A KRYLOV SUBSPACE METHOD FOR INFORMATION RETRIEVAL.

KATARINA BLOM[*] AND AXEL RUHE[†]

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.

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

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 $A = [a_1, a_2, \ldots, a_n]$.)

[*]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`

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 [8], 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* [6, 13]. One may also apply a probability model. This leads to computing convex combinations of nonegative basis vectors, [12, 2].

The purpose of this contribution is to develop 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 [14] and studied by Xu and Croft [15].

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 [9], 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 [11].

We have formulated our algorithm and got some preliminary results in the licentiate thesis of the first author [3]. Further developments, like term weighting, experiments on more data sets and the inclusion of relevance feed back is discussed in the thesis [4]. Experiments on small matrices are reported in more detail in the conference contribution [5].

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

$$A = U\Sigma V^T \tag{1.1}$$

be the SVD of A, see [10]. The best rank s approximation to A in the Frobenius or sum of squares norm is

$$A^{(s)} = U_s \Sigma_{ss} V_s^T \tag{1.2}$$

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

$$\mathcal{K}_r(C, v) = \text{span}\{v, Cv, C^2v, \ldots C^{r-1}v\}. \tag{1.3}$$

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 [11].

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, [8] uses the singular value decomposition, SVD (1.1) of the term document matrix

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

and choose the space of the leading s singular vectors (1.2)

$$\mathcal{S} = \operatorname{span}[U_s]$$

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 [7]. 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 [16].

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 [6]. Park et al [13] 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* $p = A^Tq$. 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 $p_2 = A^Tq_2$ 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,

$$\mathcal{S} = \mathcal{K}_k(AA^T, q) \tag{2.1}$$

after k steps, see (1.3).

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

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

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 (2.1), our choice of query expansion is related to the technique of Sparck Jones [14]. The second vector in the the Krylov sequence (2.1), $\tilde{q}_2 = AA^T q$, 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 [15].

3. The Krylov subspace algorithm.

We use the Golub Kahan bidiagonalization algorithm [9] to compute the Krylov sequence of subspaces (2.1). 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 [10] 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

$$\begin{aligned} Q_{r+1} &= \begin{bmatrix} q_1 & q_2 & \cdots & q_{r+1} \end{bmatrix}, \\ P_r &= \begin{bmatrix} p_1 & p_2 & \cdots & p_r \end{bmatrix} \\ B_{r+1,r} &= \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_r \\ & & & \beta_{r+1} \end{bmatrix}. \end{aligned} \tag{3.1}$$

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 (2.1),

$$\operatorname{span}[Q_r] = \mathcal{K}_r(AA^T, q) \subseteq \mathcal{R}([A\, q]) \tag{3.2}$$

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

$$\operatorname{span}[P_r] = \mathcal{K}_r(A^T A, A^T q) \subseteq \mathcal{R}(A^T)\,, \tag{3.3}$$

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 (3.2), 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 (3.2) and the column space of A,

$$\mathcal{R}(AP_r) = \operatorname{span}[Q_{r+1} B_{r+1,r}] \subseteq \mathcal{R}(A) \tag{3.4}$$

The basic recursion (3.2) implies that it has the orthonormal basis W_r, where

$$W_r = Q_{r+1} H_{r+1,r}\,, \tag{3.5}$$

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

$$B_{r+1,r} = H_{r+1,r+1} R\,. \tag{3.6}$$

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 (3.5) to project the query and the documents into the reached subspace (3.4). The projected query $\hat{q}$ is

$$\hat{q} = P_{\mathcal{R}(AP_r)} q = W_r W_r^T q = W_r H_{r+1,r}^T e_1 = W_r \begin{pmatrix} h_{1,1} \\ h_{1,2} \\ \vdots \\ h_{1,r} \end{pmatrix} \tag{3.7}$$

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,

$$\hat{a}_j = W_r W_r^T a_j\,. \tag{3.8}$$

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 (3.4),

$$\mathrm{Css}_j^{(r)} = \frac{\hat{q}^T \hat{a}_j}{\|\hat{q}\|_2 \|\hat{a}_j\|_2} \quad j = 1 \ldots n. \tag{3.9}$$

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

$$\mathrm{Cqe}_j^{(r)} = \frac{\hat{q}^T a_j}{\|\hat{q}\|_2 \|a_j\|_2} \quad j = 1 \ldots n. \tag{3.10}$$

We compute these quantities using the basis W (3.5) and the small orthogonal Hessenberg $H_{r+1,r+1}$ (3.6). 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_rS_r forms a new basis of the reached subspace (3.4). 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 $\hat{a}_j$ (3.8) as the norm of the remaining elements in $y_j^{(r)}$. Thus the subspace cosine (3.9) is

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

while the query expansion cosine (3.10) 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 (3.10) of the angles between projected query and original documents for scoring, often gives better performance than the subspace cosines Css_j (3.9), 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 3.1 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

$$\min_x \|Ax - q\|_2, \tag{3.11}$$

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 [10]). 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 (3.11) 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 (3.11) 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 $\mathrm{Cqe}_j^{(k)}$ (3.10), tends to the scoring of the vector model. This is precisely what we see in figure 3.1. 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 3.1, 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 $\hat{q}^{(k)}$ is

$$\begin{aligned} d^{(k)} &= q - \hat{q}^{(k)} \\ &= Q_{k+1}e_1 - Q_{k+1}H_{k+1,k}H_{k+1,k}^T e_1 \\ &= Q_{k+1}(I - H_{k+1,k}H_{k+1,k}^T)e_1 \\ &= Q_{k+1}h_{k+1}^{(k)}h_{k+1}^{(k)T}e_1 \\ &= Q_{k+1}h_{k+1}^{(k)}h_{1,k+1}^{(k)} \end{aligned} \tag{3.12}$$

and its norm is just

$$\|d^{(k)}\| = |h_{1,k+1}^{(k)}| \tag{3.13}$$

The normal equation residual is

$$\begin{aligned} A^T d^{(k)} &= A^T Q_{k+1}h_{k+1}^{(k)}h_{1,k+1}^{(k)} \\ &= P_{k+1}B_{k+1,k+1}^T h_{k+1}^{(k)}h_{1,k+1}^{(k)} \\ &= P_{k+1}\begin{pmatrix} B_{k+1,k}^T \\ 0 \quad \alpha_{k+1} \end{pmatrix} h_{k+1}^{(k)}h_{1,k+1}^{(k)} \\ &= P_{k+1}\begin{pmatrix} 0 \\ \alpha_{k+1}h_{k+1,k+1}^{(k)} \end{pmatrix} h_{1,k+1}^{(k)}. \end{aligned} \tag{3.14}$$

Its norm is

$$\|A^T d^{(k)}\| = |\alpha_{k+1}h_{k+1,k+1}^{(k)}h_{1,k+1}^{(k)}|. \tag{3.15}$$

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 $P_r^T P_r = I$ (3.1). In standard

[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.

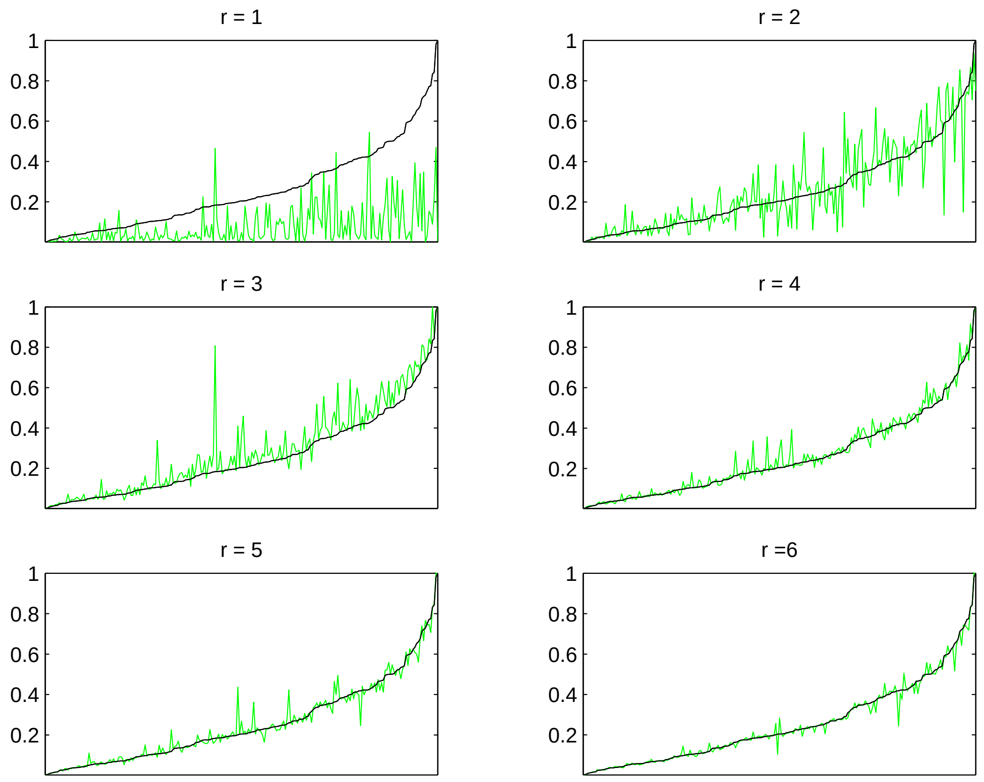

FIG. 3.1. *Average precision, for all 225 queries using the Cranfield set for* $r = 1, 2 \dots 6$ *in the* BIDIAG *algorithm. The dark lines are the vector model and the light grey lines are the Krylov subspace model. Queries are sorted after increasing vector model apr.*

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 (3.9) (3.10) again is in the size of multiplying a sparse matrix with a dense vector.

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) [11]. 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

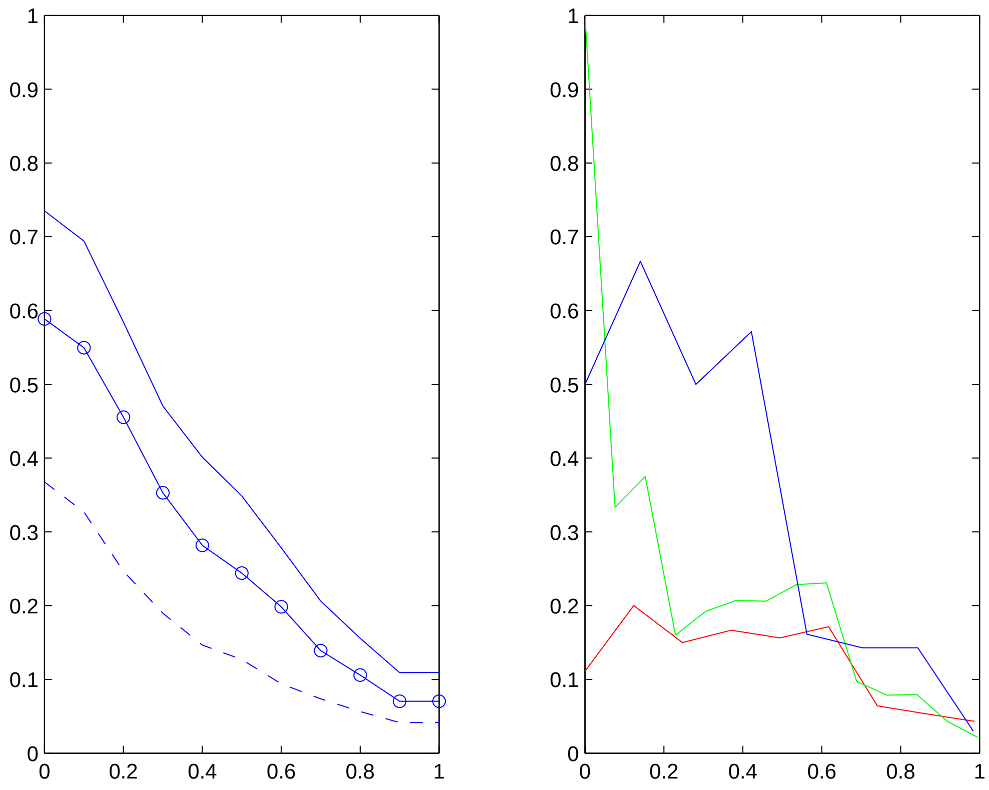

FIG. 4.1. *Precision as a function of recall for the Cranfield collection. Left: Interpolated and averaged over all queries (recall level precision average). Dashed (- -) is vector model, line with circle (-o) is LSI for rank s = 296, line (-) is our Krylov algorithm for r=3 steps. Right: Our Krylov algorithm to r=3 for 3 different queries, precision at actual recall levels.*

and Congressional Record collections. See reports in [4, 5]!

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

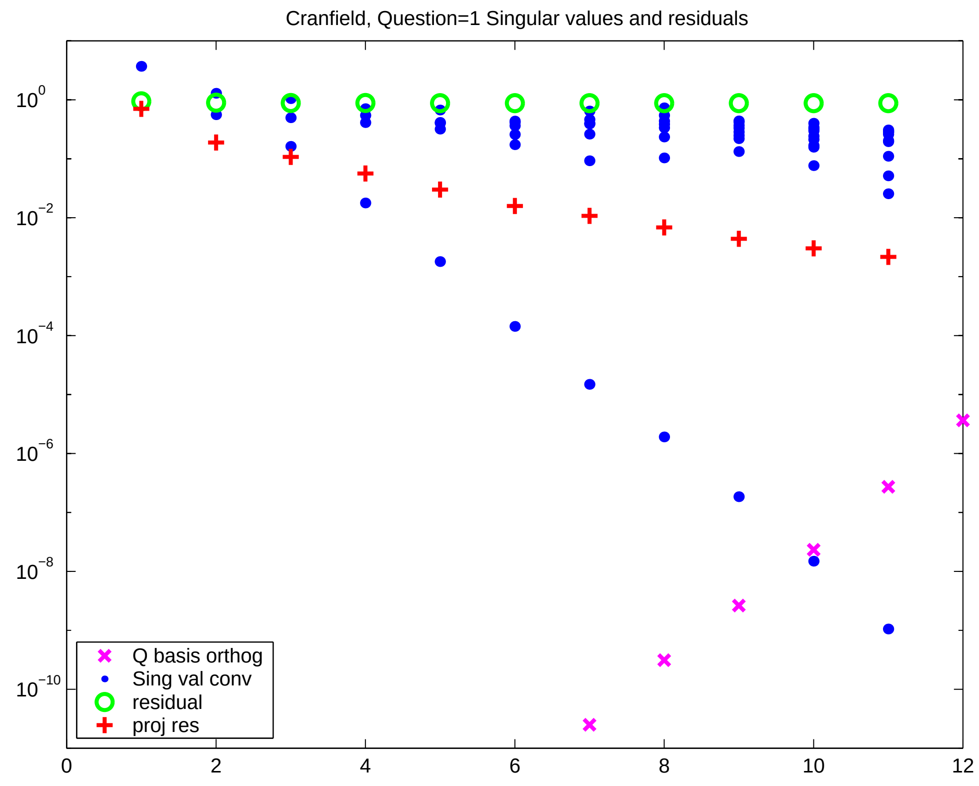

FIG. 4.2. *Cranfield matrix, follow convergence of bidiagonalization procedure starting at query* q_1.

compare our results to these correct answers.

We first summarize the performance in an averaged precision-recall graph. In figure 4.1 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 (1.2) 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 4.2 we follow the progress in linear algebra terms, as we execute the algorithm for steps $k = 1, \ldots, 12$. Circles are the residual norms $\|r^{(k)}\|$, (3.13), 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 (3.4), it has a projection of length 0.477. We plot the normal equation residuals $\|A^T r^{(k)}\|$, (3.15), 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

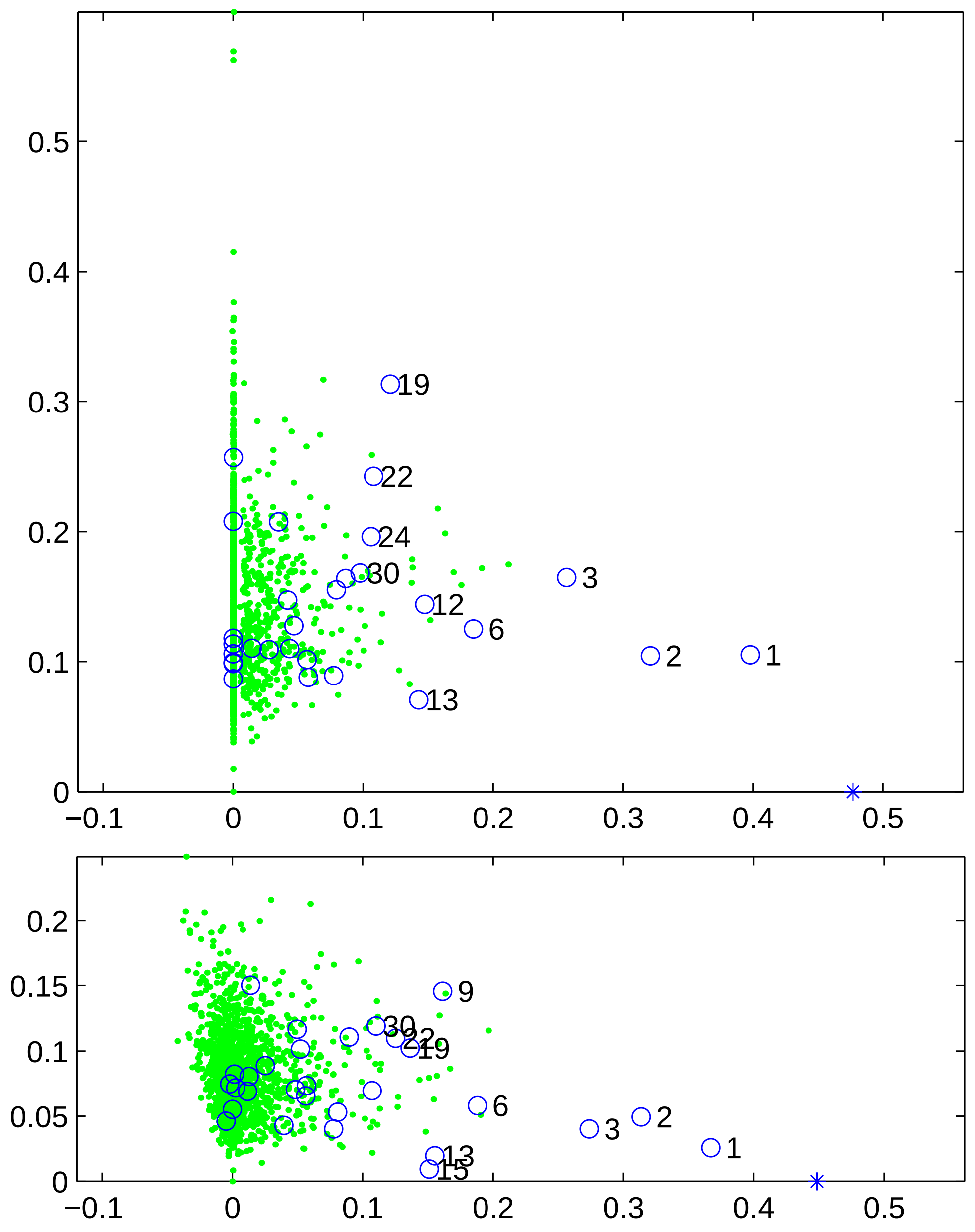

FIG. 4.3. *Cranfield matrix, Query 1, upper half step $r = 12$, lower half step $r = 2$. Numbers are rankings given by the algorithm to relevant documents. Circles mark relevant documents while points mark those not relevant. Asterix marks the projected query.*

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 4.3. The x axis is along the projected query $\hat{q}$ (3.7). The y axis is used to plot the component of each a_j in the reached subspace (3.8) 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 (3.10), taking angles between the original documents and the projected query, we would choose documents from right to left as plotted in figure 4.3, 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 4.3 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 average precision after 10 documents is 0.5. The average precision over all relevant documents [11], 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 4.3, 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 (3.9), these can be seen as angles to the x axis in figure 4.3. It did not differ much from the standard query expansion scoring (3.10), 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 4.3, 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 4.4 the vector model is compared to our algorithm run to $r = 3$. The experiments were made in the same way as for figure 4.1, but we did not have results for LSI for this large matrix. Documents were scored using the standard query expansion scores (3.10). 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 4.5. As for figure 4.3,

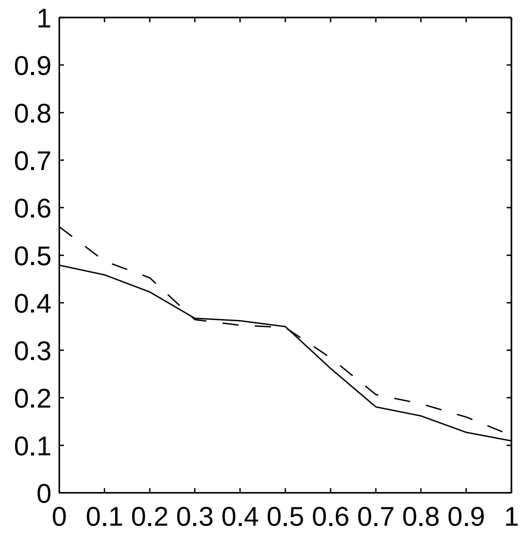

FIG. 4.4. *Interpolated precisions for recall levels* $0, 0.1, \ldots 1$ *for the Financial Times collection from the TREC data base. The vector model (–) is compared to our algorithm for* $r = 3$ *(- -) . The average of the 25 documents that are best ranked by the vector space method are included.*

the x axis is along the projected query $\hat{q}$ (3.7) 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 4.4.

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 experiments, 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

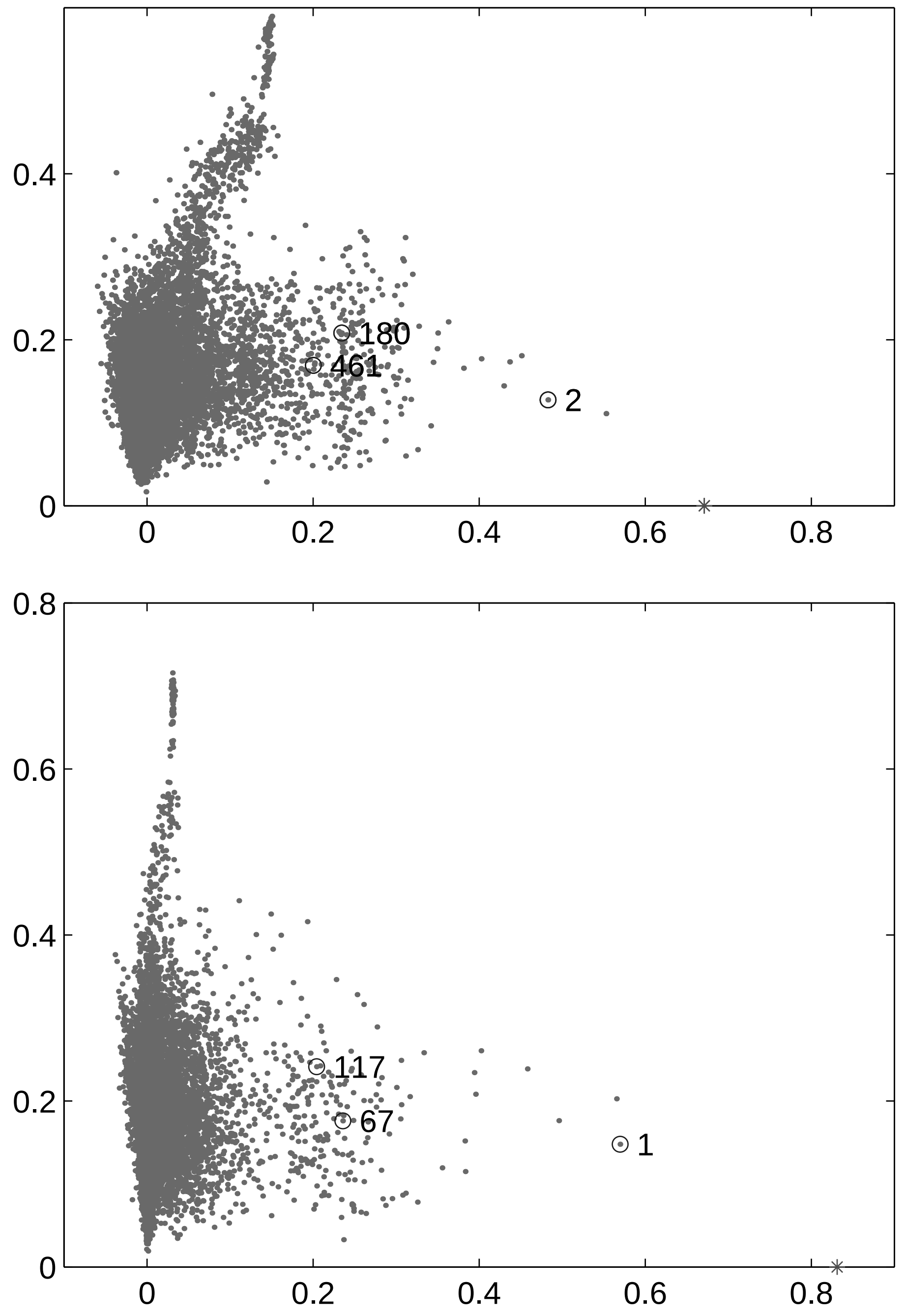

FIG. 4.5. *The TREC Financial Times matrix. Query no. 344, upper half step r = 3 and lower half step r = 6, numbers rankings of relevant documents. For the upper half 97 % of the documents are in the interval* < 0.1 *and for the lower half 99 % of the documents are in that interval, only a sample of those are shown. Asterix marks projected query*

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] D. M. Blei, A. Y. Ng, and M. I. Jordan, *Latent Dirichlet allocation*, J. Mach. Learn. Res., 3 (2003), pp. 993–1022.

[3] 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.

[4] K. Blom, *Information Retrieval using Krylov subspace methods*, PhD thesis, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, 2004. ISBN 91-7291-453-X.

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

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

[12] T. Hofmann, *Probabilistic latent semantic indexing*, in Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in Information Retrieval, ACM Press, 1999, pp. 50–57.

[13] H. Park, M. Jeon, and J. B. Rosen, *Lower diimensional 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.

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

[15] 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.

[16] 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.