documents are connected to some of the retrieved relevant documents iterating a few steps in the BIDIAG with this query vector will capture at least one of the remaining two relevant documents.

Summary Using the expanded query measure (6) to rank the documents for relevancy and with a carefully picked starting vector for the BIDIAG procedure we are able to retrieve relevant document vectors that are orthogonal to the query vector. In the relevance feedback cycle (section 4.1.1) we used judgements from the user to create a new starting vector. The new starting vector again was orthogonal to the misses, but closer in angles to some non relevant documents that in their turn were close to one of the misses. In this way we were able to steer the Krylov method to further resemble relevant documents.

4.1.2 Using negative information

Documents discussing serial polygamy are not relevant to this topic (see the narrative field for this topic in section 2.2). There is at least one document

dealing with serial polygamy in the FT set. This (irrelevant) document is scored 17 using the vector model with the short query vector (7). Clearly removing the term polygamy will decrease retrieval performance since all of the retrived relevant documents were found merely through this term. By using the BIDIAG$_o$ procedure (4) we are able to orthogonalize against non wanted information. We let the negative query vector $q^{(-)}$ have the terms serial and polygamy and q be the short query vector. Average precision will be 0.33 and the document dealing with serial polygamy is scored 120. (The two groups of retrieved relevant documents and misses still remains).

5 Conclusions

There are usually many ways to express a given concept so the terms in a user's query may not match those of a relevant document. The query vector for topic no 316 used in section 4 is orthogonal to some of the relevant document vectors and thus splits the set of relevant documents into those that are retrieved (retrieved relevant) and those that are not retrieved (misses) when using the vector model scoring (2). However some of the misses are connected to some of the retrieved relevant documents through common terms. With the BIDIAG procedure we are able to indicate this (rather weak) connection between the two sets of document vectors. With some relevance feedback from the user some of the misses were ranked well enough to be retrieved.

Many terms have multiple meanings, so terms in a user's query will match terms in documents that are not of interest to the user. Reformulating the BIDIAG procedure slightly it is possible to orthogonalize against unwanted directions and thus avoid subspaces (or vectors) spanned by terms that is of no interest. The projected query vector (5) used in the expanded query measure (6) will be orthogonal to these subspaces (or vectors) and there by the irrelevant retrieved documents is likely to be ranked further down the list.

A The Golub Kahan bidiagonalization procedure

The Golub Kahan bidiagonalization procedure is a variant of the Lanczos tridiagonalization algorithm and it is widely used in the numerical linear algebra community.

The Golub Kahan algorithm starts with the normalized query vector $q_1 = q/\|q\|$, and computes two orthonormal bases P and Q, adding one column for each step k, see [5] in section 9.3.3.

ALGORITHM BIDIAG(A,q,r):
Start with $q_1 = q/\|q\|$, $\beta_1 = 0$
for $k = 1, 2, \ldots r$ **do**
  $\alpha_k p_k = A^T q_k - \beta_k p_{k-1}$
  $\beta_{k+1} q_{k+1} = A p_k - \alpha_k q_k$
end.

The scalars α_k and β_k are chosen to normalize the corresponding vectors.

Define

$$\begin{aligned} Q_{r+1} &= \begin{bmatrix} q_1 & q_2 & \ldots & q_{r+1} \end{bmatrix}, \\ P_r &= \begin{bmatrix} p_1 & p_2 & \ldots & p_r \end{bmatrix}, \end{aligned}$$

$$B_{r+1} = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & & \alpha_r \\ & & & \beta_{r+1} \end{bmatrix}.$$

After r steps k we have the basic recursions

$$\begin{aligned} A^T Q_r &= P_r B_r^T \\ A P_r &= Q_{r+1} B_{r+1}. \end{aligned}$$

The columns of Q_r will be an orthonormal basis of the Krylov subspace $\mathcal{K}_{r+1}(AA^T, q)$ and the columns of P_r forms an orthonormal basis for the Krylov subspace

$\mathcal{K}_r(A^TA, A^Tq)$. The lower bidiagonal matrix $B_{r+1} = Q_{r+1}^T AP_r$ is the projection of A onto these Krylov subspaces and some of the singular values of B_{r+1} will be approximations of those of A.

Technically it is easy to rewrite the BIDIAG procedure to incrementally compute vectors q_i in Q_{r+1} orthogonal to a vector c.

ALGORITHM BIDIAG$_o(A,q,c,r)$:
Start with $q_1 = q/\|q\|$, $\beta_1 = 0$
for $k = 1, 2, \ldots r$ **do**
  $\alpha_k p_k = A^T q_k - \beta_k p_{k-1}$
  $y = Ap_k - \alpha_k q_k$
  $\beta_{k+1} q_{k+1} = y - cc^T y$
end.

The two calls BIDIAG$_o(A,q,c,r)$ and BIDIAG$((I - cc^T)A,q,r)$ are equivalent. The procedure BIDIAG$_o$ is further discussed in [2].

References

[1] K. BLOM, *Experimenting with different weighting schemes for the Krylov Subspace method used for IR*, tech. rep., Dept. of Mathematics, Chalmers university of Technology, 2003.

[2] ——, *Modified Krylov subspace methods for information retrieval*, tech. rep., Dept. of Mathematics, Chalmers university of Technology, 2003.

[3] K. BLOM AND A. RUHE, *Information Retrieval using a Krylov Subspace method*, submitted for publication, (2003).

[4] W. B. FRAKES AND R. BAEZA-YATES, *Information Retrieval, Data Structures and Algorithms*, Prentice Hall, 1992.

[5] G. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins, 3 ed., 1996.

[6] D. HARMAN, *The Eighth Text REtrieval Conference (TREC-8)*, NIST Special Publication 500–246. http://trec.nist.gov/pubs/trec8/t8_proceedings, (2000), p. A1 (Appendix).