FAST UPDATING ALGORITHMS FOR LATENT SEMANTIC INDEXING[*]

EUGENE VECHARYNSKI[†] AND YOUSEF SAAD[‡]

Abstract. This paper discusses a few algorithms for updating the approximate Singular Value Decomposition (SVD) in the context of information retrieval by Latent Semantic Indexing (LSI) methods. A unifying framework is considered which is based on Rayleigh-Ritz projection methods. First, a Rayleigh-Ritz approach for the SVD is discussed and it is then used to interpret the Zha-Simon algorithms [SIAM J. Scient. Comput. vol. 21 (1999), pp. 782-791]. This viewpoint leads to a few alternatives whose goal is to reduce computational cost and storage requirement by projection techniques that utilize subspaces of much smaller dimension. Numerical experiments show that the proposed algorithms yield accuracies comparable or better than those obtained from standard ones at a much lower computational cost.

Key words. Latent Semantic Indexing, text mining, updating algorithm, singular value decomposition, Rayleigh Ritz procedure, Ritz singular values, Ritz singular vectors, min-max characterization, low-rank approximation

AMS subject classifications. 15A18, 65F15, 65F30

1. Introduction.

Latent Semantic Indexing (LSI), introduced in [9], is a well-established text mining technique that aims at finding documents in a given collection that are relevant to a user's query. The method is a variation of the Principal Component Analysis (PCA) [5], where the multidimensional textdata set is projected to a low-dimensional subspace. When properly defined, this subspace captures the essence of the original data. In the projected space, *semantically* similar documents tend to be close to each other in a certain measure, which allows to compare them according to their *latent semantics* rather than a straightforward word matching.

LSI can be viewed as an extension of the *vector space* model for Information Retrieval (IR) [23]. As such, it begins with a preprocessing phase (see, e.g, [1, 26]) to summarize the whole text collection in the m-by-n *term-document* matrix A, where m and n are the total numbers of terms and documents in the collection, respectively. Thus, each column of A represents a separate document, where nonzero entries are the weights, or essentially the frequencies, of the terms occuring in this document. For the discussion of the available term weighting schemes we refer the reader to [17].

We consider a widely used and standard implementation of LSI that is based on the partial Singular Value Decomposition (SVD) [12] of the term-document matrix. In this case, LSI resorts to calculating the singular triplets (σ_j, u_j, v_j) associated with the k largest singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_k \geq 0$ of A. Throughout, we call these k triplets the *dominant* singular triplets. The *left* singular vectors u_j are then used to construct the low-dimensional subspace for data projection and, along with σ_j and the *right* singular vectors v_j, to evaluate the relevance scores. We note that a number of "SVD avoiding" LSI algorithms have been proposed in recent years, e.g., [6, 8, 10, 16, 17], for example, by replacing the SVD by the Lanczos decomposition. We will briefly discuss one such alternative based on using Lanczos vectors.

Given a user's query q, formalized by a vector of size m, i.e., regarded as a document, the associated vector s of n relevance scores is evaluated by

$$r = \operatorname{diag}(\gamma_1, \ldots, \gamma_n) \left(V_k \Sigma_k^{1-\alpha}\right) \Sigma_k^{\alpha} U_k^T q. \tag{1.1}$$

Here, $\Sigma_k = \operatorname{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_k\} \in \mathbb{R}^{k \times k}$; $U_k = [u_1, \ldots, u_k] \in \mathbb{R}^{m \times k}$ and $V_k = [v_1, \ldots, v_k] \in \mathbb{R}^{n \times k}$ are the matrices of the orthonormal left and right singular vectors, respectively. The diagonal elements γ_j are chosen to normalize the rows of $V_k \Sigma_k^{1-\alpha}$ so that each row has a unit norm. The scalar α is a splitting parameter and has no affect on ranking if the normalization is disabled ($\gamma_j = 1$); see, e.g., [17] for a detailed discussion.

The j-th entry of r, denoted by $r(j)$, quantifies the relevance between the j-th document and the query. The documents with the highest relevance scores are returned to the user in response to the query q. Note that, for example, in the case where $\alpha = 0$, $r(j)$ is the (scaled) cosine of the angle between the j-th projected document $U_k^T a_j$ and the projected query $U_k^T q$.

[*]This work was supported by NSF, under grant CCF-1318597, and by the Minnesota Supercomputing Institute.

[†]Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (`eugene.vecharynski@gmail.com`).

[‡] Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455, USA (`saad@cs.umn.edu`).

In practical applications, where the amount of data tends to be extremely large, the implementation of LSI faces two major difficulties. The first difficulty is the requirement to compute the dominant singular triplets of a very large matrix, a problem that has been relatively well investigated. Possible solutions include invoking iterative singular value solvers, e.g., [4, 14], that take advantage of sparsity and fast matrix-vector products. Other solutions leverage specific spectral properties of the term-document matrices, and rely on incremental or divide-and-conquer techniques; e.g., [27, 7].

The second computational difficulty of LSI is related to the fact that document collections are dynamic, i.e., the term-document matrices are subject to repeated updates. Such updates result from adding, e.g., new documents or terms to the collection. In the language of the vector space model, this translates into adding new columns or rows to A. Another type of update is when the term weights are corrected, which corresponds to modifying entries of the term-document matrix. Thus, in order to maintain the quality of the subsequent query processing, the available singular triplets should be accordingly modified after each update. A straightforward solution to this problem is to recompute the partial SVD of the updated term-document matrix from scratch. However, even with the most sophisticated singular value solvers, this naive approach is not practical as it is exceedingly costly for realistic large-scale text collections. Therefore, a critical question is how to *update* the available Σ_k, U_k, and V_k without fully recomputing the high-cost partial SVD of the modified matrix, so that the retrieval quality is not affected.

This paper addresses this specific question. It starts by revisiting the well-known updating algorithms of Zha and Simon [27], currently the state-of-the-art approach for the LSI updating problem. Specifically, the paper interprets these schemes as Rayleigh-Ritz projection procedures. A by-product of this viewpoint is a min-max type characterization of the Ritz singular values obtained in the process. On the practical side, this projection viewpoint unravels a certain redundancy in the computations, showing that it is possible to further improve the efficiency of the techniques without sacrificing retrieval quality. Based on these findings, we propose a family of new updating algorithms which can be substantially faster and less storage-demanding than the methods in [27].

The motivation for the present work comes from the observation that the methods in [27] (reviewed in more detail in Section 3) rely on the SVD of a $(k+p)$-by-$(k+p)$ dense matrix and orthogonalization of p (m- or n-dimensional) vectors, where p denotes the size of the update, i.e., the number of added columns, rows, or corrected terms. In particular, this suggests that the computational complexity of the overall updating procedure scales cubically with respect to p.

While the effect of the cubic scaling is marginal for smaller document collections, where the update sizes are typically given by only a few terms or documents, the situation becomes different for large-scale datasets. In this case, even if p is a tiny fraction of terms or documents, its (cubed) value may be large enough to noticeably affect the efficiency of the updating methods. In other words, for p sufficiently large, the computational costs associated with the SVD of a $(k+p)$-by-$(k+p)$ matrix and orthogonalization (QR decomposition) of p vectors become non-negligible and may dominate the overall updating procedure.

Another context in which larger updates are to be processed can be found in the recent works [25, 18], where the authors suggest to postpone invoking the updating schemes from [27] until the update size becomes sufficiently large. In between the updates, a fast *folding-in* procedure [3, 2] is performed. Such a combination of folding-in and updates, called *folding-up*, has been shown to yield a substantial reduction in the time spent for updating without a significant loss in the retrieval accuracy.

To adapt their algorithms to the cases of large p, the authors of [27] suggest splitting the current update into a series of smaller sequential updates, and performing the whole updating procedure in an incremental fashion. While this approach indeed leads to memory savings, it requires more time to complete the *overall* updating task than to perform the whole update at once. This can be seen, e.g., from Table 4.2 in the original paper [27], after multiplying the reported average CPU times per update by the number of updates. A similar observation has been made in [25, Table 1]. Additionally, as has also been observed in [25], breaking a given update into a sequence of smaller updates can potentially lead to a faster deterioration of the retrieval accuracy.

The updating algorithms proposed in this paper require orthonormalizing sets of significantly fewer vectors than those in [27] and rely on the SVD of much smaller matrices. As a result, the new schemes are less sensitive to the increase in the update sizes. As shown in our experiments, the presented algorithms significantly reduce the runtime, whereas the retrieval accuracy is not affected and is even higher for some examples.

Finally, let us recall that the methods in [27] were introduced as a solution to the problem of the deteriorating retrieval accuracy exhibited by existing methods [3, 19] in the mid-1990s. This solution essentially traded the SVD of a k-by-k matrix in [3, 19] for the above mentioned orthonormalization of a set of p extra vectors plus a $(k+p)$-by-$(k+p)$ SVD. The updating schemes introduced in this work can be viewed as a compromise between [3, 19] and [27], where the runtime resembles that of the former while the retrieval accuracy is comparable to the latter.

The rest of the paper is organized as follows. Projection methods for the SVD are reviewed in Section 2 followed by a discussion of their applications to LSI in Section 3. In Section 4 a number of alternative algorithms are presented with a goal of reducing cost. Section 5 presents numerical experiments to test the various methods introduced, and Section 6 concludes the paper with a few remarks.

2. Projection methods for singular value problems. It will be useful to explore projection methods for the singular value problem in order to understand the mechanisms at play when updating the SVD. We begin with a little background on standard projection methods. Recall that given a Hermitian $n \times n$ matrix M, a Rayleigh-Ritz (RR) projection method extracts approximate eigenpairs for A from a *search subspace* $\texttt{span}\{Z\}$, where $Z \in \mathbb{R}^{n\times s}$. It does so by imposing two conditions. First, any approximate eigenvector must belong to $\texttt{span}\{Z\}$, i.e., it can be written as $z = Zc$ (where $c \in \mathbb{R}^s$). Second, the approximate eigenpair (θ, z) must satisfy the Galerkin condition that the residual is orthogonal to $\texttt{span}\{Z\}$, i.e., we must have $(M - \theta I)z \perp \texttt{span}\{Z\}$. This yields the projected $s \times s$ eigenvalue problem $(Z^T M Z)c = \theta c$ from which we obtain the Ritz values θ_i, the corresponding eigenvectors c_i, and the Ritz vectors $z_i = Zc_i, i = 1, \ldots, s$. Details can be found in, e.g., [20, 21].

2.1. Application to the SVD. Consider now the singular value problem for a matrix $A \in \mathbb{R}^{m\times n}$. The above projection procedure can be adapted to the singular value problem in a number of ways. For example, we can apply the RR idea to one of the two standard eigenvalue problems with either A^TA or AA^T. This, however, is not appealing due to its "non-symmetric" nature: it puts an emphasis on one set of singular vectors (left or right) and will encounter difficulties with the smallest singular values due to their squaring. An altnernative approach is to apply the Rayleigh-Ritz procedure to the augmented matrix

$$B = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}. \tag{2.1}$$

We will consider this second approach as it leads more naturally to a separation of the right and left singular vectors. A straightforward application of the RR procedure would now use a subspace $\texttt{span}\{Z\}$ where $Z \in \mathbb{R}^{(m+n)\times s}$ and would write a test eigenvector in the form $z = Zc$ where $c \in \mathbb{R}^s$. It then imposes the Galerkin condition $Z^T(B - \theta I)Zc = 0$ from which Ritz values and vectors are obtained. Observe that z can be written as $z = \binom{u}{v}$, where $u \in \mathbb{R}^m$ approximates a left singular vector of A and $v \in \mathbb{R}^n$ approximates a right singular vector of A.

One weakness of this viewpoint is that there is no reason why the left approximate singular vector (vector u, i.e, top part of $z = Zc$) and the right approximate singular vector (vector v or bottom part of $z = Zc$) should be expressed in the basis Z with the same basis coefficients c. In practice, we have two bases available, one for the left singular vectors and one for the right singular vectors. Therefore, let $U \in \mathbb{R}^{m\times s_1}$ be a basis for the *left search subspace* and $V \in \mathbb{R}^{n\times s_2}$ a basis for the *right search subspace*, with $s_1 + s_2 = s$. Both U and V are assumed to be orthonormal bases and note that s_1 and s_2 need not be the same. Then, the approximate right singular vector can be expressed as $u = Uf$ (with $f \in \mathbb{R}^{s_1}$) and the approximate left singular vector as $v = Vg$ (with $g \in \mathbb{R}^{s_2}$). This gives us $s_1 + s_2 = s$ degrees of freedom. To extract f and g we would need s constraints which are to be imposed on the residual vector $(B - \theta I)z$ where $z = \binom{u}{v}$. This residual is

$$r = \begin{pmatrix} Av - \theta u \\ A^T u - \theta v \end{pmatrix}. \tag{2.2}$$

It is natural to impose the condition that the first part, which is in $\mathbb{R}^m$, be orthogonal to $\texttt{span}\{U\}$ and the second part, which is in $\mathbb{R}^n$, be orthogonal to $\texttt{span}\{V\}$:

$$\left\{ \begin{array}{lcl} U^T(AVg - \theta Uf) & = & 0 \\ V^T(A^TUf - \theta Vg) & = & 0 \end{array} \right. . \tag{2.3}$$

System (2.3) leads to the projected singular value problem $Hg = \theta f$ and $H^T f = \theta g$, where $H = U^T AV$. Let (θ_i, f_i, g_i) be the singular triplets of the projected matrix H, i.e., $Hg_i = \theta_i f_i$ and $H^T f_i = \theta_i g_i$; $i = 1, \ldots, \min\{s_1, s_2\}$. Then the scalars θ_i are the *Ritz singular values*, the vectors Uf_i are the *left Ritz singular vectors* of A, and the vectors Vg_i are the *right Ritz singular vectors* of A. It is important to note that when $\theta_i > 0$ then the above conditions imply that $\|f_i\| = \|g_i\|$. This can be easily seen by multiplying (from left) both sides of the equality $Hg_i = \theta_i f_i$ by f_i^T and of the equality $H^T f_i = \theta_i g_i$ by g_i^T.

The "doubled" form of the Galerkin condition (2.3) has been considered in [15] in the context of the correction equation for a Jacobi-Davidson approach. These are not quite standard Galerkin conditions since they amount to two separate orthogonality constraints. However, it is also possible to interpret this approach as a standard Galerkin/RR procedure, which is the viewpoint we develop next.

Consider the following new basis for a subspace of $\mathbb{R}^{m+n}$ given by

$$Z = \begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix}. \tag{2.4}$$

Then a test vector z in $\mathtt{span}\{Z\}$ can be written as

$$z = \begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix} \begin{pmatrix} f \\ g \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix}. \tag{2.5}$$

Clearly, the residual vector $(B - \theta I)z$ of this test vector for the approximate eigenvalue θ and the matrix B is again given by (2.2) and the standard Galerkin condition $Z^T r = 0$ yields exactly the doubled form (2.3) of the Galerkin condition.

PROPOSITION 2.1. *The RR procedure defined by the doubled form of the Galerkin condition* (2.3) *is mathematically equivalent to the standard RR procedure applied to the symmetric matrix B using the basis given by* (2.4).

In essence the procedure defined in this way puts an emphasis on *not mixing* the U-space and the V-space as indicated by zeros in the appropriate locations in (2.4) and this is achieved by a restricting the choice of basis for the search space. This is in contrast to the general procedure described at the very beginning of this section, where Z makes no distinction between the U- and V- spaces.

2.2. Application to LSI. Let A be a certain term-document matrix considered at some stage in the updating and querying process and let $U \in \mathbb{R}^{m \times s_1}$ and $V \in \mathbb{R}^{n \times s_2}$ be the orthonormal bases of the left and right search subspaces, respectively. Suppose that our goal is to construct approximations $(\tilde{\sigma}_i, \tilde{u}_i, \tilde{v}_i)$ to the k dominant singular triplets (σ_i, u_i, v_i), such that $\tilde{\sigma}_i = \theta_i \geq 0$, $\tilde{u}_i = Uf_i$, and $\tilde{v}_i = Vg_i$; $i = 1, \ldots k$. The matrices of the singular values and vectors of A are then approximated by

$$\tilde{\Sigma}_k = \Theta_k, \quad \tilde{U}_k = UF_k, \quad \tilde{V}_k = VG_k\ ; \tag{2.6}$$

where $\Theta_k = \mathrm{diag}\,\{\theta_1, \ldots, \theta_k\}$, and $F_k = [f_1, \ldots, f_k] \in \mathbb{R}^{s_1 \times k}$ and $G_k = [g_1, \ldots, g_k] \in \mathbb{R}^{s_2 \times k}$ are the "coefficient matrices" with orthonormal columns. The resulting $\tilde{\Sigma}_k$, $\tilde{U}_k = [\tilde{u}_1, \ldots, \tilde{u}_k]$, and $\tilde{V}_k = [\tilde{v}_1, \ldots, \tilde{v}_k]$ can be used to evaluate the relevance scores in (1.1) instead of the exact Σ_k, U_k, and V_k.

A solution to this problem can be obtained by simultaneously imposing the *Galerkin* conditions seen in Section 2.1 to the k residuals, which along with the assumption on $(\tilde{\sigma}_i, \tilde{u}_i, \tilde{v}_i)$ lead to the equations

$$\left\{ \begin{array}{lcl} (U^T AV)g_i & = & \theta_i f_i \\ (U^T A\bar{V})^T f_i & = & \theta_i g_i \end{array} \right. , \qquad i = 1, \ldots, k\ . \tag{2.7}$$

The unknown triplets (θ_i, f_i, g_i) are determined by the SVD of the projected matrix $H = U^T AV \in \mathbb{R}^{s_1 \times s_2}$. The approximations to the k dominant singular triplets of A are then defined by (2.6) where Θ_k, F_k, and G_k correspond to the k dominant singular triplets of H. The diagonal entries of Θ_k are the Ritz singular values and the columns of $\tilde{U}_k$ and $\tilde{V}_k$ in (2.6) are the left and right Ritz singular vectors, respectively.

We will refer to the above approximation scheme as the *singular value Rayleigh-Ritz* procedure for the matrix A with respect to U and V or, shortly, SV-RR(A, U, V). The overall scheme is summarized in Algorithm 2.1.

ALGORITHM 2.1 (SV-RR (A, U, V)). Input: A, U, V. Output: $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. Construct $H = U^T AV$.

2. Compute the SVD of H. Form matrices Θ_k, F_k, and G_k that correspond to the k dominant singular triplets of H.
3. Return $\tilde{\Sigma}_k$, $\tilde{U}_k$, and $\tilde{V}_k$, given by *(2.6)*.

2.3. Optimality. Since the process just described is a standard RR procedure applied to the matrix (2.1), well-established optimality results for the RR procedure apply. Here, we show a few consequences of this viewpoint, some of which coincide with results that can be found elsewhere, see, e.g., [15], using a different approach. The presented Min-Max results, however, are new to the best of our knowledge.

Let us first examine the Rayleigh Quotient (RQ) associated with the augmented matrix B for a vector $z = \binom{u}{v}$. We find that

$$\rho(z) \equiv \frac{(Bz, z)}{(z, z)} = \frac{2u^T Av}{\|u\|^2 + \|v\|_2^2}. \tag{2.8}$$

As a next step we study properties of this function at points that yield its maximum values in subspaces of $\mathtt{span}(Z)$, where Z is defined in (2.4). The results will be used shortly to establish an optimality result for the SV-RR procedure.

We start with by observing that any subspace $\mathcal{S}$ of $\mathtt{span}(Z)$ has a special structure:

$$\mathcal{S} = \left\{ \begin{pmatrix} u \\ v \end{pmatrix} \in \mathbb{R}^{m+n} : u \in \mathcal{S}_u \text{ and } v \in \mathcal{S}_v \right\}, \quad \dim(\mathcal{S}) = \dim(\mathcal{S}_u) + \dim(\mathcal{S}_v), \tag{2.9}$$

where $\mathcal{S}_u$ and $\mathcal{S}_v$ are some subspaces of $\mathcal{U} = \mathtt{span}(U)$ and $\mathcal{V} = \mathtt{span}(V)$, respectively. In particular, representation (2.9) suggests that if $z = \binom{u}{v}$ is in $\mathcal{S}$ then, for any scalars α and β, the vector $\bar{z} = \begin{pmatrix} \alpha u \\ \beta v \end{pmatrix}$ is also in $\mathcal{S}$. The following lemma is a simple consequence of this property.

Lemma 2.2. *The maximum of the RQ in* (2.8) *over a subspace* $\mathcal{S} \subseteq \operatorname{span}(Z)$ *is nonnegative.*

Proof. Let $z^* = (u_*^T \; v_*^T)^T$ be a maximizer of (2.8) in $\mathtt{span}(Z)$, and assume that the maximum is negative, i.e., $\rho(z_*) < 0$. The vector $\bar{z} = \left(-u_*^T \; v_*^T\right)^T$ is also in $\mathcal{S}$, and we have $\rho(\bar{z}) = -\rho(z_*) > 0 > \rho(z_*)$, contradicting our assumption that $\rho(z_*)$ is the maximum. Therefore, $\rho(z_*) \geq 0$. ◻

As seen earlier, if $\theta_i > 0$ then the conditions $Hg = \theta f$ and $H^T f = \theta g$, arising in the projection procedure, imply that $\|g\| = \|f\|$, so that the approximate singular vectors satisfy $\|u\| = \|v\|$. Viewed from a different angle, we can now ask the question: Does the maximizer of the RQ (2.8) over $z \in \mathtt{span}\{Z\}$, satisfy the property that $\|u\| = \|v\|$? The answer to the question is yes as the following lemma shows.

Lemma 2.3. *Let the maximum of the RQ* (2.8) *over a subspace* $\mathcal{S} \subseteq \operatorname{span}(Z)$ *be achieved at a vector* $z_* = \begin{pmatrix} u_* \\ v_* \end{pmatrix}$. *If the maximum is positive then* $\|u_*\| = \|v_*\|$.

Proof. Let us assume that the contrary is true, i.e., that $\|u_*\| \neq \|v_*\|$. Since $\rho(z_*) > 0$, both u_* and v_* are nonzero. Thus, we can define $\alpha = \sqrt{\|v_*\|/\|u_*\|}$ and introduce the vector $\bar{z} = \left(\alpha u_*^T \; (1/\alpha) v_*^T\right)^T$, which is also in $\mathcal{S}$. But then $\rho(\bar{z}) = (2u_*^T A v_*)/(2\|u_*\|\|v_*\|)$ which is larger than $\rho(z_*)$ under the assumption that $\|u_*\| \neq \|v_*\|$. Hence we have increased the value of $\rho(z_*)$ contradicting the fact that it is the maximum. Therefore we must have $\|u_*\| = \|v_*\|$. ◻

While the above result concerns the case where the maximum of the RQ is positive, the next lemma shows that if a subspace $\mathcal{S}$ is sufficiently large then one can also find vectors $z \in \mathcal{S}$ with $\|u\| = \|v\|$ that correspond to a zero maximum.

Lemma 2.4. *Let the maximum of the RQ* (2.8) *over* $\mathcal{S} \subseteq \operatorname{span}(Z)$ *be zero and assume that* $\dim(\mathcal{S}) > \max(s_1, s_2)$, *where* $s_1 = \dim(\mathcal{U})$ *and* $s_2 = \dim(\mathcal{V})$. *Then there exists a maximizer* $z^* \in \mathcal{S}$, *such that* $\|u^*\| = \|v^*\|$.

Proof. First we show the existence of a maximizer z_0 of (2.8) with nonzero components u_0 and v_0. The condition $\dim(\mathcal{S}) \equiv \dim(\mathcal{S}_u) + \dim(\mathcal{S}_v) > \max(s_1, s_2)$, with $\dim(\mathcal{S}_u) \leq s_1$ and $\dim(\mathcal{S}_v) \leq s_2$, implies that neither of the subspaces $\mathcal{S}_u, \mathcal{S}_v$ has zero dimension, so $\mathcal{S}$ contains a vector $z_0 = (u_0^T, v_0^T)^T$ such that $u_0, v_0 \neq 0$. Without loss of generality, we assume that $\rho(z_0) \geq 0$ (otherwise, choose $z_0 = (-u_0^T, v_0^T)^T$ which is also in $\mathcal{S}$). But we cannot have $\rho(z_0) > 0$ since the maximum of ρ is zero. Therefore $\rho(z_0) = 0$ and z_0 is the desired vector. To complete the proof, define $z_* \equiv (u_*^T, v_*^T)^T$ with $u_* = u_0/\|u_0\|$, $v_* = v_0/\|v_0\|$. We clearly still have $\rho(z_*) = 0$ and the vector z_*, which belongs to $\mathcal{S}$, has the desired property. ◻

The lemmas show that under the specified conditions, the vectors u and v of the RQ maximizers are (or can be chosen to be) of the same norm which can be set to one without loss of generality. In particular, since

θ_1 is the maximum of the RQ over the space of all nonzero vectors z of the form (2.5), we readily obtain the following characterization of the largest Ritz singular value, which can also be found in [15]:

$$\theta_1 = \max_{\substack{u\in\mathcal{U};v\in\mathcal{V};\\ \|u\|=\|v\|=1}} u^T A v \geq 0. \tag{2.10}$$

We now wish to generalize this result by establishing a min-max type characterization for all Ritz singular values. For an $n \times n$ Hermitian matrix M and a search subspace $\mathcal{Z}$ of dimension s the decreasingly labeled Ritz values θ_i are given by

$$\theta_i = \min_{\substack{\mathcal{S}\subseteq\mathcal{Z}\\ \dim(\mathcal{S})=s-i+1}} \max_{x\,\in\,\mathcal{S},x\neq 0} \frac{(Mx,x)}{(x,x)}, \quad i=1,\ldots,s; \tag{2.11}$$

see, e.g., [20, 21]. This expression, combined with the results of the current section, applied to our situation, where $M = B$ is the augmented matrix in (2.1) and $\mathcal{Z} = \texttt{span}\{Z\}$ where Z is defined in (2.4), results in the following Min-Max characterization of the Ritz singular values.

Theorem 2.5. *The Ritz singular values of a matrix $A \in \mathbb{R}^{m\times n}$ with respect to the left and right search subspaces $\mathcal{U} = \text{span}\{U\} \subseteq \mathbb{R}^m$ and $\mathcal{V} = \text{span}\{V\} \subseteq \mathbb{R}^n$, with dim$(\mathcal{U}) = s_1$, $\dim(\mathcal{V}) = s_2$, and $s_1 + s_2 = s$, admit the following characterization:*

$$\theta_i = \min_{\substack{\mathcal{S}_u\subseteq\mathcal{U},\mathcal{S}_v\subseteq\mathcal{V},\\ dim(\mathcal{S}_u)+dim(\mathcal{S}_v)=s-i+1}} \max_{\substack{u\in\mathcal{S}_u;\ v\in\mathcal{S}_v;\\ \|u\|=\|v\|=1}} u^T A v, \qquad i=1,\ldots,\min(s_1,s_2). \tag{2.12}$$

Proof. We start from (2.11). Let $\mathcal{Z}$ be spanned by a basis of the form (2.4). A candidate subspace $\mathcal{S}$ of $\mathcal{Z}$ of this type is of the form (2.9). The dimension of this subspace $\mathcal{S}$ must be $s-i+1$, which translates to the requirement that dim $(\mathcal{S}_u)$ + dim $(\mathcal{S}_v) = s - i + 1$. Next, we replace M by B in (2.11). Then $(Mx,x)/(x,x)$ yields the expression in (2.8). Lemma 2.3 shows that a positive maximum of this RQ in (2.11) is reached at vectors with $\|u\| = \|v\|$, so we can scale both u and v to have unit norm. Note that the assumption $i \leq \min(s_1, s_2)$ implies that $\dim(\mathcal{S})$ is larger than $\max(s_1, s_2)$. Therefore, if the maximum is zero then, by Lemma 2.4, there exists a maximizer with $\|u\| = \|v\|$ which can also be scaled so that both u and v have unit norm. This yields (2.12). □

Note that the same ideas applied to the augmented matrix (2.1) with the search subspace $\texttt{span}(Z) = \mathbb{R}^{m+n}$, where U and V in (2.4) are orthonormal bases of $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively, lead to the Min-Max characterization of the singular values of A:

$$\sigma_i = \min_{\substack{\mathcal{S}_u\subseteq\mathbb{R}^m,\mathcal{S}_v\subseteq\mathbb{R}^n,\\ \dim(\mathcal{S}_u)+\dim(\mathcal{S}_v)=m+n-i+1}} \max_{\substack{u\in\mathcal{S}_u;\ v\in\mathcal{S}_v;\\ \|u\|=\|v\|=1}} u^T A v, \qquad i=1,\ldots,\min(m,n). \tag{2.13}$$

One can see that (2.12) is identical to (2.13) with $\mathbb{R}^m$ replaced by $\mathcal{U}$ and $\mathbb{R}^n$ by $\mathcal{V}$. This observation provides an interpretation of the optimality of the SV-RR procedure: it constructs a solution that admits the same characterization of the singular values in the given subspaces.

The following statement is a direct consequence of Theorem 2.5.

Corollary 2.6. *Let σ_i and θ_i be labeled in a decreasing order. Then the Ritz values approximate the singular values from below, i.e., $\theta_i \leq \sigma_i$; $i = 1,\ldots,\min(s_1,s_2)$. Additionally, if we have a sequence of expanding subspaces $\{\mathcal{Z}_j\}$, such that $\mathcal{Z}_j \subseteq \mathcal{Z}_{j+1}$, then $\theta_i^{(j)} \leq \theta_i^{(j+1)} \leq \sigma_i$, where $\theta_i^{(j)}$ is the i-th Ritz singular value with respect to the subspace $\mathcal{Z}_j$.*

The results of this section suggest that the proximity of $\tilde{\Sigma}_k$, $\tilde{U}_k$, and $\tilde{V}_k$, produced by Algorithm 2.1, to the singular triplets of A is governed by the choice of U and V. If both are properly chosen, then SV-RR(A, U, V) can give an appealing approach for the updating problem. In particular, as can be seen from Theorem 2.5, if $\texttt{span}(U)$ and $\texttt{span}(V)$ contain the left and right dominant singular subspaces of A, $\texttt{span}(U_k)$ and $\texttt{span}(V_k)$, then Algorithm 2.1 readily delivers the k *exact* singular triplets of interest. If, additionally, the number of columns in U or V is sufficiently small (not much larger than k) then the computational costs related to the procedure can be negligibly low.

In practice, however, the construction of search subspaces that include the targeted singular subspaces and, at the same time, have small dimensions can be problematic. It is likely that in order to obtain the inclusion of the singular subspaces, both s_1 and s_2 have to be large (see Corollary 2.6), and this can make Algorithm 2.1 too costly to be used as a fast updating scheme. In what follows, we adapt this viewpoint to address possible difficulties with existing methods.

3. LSI updating methods viewed as projection procedures.

We now consider a common situation in IR which arises when a certain term-document matrix A is updated. In the case when a few documents are added, the updated term-document matrix can be written as

$$\tilde{A}_D = [A,\ D], \tag{3.1}$$

where $D \in \mathbb{R}^{m\times p}$, represents the matrix of added documents. Similarly, if terms are added, then the update matrix takes the form

$$\tilde{A}_T = \begin{bmatrix} A \\ T \end{bmatrix}, \tag{3.2}$$

where $T \in \mathbb{R}^{p\times n}$ corresponds to the added terms. It is also possible to correct the weights of the terms, in which case the updated A is given by

$$\tilde{A}_{CW} = A + CW, \tag{3.3}$$

where $C \in \mathbb{R}^{m\times p}$ is a "selection matrix" of unit vectors that define the p corrected terms and $W \in \mathbb{R}^{p\times n}$ contains the corresponding weight corrections.

The goal of the updating algorithms is to compute approximations $\tilde{\Sigma}_k$, $\tilde{U}_k$, and $\tilde{V}_k$ to the dominant singular triplets of the updated matrices in (3.1)–(3.3) by exploiting the knowledge of the singular triplets Σ_k, U_k, and V_k of A. A common approach is based on the idea of replacing A by its best rank-k approximation $A_k = U_k\Sigma_k V_k^T$, and considering

$$A_D = [A_k\ D], \qquad A_T = \begin{bmatrix} A_k \\ T \end{bmatrix}, \qquad \text{and} \qquad A_{CW} = A_k + CW \tag{3.4}$$

as substitutes for the updated matrices in (3.1)–(3.3). The k dominant singular triplets of (3.4) are then regarded as approximations of the "true" updated singular triplets of (3.1)–(3.3), and are used to evaluate the relevance scores in (1.1).

The result of Zha and Simon [27] shows that it is possible to compute the *exact* k dominant singular triplets of the matrices in (3.4) without invoking standard singular value solvers from scratch. The closeness of the computed triplets to those of $\tilde{A}_D$, $\tilde{A}_T$, and $\tilde{A}_{CW}$ in (3.1)–(3.3) is justified by exploiting the so-called approximate "low-rank-plus-shift" structure of A; see also [28] for a more rigorous analysis. Below, we briefly review the updating schemes presented in [27].

3.1. Updating algorithms of Zha and Simon [27].

Consider first the case of adding new documents D. Let

$$(I - U_kU_k^T)D = \hat{U}_pR \tag{3.5}$$

be the truncated QR decomposition of $(I - U_kU_k^T)D$, where $\hat{U}_p \in \mathbb{R}^{m\times p}$ has orthonormal columns and $R \in \mathbb{R}^{p\times p}$ is upper triangular. Given (3.5), one can observe that

$$A_D = [U_k,\ \hat{U}_p]H_D \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix}^T, \ H_D = \begin{bmatrix} \Sigma_k & U_k^TD \\ 0 & R \end{bmatrix}, \tag{3.6}$$

where I_p denotes the p-by-p identitity matrix. Thus, if Θ_k, F_k, and G_k are the matrices corresponding to the k dominant singular values of $H_D \in \mathbb{R}^{(k+p)\times(k+p)}$ and their left and right singular vectors, respectively, then the desired updates $\tilde{\Sigma}_k$, $\tilde{U}_k$, and $\tilde{V}_k$ are given by

$$\tilde{\Sigma}_k = \Theta_k,\ \tilde{U}_k = [U_k,\ \hat{U}_p]F_k,\ \text{and}\ \tilde{V}_k = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix} G_k\ . \tag{3.7}$$

This updating procedure is summarized in the following algorithm.

Algorithm 3.1 (Adding documents (Zha-Simon [27])). Input: Σ_k, U_k, V_k, D. Output: $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. Construct $(I - U_kU_k^T)D$. Compute the QR decomposition (3.5).
2. Construct H_D in *(3.6)*. Compute the matrices Θ_k, F_k, and G_k that correspond to the k dominant singular triplets of H_D.

3. Return $\tilde{\Sigma}_k$, $\tilde{U}_k$, and $\tilde{V}_k$ defined by *(3.7)*.

Similarly, if new terms are added, then the following equality holds:

$$A_T = \begin{bmatrix} U_k^T & 0 \\ 0 & I_p \end{bmatrix} H_T [V_k,\ \hat{V}_p]\ ,\ H_T = \begin{bmatrix} \Sigma_k & 0 \\ TV_k & L \end{bmatrix}\ . \tag{3.8}$$

Here, $\hat{V}_p$ and L^T are the factors in the truncated QR decomposition of $(I - V_k V_k^T)T^T$,

$$(I - V_k V_k^T)T^T = \hat{V}_p L^T\ . \tag{3.9}$$

The updated singular triplets are then defined as

$$\tilde{\Sigma}_k = \Theta_k,\ \tilde{U}_k = \begin{bmatrix} U_k & 0 \\ 0 & I_p \end{bmatrix} F_k,\ \text{and}\ \tilde{V}_k = [V_k,\ \hat{V}_p]G_k, \tag{3.10}$$

where Θ_k, F_k, and G_k now denote the matrices of the k dominant singular triplets of $H_T \in \mathbb{R}^{(k+p)\times(k+p)}$.

Algorithm 3.2 (Adding terms (Zha-Simon [27])). *Input:* Σ_k, U_k, V_k, T. *Output:* $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. Construct $(I - V_k V_k^T)T^T$. Compute the QR decomposition (3.9).
2. Construct H_T in *(3.8)*. Compute the matrices Θ_k, F_k, and G_k that correspond to the k dominant singular triplets of H_T.
3. Return $\tilde{\Sigma}_k$, $\tilde{U}_k$, and $\tilde{V}_k$ defined by *(3.10)*.

Finally, in the case of correcting the term weights,

$$A_{CW} = [U_k,\ \hat{U}_p]H_{CW}[V_k,\ \hat{V}_p]^T\ ,\ H_{CW} = \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_k^T C \\ R \end{bmatrix} [WV_k,\ L]\ , \tag{3.11}$$

where $\hat{U}_p$, R, $\hat{V}_p$, and L are given by the truncated QR decompositions

$$(I - U_k U_k^T)C = \hat{U}_p R, \qquad (I - V_k V_k^T)W^T = \hat{V}_p L^T. \tag{3.12}$$

Thus, assuming that Θ_k, F_k, and G_k are the matrices of the k dominant singular triplets of $H_{CW} \in \mathbb{R}^{(k+p)\times(k+p)}$,

$$\tilde{\Sigma}_k = \Theta_k,\ \tilde{U}_k = [U_k,\ \hat{U}_p]F_k,\ \text{and}\ \tilde{V}_k = [V_k, \hat{V}_p]G_k. \tag{3.13}$$

Algorithm 3.3 (Correcting weights (Zha-Simon [27])). *Input:* Σ_k, U_k, V_k, C, W. *Output:* $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. Construct $(I - U_k U_k^T)C$ and $(I - V_k V_k^T)W^T$. Compute the QR decompositions *(3.12)*.
2. Construct H_{CW} in *(3.11)*. Compute the matrices Θ_k, F_k, and G_k that correspond to the k dominant singular triplets of H_{CW}.
3. Return $\tilde{\Sigma}_k$, $\tilde{U}_k$, and $\tilde{V}_k$ defined by *(3.13)*.

3.2. The Rayleigh-Ritz viewpoint.

It is easy to see that Algorithm 3.1 is equivalent to SV-RR(A_D, U, V) with

$$U = \left[U_k,\ \hat{U}_p\right],\ V = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix}. \tag{3.14}$$

Note that, in this case, the matrix H_D in (3.6) is precisely the projected matrix H in step 1 of Algorithm 2.1.

The fact that Algorithm 2.1 yields the exact dominant singular triplets of A_D comes from the observation that

$$\texttt{range}(A_D) = \texttt{span}(U_k) \oplus \texttt{range}((I - U_k U_k^T)D)\ ,$$

and that the columns of $\hat{U}_p$ in (3.5) form an orthonormal basis of $\texttt{range}((I - U_k U_k^T)D)$. Hence, $\texttt{span}(U) = \texttt{range}(A_D)$, i.e., the search subspace $\texttt{span}(U)$ with U defined in (3.14) must contain the left dominant singular subspace of A_D. The corresponding right singular subspace is then contained in $\texttt{range}(A_D^T U)$. But

$$A_D^T U = \begin{bmatrix} A_k^T U_k & A_k^T \hat{U}_p \\ D^T U_k & D^T \hat{U}_p \end{bmatrix} = \begin{bmatrix} V_k \Sigma_k & 0 \\ D^T U_k & D^T \hat{U}_p \end{bmatrix} = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ D^T U_k & D^T \hat{U}_p \end{bmatrix},$$

implying that $\mathtt{span}(V) = \mathtt{range}(A_D^T U)$. Thus, the search subspace $\mathtt{span}(V)$ defined in (3.14) must contain the right singular subspace of A_D. As a result, since both $\mathtt{span}(U)$ and $\mathtt{span}(V)$ contain the dominant singular subspaces, a run of SV-RR(A_D, U, V) in Algorithm 3.1 is indeed guaranteed to deliver the k exact dominant singular triplets of A_D.

Similarly, for the case of added new terms, Algorithm 3.2 , can be interpreted as SV-RR(A_T, U, V) with

$$U = \begin{bmatrix} U_k & 0 \\ 0 & I_p \end{bmatrix}, \; V = \begin{bmatrix} V_k, \; \hat{V}_p \end{bmatrix} , \tag{3.15}$$

where $\hat{V}_p$ is defined by the QR decomposition (3.9). Algorithm 3.3 is equivalent to SV-RR(A_{CW}, U, V) with

$$U = \begin{bmatrix} U_k, \; \hat{U}_p \end{bmatrix}, \; V = \begin{bmatrix} V_k, \; \hat{V}_p \end{bmatrix} , \tag{3.16}$$

where $\hat{U}_p$ and $\hat{V}_p$ are given by (3.12). The matrices H_T and H_{CW} in (3.8) and (3.11) are the projected matrices that arise at step 1 of Algorithm 2.1 with the appropriate input. The fact that the dominant singular triplets of A_T and A_{CW} are computed *exactly* can be deduced by following the arguments similar to those used above to justify the exactness of Algorithm 3.1.

Finally, note that the updating methods from [3, 19], which preceded the schemes of Zha and Simon [27], can also be easily interpreted in the framework of the projection procedure in Algorithm 2.1. For example, the case of adding new documents in [3, 19] is realized by SV-RR(A_D, U_k, V) with V in (3.14). The updating strategies for the remaining two types of update correspond to SV-RR(A_T, U, V_k) with U in (3.15) and SV-RR(A_{CW}, U_k, V_k).

4. Updating by the SV-RR with smaller subspaces.

Consider the computational complexity of Algorithm 3.1. In its first step, the algorithm performs $\mathcal{O}(mkp)$ operations to form the matrix $(I - U_k U_k^T)D$ and $\mathcal{O}(mp^2)$ operations to complete the QR decomposition (3.5). Assuming that $U_k^T D$ has been precomputed at step 1, the cost of the second step is $\mathcal{O}((k+p)^3)$, which corresponds to the cost of the SVD of the $(k+p)$-by-$(k+p)$ matrix H_D. Finally, in the third step, the algorithm requires $\mathcal{O}(k^2(m+n) + mkp)$ operations to evaluate (3.7). The complexities of all the updating algorithms considered in this paper are summarized in Tables 4.1-4.3 of subsection 4.3.

The above analysis makes it clear that the complexity of Algorithm 3.1 scales cubically with respect to the update size p. When p is small this cubic scaling behavior will have an un-noticeable effect. However, it can lead to a substantial slow down for moderate to large p. In other words, for sufficiently large updates, the performance of Algorithm 3.1 can be dominated by the SVD of the projected matrix in step 2 and, to a lesser extent, by the QR decomposition of $(I - U_k U_k^T)D$ in step 1.

In subsection 3.2, we have established the relation between the existing updating methods [27] and a projection scheme for the singular value problem, which allows us to interpret Algorithm 3.1 as SV-RR(A_D, U, V) with U and V specified in (3.14). In particular, this finding suggests that the size of the potentially critical SVD in step 2 of Algorithm 3.1 is determined by the dimensions of the left and right search subspaces.

In this paper, we propose to reduce the dimension of at least one of the two search subspaces and perform the projection procedure with respect to the resulting smaller subspace(s). For example, in the case of adding new documents, we suggest to reduce the dimension of the *left* search subspace $\mathtt{span}(U)$. Based on different options for the dimension reduction, we devise new updating schemes that correspond to Algorithm 2.1 with the "reduced" left search subspaces, $A \equiv A_D$, and V in (3.14).

More precisely, the idea is to replace $U \in \mathbb{R}^{m \times (k+p)}$ in (3.14) by a matrix

$$\bar{U} = [U_k, \; Z_l] \in \mathbb{R}^{m \times (k+l)}, \qquad Z_l \in \mathbb{R}^{m \times l}, \qquad l \ll p, \tag{4.1}$$

with a significantly smaller number of columns. The matrix $Z_l \in \mathbb{R}^{m \times l}$, to be determined later, is assumed to have orthonormal columns and is such that

$$\mathtt{span}(Z_l) \subset \mathtt{range}\left((I - U_k U_k^T)D\right) . \tag{4.2}$$

This implies that $Z_l^T U_k = \mathbf{0}$ and, hence, all the columns of $\bar{U}$ in (4.1) are orthonormal. Thus, given (4.1) and (4.2), one can obtain an updating scheme by applying SV-RR(A_D, $\bar{U}$, V) with V defined in (3.14). Different choices of Z_l will lead to different updating schemes.

The following proposition states the general form of the projected matrix produced by SV-RR($A_D, \bar{U}, V$).

PROPOSITION 4.1. *The application SV-RR(A_D, $\bar{U}$, V) of Algorithm 2.1 with $\bar{U}$ defined in (4.1)–(4.2) and V in (3.14) produces the $(k+l)\times(k+p)$ projected matrix*

$$H = \begin{bmatrix} \Sigma_k & U_k^T D \\ 0 & Z_l^T (I - U_k U_k^T) D \end{bmatrix} . \tag{4.3}$$

Proof. The statement is verified directly by constructing $H = \bar{U}^T A_D V$. Since $A_k = U_k \Sigma_k V_k^T$,

$$A_D V = [A_k,\ D] \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix} = [U_k \Sigma_k,\ D] \ .$$

Thus, using (4.1),

$$H = \bar{U}^T (A_D V) = [U_k,\ Z_l]^T \, [U_k \Sigma_k,\ D] = \begin{bmatrix} \Sigma_k & U_k^T D \\ Z_l^T U_k \Sigma_k & Z_l^T D \end{bmatrix} .$$

Since, by (4.2), $Z_l^T U_k = \mathbf{0}$, the $(2,1)$-block of H is zero. From (4.2), we also note that $Z_l = (I - U_k U_k^T) Z_l$. Hence, the $(2,2)$-block equals $Z_l^T (I - U_k U_k^T) D$, which completes the proof. □

A similar idea can be applied for the case of adding new terms. In particular, we replace $V \in \mathbb{R}^{n\times(k+p)}$ in (3.15) by an n-by-$(k+l)$ matrix

$$\bar{V} = [V_k,\ Z_l] \in \mathbb{R}^{n\times(k+l)}, \qquad Z_l \in \mathbb{R}^{n\times l}, \qquad l \ll p, \tag{4.4}$$

where Z_l has orthonormal columns and

$$\mathtt{span}(Z_l) \subset \mathtt{range}\left((I - V_k V_k^T) T^T\right) . \tag{4.5}$$

Then, by (4.5), $Z_l^T V_k = \mathbf{0}$ and, therefore, all the columns of $\bar{V}$ in (4.4) are orthonormal. Thus, an updating scheme can be obtained by applying SV-RR(A_T, U, $\bar{V}$). The corresponding projected matrix is given by the following proposition.

PROPOSITION 4.2. *The application of SV-RR(A_T, U, $\bar{V}$) of Algorithm 2.1 with U defined in (3.15) and $\bar{V}$ in (4.4)–(4.5) produces the $(k+p)\times(k+l)$ projected matrix*

$$H = \begin{bmatrix} \Sigma_k & 0 \\ T V_k & T(I - V_k V_k^T) Z_l \end{bmatrix} . \tag{4.6}$$

Finally, if the term weights are corrected, in (3.16), we subtitute U by

$$\bar{U} = [U_k,\ Z_{l_1}^{(1)}], \qquad Z_{l_1}^{(1)} \in \mathbb{R}^{m\times l_1}, \qquad l_1 \ll p, \tag{4.7}$$

and V by

$$\bar{V} = [V_k,\ Z_{l_2}^{(2)}], \qquad Z_{l_2}^{(2)} \in \mathbb{R}^{n\times l_2}, \qquad l_2 \ll p \ . \tag{4.8}$$

Similarly, $Z_{l_1}^{(1)} \in \mathbb{R}^{m\times l_1}$ and $Z_{l_2}^{(2)} \in \mathbb{R}^{n\times l_2}$ are assumed to have orthonormal columns, and are such that

$$\mathtt{span}(Z_{l_1}^{(1)}) \subset \mathtt{range}\left((I - U_k U_k^T) C\right), \qquad \mathtt{span}(Z_{l_2}^{(2)}) \subset \mathtt{range}\left((I - V_k V_k^T) W^T\right) . \tag{4.9}$$

The latter implies that $Z_{l_1}^{(1)T} U_k = \mathbf{0}$ and $Z_{l_2}^{(2)T} V_k = \mathbf{0}$, i.e., both $\bar{U}$ and $\bar{V}$ in (4.7)–(4.8) have orthonormal columns. As a result, an updating scheme can be given by SV-RR(A_{CW}, $\bar{U}$, $\bar{V}$), with $\bar{U}$ and $\bar{V}$ defined in (4.7)–(4.9).

PROPOSITION 4.3. *The SV-RR(A_{CW}, $\bar{U}$, $\bar{V}$) run of Algorithm 2.1 with $\bar{U}$ and $\bar{V}$ defined in (4.7)–(4.9) produces the $(k+l_1)\times(k+l_2)$ projected matrix*

$$H = \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_k^T C \\ Z_{l_1}^{(1)T} (I - U_k U_k^T) C \end{bmatrix} \left[W V_k,\ W(I - V_k V_k^T) Z_{l_2}^{(2)} \right] . \tag{4.10}$$

It is important to emphasize that, in contrast to Algorithms 3.1–3.3, the updating framework introduced above, which uses smaller search subspaces, is no longer expected to deliver the exact singular triplets of A_D, A_T, or A_{CW}. However, as the numerical experiments in section 5 will demonstrate, satisfactory retrieval accuracies can be achieved without computing these triplets exactly, and only approximations suffice in practice. This observation can be related to the fact that the matrices A_D, A_T, and A_{CW} are themselves approximations of the "true" updated matrices $\tilde{A}_D$, $\tilde{A}_T$, and $\tilde{A}_{CW}$ in (3.1)–(3.3), and so there is no need to compute their singular triplets with high accuracy.

The rest of this section studies several choices of Z_l in (4.1)–(4.2) and (4.4)–(4.5), as well as of $Z^{(1)}_{l_1}$ and $Z^{(2)}_{l_2}$ in (4.7)–(4.9).

4.1. Optimal rank-l approximations.. We start with the approach based on SV-RR(A_D, $\bar{U}$, V), where $\bar{U}$ is assumed to satisfy (4.1)–(4.2) and V is defined in (3.14). Our goal is to specify a suitable choice of Z_l and turn the general projection procedure into a practical updating algorithm that handles the case of adding new documents.

As has been pointed out in subsection 3.2, a run of SV-RR(A_D, U, V) with U and V from (3.14) is equivalent to Algorithm 3.1, which is known to compute the exact dominant singular triplets of A_D. This property of the algorithm, paired with the analysis in [28], explains the generally satisfactory retrieval quality maintained by the updating scheme. From these considerations, it is desirable that the new search subspace $\mathtt{span}(\bar{U})$ utilized by SV-RR(A_D, $\bar{U}$, V) approximates $\mathtt{span}(U)$. In this case, our expectation is that SV-RR(A_D, $\bar{U}$, V) will exhibit a retrieval accuracy comparable to that of SV-RR(A_D, U, V) implemented by Algorithm 3.1.

By definition, $\mathtt{span}(\bar{U}) = \mathtt{span}(U_k) \oplus \mathtt{span}(Z_l)$ and $\mathtt{span}(U) = \mathtt{span}(U_k) \oplus \mathtt{range}\left((I - U_kU_k^T)D\right)$. Therefore, we seek to construct Z_l such that $\mathtt{span}(Z_l)$ approximates $\mathtt{range}((I - U_kU_k^T)D)$. In other words, we would like to compress the information contained in the subspace $\mathtt{range}\left((I - U_kU_k^T)D\right)$ into the span of a set of l orthonormal vectors. Formally, this requirement translates into the problem of constructing Z_l, such that

$$\mathtt{span}(Z_l) = \mathtt{range}(M), \tag{4.11}$$

where $M \approx \left(I - U_kU_k^T\right) D \in \mathbb{R}^{m\times p}$ is some matrix with rank$(M) = l$.

This is related to the standard task of constructing a low-rank approximation M of $(I - U_kU_k^T)D$. It is well known, e.g., [12], that an *optimal* rank-l approximation of $(I - U_kU_k^T)D$ is given by $M = X_lS_lY_l^T$, where S_l is a diagonal matrix of l dominant singular values of $(I - U_kU_k^T)D$, and $X_l \in \mathbb{R}^{m\times l}$ and $Y_l \in \mathbb{R}^{p\times l}$ are the matrices of the corresponding left and right singular vectors, i.e.,

$$(I - U_kU_k^T)DY_l = X_lS_l, \qquad D^T(I - U_kU_k^T)^TX_l = Y_lS_l\ . \tag{4.12}$$

Thus, it is natural to choose $Z_l = X_l$, i.e., augment $\bar{U}$ in (4.1) with a few left dominant singular vectors of $(I - U_kU_k^T)D$. It is clear that this choice of Z_l indeed satisfies (4.2). Then, by Proposition 4.1 and (4.12), a run of SV-RR(A_D, $\bar{U}$, V) with $\bar{U} = [U_k,\ X_l]$ and V in (3.14) produces the projected matrix

$$H = \begin{bmatrix} \Sigma_k & U_k^TD \\ 0 & S_lY_l^T \end{bmatrix} \in \mathbb{R}^{(k+l)\times(k+p)}\ . \tag{4.13}$$

This leads to the following updating scheme based on the singular vectors (SV) of $(I - U_kU_k^T)D$.

Algorithm 4.1 (Adding documents (SV)). Input: Σ_k, U_k, V_k, D. Output: $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

```
1. Compute $l$ largest singular triplets of $(I - U_kU_k^T)D$ in (4.12).
2. Construct $H$ in (4.13). Compute the matrices $\Theta_k$, $F_k$, and $G_k$ that correspond to the $k$ dominant
   singular triplets of $H$.
3. Set $\tilde{\Sigma}_k = \Theta_k$, $\tilde{U}_k = [U_k,\ X_l]F_k$, and $\tilde{V}_k = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix} G_k$.
```

Observe that the projected matrix produced by Algorithm 4.1 has significantly fewer rows than the one in Algorithm 3.1, bringing the $\mathcal{O}((k+p)^3)$ complexity of step 2 to $\mathcal{O}((k+p)(k+l)^2)$, which is linear in p. From a practical point of view, note also that, in contrast to Algorithm 3.1, $(I - U_kU_k^T)D$ in step 1 should not be explicitly constructed, because the factorization (4.12) can be obtained by an iterative procedure, e.g., [4, 14], that accesses $(I-U_kU_k^T)D$ (and its transpose) through matrix-vector multiplications with D (and D^T) and orthogonalizations against the columns of U_k. The triplets S_l, X_l, and Y_l need to be approximated only with modest accuracy.

Similar considerations can be exploited for the case of adding new terms with the aim of constructing Z_l in (4.4)–(4.5) satisfying (4.11) with $M \approx \left(I - V_kV_k^T\right) T^T \in \mathbb{R}^{n\times p}$ and $\text{rank}(M) = l$. Let S_l, $X_l \in \mathbb{R}^{n\times l}$, and $Y_l \in \mathbb{R}^{p\times l}$ now be the factors of the rank-l SVD approximation of the matrix $(I - V_kV_k^T)T^T$, i.e.,

$$(I - V_kV_k^T)T^TY_l = X_lS_l, \quad T(I - V_kV_k^T)X_l = Y_lS_l \,. \tag{4.14}$$

An optimal rank-l approximation of $(I - V_kV_k^T)T^T$ is then given by $M = X_lS_lY_l^T$, and analogy with the previous case suggests that we choose $Z_l = X_l$. Thus, by Proposition 4.2 and (4.14), a run of SV-RR(A_T, U, $\bar{V}$) with U defined in (3.15) and $\bar{V} = [V_k,\ X_l]$ gives the projected matrix

$$H = \begin{bmatrix} \Sigma_k & 0 \\ TV_k & Y_lS_l \end{bmatrix} \in \mathbb{R}^{(k+p)\times(k+l)} \,. \tag{4.15}$$

As a result, we obtain the following updating scheme.

Algorithm 4.2 (Adding terms (SV)). *Input:* Σ_k, U_k, V_k, T. *Output:* $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. *Compute* l *largest singular triplets of* $(I - V_kV_k^T)T^T$ *in (4.14).*
2. *Construct* H *in (4.15). Compute the matrices* Θ_k, F_k, *and* G_k *that correspond to the* k *dominant singular triplets of* H.
3. *Set* $\tilde{\Sigma}_k = \Theta_k$, $\tilde{U}_k = \begin{bmatrix} U_k & 0 \\ 0 & I_p \end{bmatrix} F_k$, *and* $\tilde{V}_k = [V_k,\ X_l]G_k$.

Finally, in the case of correcting the term weights, we would like to specify $Z_{l_1}^{(1)}$ and $Z_{l_2}^{(2)}$ in (4.7)–(4.9), such that

$$\texttt{span}(Z_{l_1}^{(1)}) = \texttt{range}(M_1), \quad \texttt{span}(Z_{l_2}^{(2)}) = \texttt{range}(M_2) \,, \tag{4.16}$$

where $M_1 \approx \left(I - U_kU_k^T\right) C \in \mathbb{R}^{m\times p}$, $M_2 \approx \left(I - V_kV_k^T\right) W^T \in \mathbb{R}^{n\times p}$; $\text{rank}(M_1) = l_1$ and $\text{rank}(M_2) = l_2$. By analogy with the previous cases, we let $S_{l_1}^{(1)}$, $X_{l_1}^{(1)} \in \mathbb{R}^{m\times l_1}$, $Y_{l_1}^{(1)} \in \mathbb{R}^{p\times l_1}$ be the matrices of the l_1 largest singular values and associated left and right singular vectors of $(I - U_kU_k^T)C$,

$$(I - U_kU_k^T)CY_{l_1}^{(1)} = X_{l_1}^{(1)}S_{l_1}^{(1)}, \quad C^T(I - U_kU_k^T)X_{l_1}^{(1)} = Y_{l_1}^{(1)}S_{l_1}^{(1)} \,, \tag{4.17}$$

and $S_{l_2}^{(2)}$, $X_{l_2}^{(2)} \in \mathbb{R}^{n\times l_2}$, $Y_{l_2}^{(2)} \in \mathbb{R}^{p\times l_2}$ correspond to the l_2 dominant triplets of $(I - V_kV_k^T)W^T$,

$$(I - V_kV_k^T)W^TY_{l_2}^{(2)} = X_{l_2}^{(2)}S_{l_2}^{(2)}, \quad W(I - V_kV_k^T)X_{l_2}^{(2)} = Y_{l_2}^{(2)}S_{l_2}^{(2)} \,. \tag{4.18}$$

Optimal low-rank approximations of $(I-U_kU_k^T)C$ and $(I-V_kV_k^T)W^T$ are then given by $M_1 = X_{l_1}^{(1)}S_{l_1}^{(1)}Y_{l_1}^{(1)T}$ and $M_2 = X_{l_2}^{(2)}S_{l_2}^{(2)}Y_{l_2}^{(2)T}$, respectively. Hence, we choose $Z_{l_1}^{(1)} = X_{l_1}^{(1)}$ and $Z_{l_2}^{(2)} = X_{l_2}^{(2)}$. In this case, by Proposition 4.3 and (4.17)–(4.18), a run of SV-RR(A_{CW}, $\bar{U}$, $\bar{V}$) with $\bar{U} = [U_k,\ X_{l_1}^{(1)}]$ and $\bar{V} = [V_k,\ X_{l_2}^{(2)}]$ yields the projected matrix

$$H = \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_k^TC \\ S_{l_1}^{(1)}Y_{l_1}^{(1)T} \end{bmatrix} \begin{bmatrix} WV_k, & Y_{l_2}^{(2)}S_{l_2}^{(2)} \end{bmatrix} \in \mathbb{R}^{(k+l_1)\times(k+l_2)} \,, \tag{4.19}$$

and can be summarized as the following updating algorithm.

Algorithm 4.3 (Correcting weights (SV)). *Input:* Σ_k, U_k, V_k, C, W. *Output:* $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. *Compute* l_1 *largest singular triplets of* $(I - U_kU_k^T)C$ *in (4.17) and* l_2 *largest singular triplets of* $(I - V_kV_k^T)W^T$ *in (4.18)* .
2. *Construct* H *in (4.19). Compute the matrices* Θ_k, F_k, *and* G_k *that correspond to the* k *dominant singular triplets of* H.
3. *Set* $\tilde{\Sigma}_k = \Theta_k$, $\tilde{U}_k = [U_k,\ X_{l_1}^{(1)}]F_k$, *and* $\tilde{V}_k = [V_k,\ X_{l_2}^{(2)}]G_k$.

4.2. The Golub-Kahan-Lanczos vectors. In this subsection, we consider an alternative option for choosing the orthonormal vectors Z_l, $Z^{(1)}_{l_1}$, and $Z^{(2)}_{l_2}$. Instead of using the dominant singular triplets of the four respective matrices $(I-U_kU_k^T)D$, $(I-V_kV_k^T)T^T$, $(I-U_kU_k^T)C$, and $(I-V_kV_k^T)W^T$, under consideration, we propose to exploit for the same purpose a few basis vectors computed from the Golub-Kahan-Lanczos (GKL) bidiagonalization procedure [11, 12].

Given an arbitrary m-by-n matrix A, the GKL procedure constructs the orthonormal bases $P_l = [p_1, \ldots, p_l]$ and $Q_{l+1} = [q_1, \ldots, q_{l+1}]$ of the *Krylov subspaces* $\mathcal{K}_l(AA^T, Aq_1) = \mathtt{span}\left\{Aq_1, (AA^T)Aq_1, \ldots (AA^T)^{l-1}Aq_1\right\}$ and $\mathcal{K}_{l+1}(A^TA, q_1) =$ $\mathtt{span}\left\{q_1, (A^TA)q_1, \ldots (A^TA)^l q_1\right\}$, respectively; and an (upper) bidiagonal matrix $\underline{B}_l \in \mathbb{R}^{l\times(l+1)}$. The matrices P_l, Q_{l+1}, and $\underline{B}_l$ are related by the fundamental identity

$$\left\{\begin{array}{lcl} AQ_l & = & P_lB_l\ , \\ A^TP_l & = & Q_{l+1}\underline{B}_l^T\ ; \end{array}\right. \tag{4.20}$$

where $B_l \in \mathbb{R}^{l\times l}$ is obtained from $\underline{B}_l$ by removing the last column. The vectors p_i and q_i that comprise the matrices P_l and Q_l are called the *left and right GKL vectors*, respectively. An implementation of the procedure is given by the algorithm below, which we further refer to as GKL(A, l).

Algorithm 4.4 (GKL(A, l)). Input: $A \in \mathbb{R}^{m\times n}$, l. Output: $\underline{B}_l$, P_l, Q_{l+1}.

```
1.  Choose $q_1$, $\|q_1\| = 1$. Set $\beta_0 = 0$ .
2.  For $i = 1, \ldots, l$ do
3.      $p_i = Aq_i - \beta_{i-1}p_{i-1}$
4.      If $(m < n)$ Orthogonalize $p_i$ against $[p_1, \ldots, p_{i-1}]$.
5.      $\alpha_i = \|p_i\|$.
6.      $p_i = p_i/\alpha_i$.
7.      $q_{i+1} = A^Tp_i - \alpha_iq_i$.
8.      If $(m \ge n)$ Orthogonalize $q_{i+1}$ against $[q_1, \ldots, q_i]$.
9.      $\beta_i = \|q_{i+1}\|$.
10.     $q_{i+1} = q_{i+1}/\beta_i$.
11. EndFor
12. Return $\underline{B}_l = \mathrm{diag}\{\alpha_1, \ldots, \alpha_l\} + \mathrm{superdiag}\{\beta_1, \ldots, \beta_l\}$, $P_l = [p_1, \ldots, p_l]$, and $Q_{l+1} = [q_1, \ldots, q_{l+1}]$.
```

Note that, in exact arithmetic, steps 4 and 8 of the algorithm are unnecessary, i.e., orthogonality of vectors p_i and q_i is ensured solely by the short-term recurrences in steps 3 and 6. In the presence of the round-off, it is generally required to reorthogonalize both p_i and q_i against all previously constructed vectors. However, in Algorithm 4.4 we follow the observation in [24] suggesting that it is possible to reorthogonalize only one of the vector sets without significant loss of orthogonality for the other.

Let us first address the case of adding new documents, and assume that $P_l \in \mathbb{R}^{m\times l}$, $Q_{l+1} \in \mathbb{R}^{p\times(l+1)}$, and $\underline{B}_l$ are produced by a run of GKL($(I - U_kU_k^T)D$, l). Then as a rank-l approximation of $(I - U_kU_k^T)D$ we choose $M = P_l\underline{B}_lQ_{l+1}^T$ and set $Z_l = P_l$, i.e., we define $\bar{U}$ in (4.1) by augmenting U_k with several left GKL vectors of $(I - U_kU_k^T)D$.

In contrast to the case of the singular vectors, this choice does not provide an optimal rank-l approximation. The optimality, however, is traded for a simpler and faster procedure to construct Z_l. Evaluation of the performance of the GKL vectors as opposed to the singular vectors in the context of the dimension reduction can be found in [8].

By (4.20), we have

$$\left\{\begin{array}{lcl} (I - U_kU_k^T)DQ_l & = & P_lB_l \\ D^T(I - U_kU_k^T)P_l & = & Q_{l+1}\underline{B}_l^T\ . \end{array}\right. \tag{4.21}$$

It can be seen from the above relation that $Z_l = P_l$ satisfies (4.2). Thus, by Proposition 4.1, the projected matrix produced by an application of SVD-RR(A_D, $\bar{U}$, V), where $\bar{U} = [U_k,\ P_l]$ and V is defined in (3.14), takes the form

$$H = \left[\begin{array}{cc} \Sigma_k & U_k^TD \\ 0 & \underline{B}_lQ_{l+1}^T \end{array}\right] \in \mathbb{R}^{(k+l)\times(k+p)}\ . \tag{4.22}$$

As a result, we obtain the following updating algorithm.

Algorithm 4.5 (Adding documents (GKL)). Input: Σ_k, U_k, V_k, D. Output: $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. *Apply* GKL$((I-U_kU_k^T)D,\, l)$*, given by Algorithm* 4.4*, to produce* P_l*,* Q_{l+1}*, and* $\underline{B}_l$ *satisfying (4.21).*
2. *Construct* H *in (4.22). Compute the matrices* Θ_k*,* F_k*, and* G_k *that correspond to the* k *dominant singular triplets of* H*.*
3. *Set* $\tilde{\Sigma}_k = \Theta_k$*,* $\tilde{U}_k = [U_k,\ P_l]F_k$*, and* $\tilde{V}_k = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix} G_k$*.*

Note that applying the GKL procedure in step 1 of Algorithm 4.5 can be carried out without explicitly constructing the matrix $(I-U_kU_k^T)D$ and its transpose. Instead, the matrices can be accessed through matrix-vector multiplications with D or D^T, and orthogonalizations against the columns of U_k.

If new terms T are added to the document collection, then l steps of the GKL procedure should be applied to the matrix $(I-V_kV_k^T)T^T$. This leads to the identity

$$\left\{ \begin{array}{lcl} (I-V_kV_k^T)T^TQ_l & = & P_lB_l \\ T(I-V_kV_k^T)P_l & = & Q_{l+1}\underline{B}_l^T \end{array} \right. , \tag{4.23}$$

where $P_l \in \mathbb{R}^{n\times l}$ and $Q_{l+1} \in \mathbb{R}^{p\times(l+1)}$ are the matrices of the left and right GKL vectors of $(I-V_kV_k^T)T^T$. Similar to the previous case, we approximate $(I-V_kV_k^T)T^T$ by $M = P_l\underline{B}_lQ_{l+1}^T$, and set $Z_l = P_l$. Then, combining Proposition 4.2 and (4.23), we obtain the projected matrix

$$H = \begin{bmatrix} \Sigma_k & 0 \\ TV_k & Q_{l+1}\underline{B}_l^T \end{bmatrix} \in \mathbb{R}^{(k+p)\times(k+l)} , \tag{4.24}$$

which is produced by SVD-RR$(A_T,\, U,\, \bar{V})$ with U in (3.15) and $\bar{V} = [V_k,\ P_l]$. The resulting updating scheme is summarized in the algorithm below.

Algorithm 4.6 (Adding terms (GKL)). *Input:* Σ_k*,* U_k*,* V_k*,* D*. Output:* $\tilde{\Sigma}_k$*,* $\tilde{U}_k$*,* $\tilde{V}_k$*.*
1. *Apply* GKL$((I-V_kV_k^T)T^T,\, l)$*, given by Algorithm* 4.4*, to produce* P_l*,* Q_{l+1}*, and* $\underline{B}_l$ *satisfying (4.23).*
2. *Construct* H *in (4.24). Compute the matrices* Θ_k*,* F_k*, and* G_k *that correspond to the* k *dominant singular triplets of* H*.*
3. *Set* $\tilde{\Sigma}_k = \Theta_k$*,* $\tilde{U}_k = \begin{bmatrix} U_k & 0 \\ 0 & I_p \end{bmatrix} F_k$*, and* $\tilde{V}_k = [V_k,\ P_l]G_k$ *.*

Finally, if the term weights are corrected, the GKL procedure is applied to both $(I-U_kU_k^T)C$ and $(I-V_kV_k^T)W$. A run of GKL$((I-U_kU_k^T)C,\, l_1)$ gives the equality

$$\left\{ \begin{array}{lcl} (I-U_kU_k^T)CQ_{l_1}^{(1)} & = & P_{l_1}^{(1)}B_{l_1}^{(1)} \\ C^T(I-U_kU_k^T)P_{l_1}^{(1)} & = & Q_{l_1+1}^{(1)}\underline{B}_{l_1}^{(1)T} \end{array} \right. , \tag{4.25}$$

and GKL$((I-V_kV_k^T)W,\, l_2)$ results in

$$\left\{ \begin{array}{lcl} (I-V_kV_k^T)W^TQ_{l_2}^{(2)} & = & P_{l_2}^{(2)}B_{l_2}^{(2)} \\ W(I-V_kV_k^T)P_{l_2}^{(2)} & = & Q_{l_2+1}^{(2)}\underline{B}_{l_2}^{(2)T} \end{array} \right. . \tag{4.26}$$

In (4.25)–(4.26), the columns of the matrix pairs $P_{l_1}^{(1)} \in \mathbb{R}^{m\times l_1}$, $Q_{l_1+1}^{(1)} \in \mathbb{R}^{p\times(l_1+1)}$ and $P_{l_2}^{(2)} \in \mathbb{R}^{n\times l_2}$, $Q_{l_2+1}^{(2)} \in \mathbb{R}^{p\times(l_2+1)}$ correspond to the left and right GKL vectors of $(I-U_kU_k^T)C$ and $(I-V_kV_k^T)W$, respectively. We then can approximate $(I-U_kU_k^T)C$ by $M_1 = P_{l_1}^{(1)}\underline{B}_{l_1}^{(1)}Q_{l_1+1}^{(1)T}$ and $(I-V_kV_k^T)W$ by $M_2 = P_{l_2}^{(2)}\underline{B}_{l_2}^{(2)}Q_{l_2+1}^{(2)T}$, choosing $Z_{l_1}^{(1)} = P_{l_1}^{(1)}$ and $Z_{l_2}^{(2} = P_{l_2}^{(2)}$. Hence, by Proposition 4.3 and (4.25)–(4.26), a run of SVD-RR$(A_{CW},\, \bar{U},\, \bar{V})$ with $\bar{U} = [U_k,\ P_{l_1}^{(1)}]$ and $\bar{V} = [V_k,\ P_{l_2}^{(2)}]$ yields the projected matrix

$$H = \begin{bmatrix} \Sigma_k & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} U_k^TC \\ \underline{B}_{l_1}^{(1)}Q_{l_1+1}^{(1)T} \end{bmatrix} \left[WV_k,\ Q_{l_2+1}^{(2)}\underline{B}_{l_2}^{(2)T} \right] \in \mathbb{R}^{(k+l_1)\times(k+l_2)} . \tag{4.27}$$

This suggests the following updating scheme.

Algorithm 4.7 (Correcting weights (GKL)). *Input:* Σ_k*,* U_k*,* V_k*,* D*. Output:* $\tilde{\Sigma}_k$*,* $\tilde{U}_k$*,* $\tilde{V}_k$*.*
1. *Apply* GKL$((I-U_kU_k^T)C,\, l_1)$ *and* GKL$((I-V_kV_k^T)W^T,\, l_2)$*, given by Algorithm* 4.4*, to produce the GKL vectors satisfying (4.25) and (4.26).*

Algorithm	Complexity
Alg. 3.1	$(k+p)^3 + mp^2 + mpk + k^2(m+n)$
Alg. 4.1	$(k+p)(k+l)^2 + ml^2 + mlk + k^2(m+n) + \texttt{nnz}(D)(l+k)$
Alg. 4.5	$(k+p)(k+l)^2 + mlk + k^2(m+n) + \texttt{nnz}(D)(l+k)$

TABLE 4.1
Asymptotic complexity of different LSI updating schemes for adding new documents.

Algorithm	Complexity
Alg. 3.2	$(k+p)^3 + np^2 + npk + k^2(m+n)$
Alg. 4.2	$(k+p)(k+l)^2 + nl^2 + nlk + k^2(m+n) + \texttt{nnz}(D)(l+k)$
Alg. 4.6	$(k+p)(k+l)^2 + nlk + k^2(m+n) + \texttt{nnz}(D)(l+k)$

TABLE 4.2
Asymptotic complexity of different LSI updating schemes for adding new terms.

2. Construct H in *(4.27)*. Compute the matrices Θ_k, F_k, and G_k that correspond to the k dominant singular triplets of H.
3. Set $\tilde{\Sigma}_k = \Theta_k$, $\tilde{U}_k = [U_k,\ P^{(1)}_{l_1}]F_k$, and $V_k = [V_k,\ P^{(2)}_{l_2}]G_k$.

4.3. Complexity analysis. We start by addressing the cost of Algorithm 4.1 that performs the updating after adding new documents. To be specific, we assume that a multiple of l iterations of a GKL-based singular value solver, e.g., [14], are used to determine the l dominant singular triplets of $(I - U_kU_k^T)D$. In this case, the cost of step 1 of the algorithm can be estimated as $\mathcal{O}(l\texttt{nnz}(D) + lmk + l^2(m+p))$, where $\texttt{nnz}(D)$ is the number of non-zero elements in D. Here the first two terms come from the matrix-vector multiplications with $(I - U_kU_k^T)D$. In particular, the first term is given by the Sparse Matrix-Vector multiplications (SpMV's) with D and D^T, and the second term is contributed by orthogonalizations against the columns of U_k. The last term accounts for reorthogonalizations of the GKL vectors and the final SV-RR procedure to extract the singular triplet approximations from the corresponding Krylov subspaces.

The complexity of step 2 of Algorithm 4.1 is $\mathcal{O}((k+p)(k+l)^2 + \texttt{nnz}(D)k + lp)$. The first term represents the cost of the SVD of the matrix H in (4.13). The remaining terms are given by the construction of U_k^TD (utilizing SpMV's with D) and the multiplication of Y_l^T by the diagonal matrix S_l to form the $(1,2)$ and $(2,2)$ blocks of H, respectively. The cost of step 3 is $\mathcal{O}(k^2(m+n) + lmk)$.

The complexity of Algorithm 4.5 that also addresses the case of adding new documents and extends Algorithm 4.1 by replacing the singular vectors of $(I - U_kU_k^T)D$ with the GKL vectors is similarly estimated. The difference in cost of the two schemes essentially comes from their initial step. In particular, the complexity of step 1 of Algorithm 4.5 is $\mathcal{O}(l\texttt{nnz}(D) + lmk + l^2p)$, i.e., it requires $\mathcal{O}(l^2m)$ less operations than the corresponding step of Algorithm 4.1. This cost reduction is due to avoiding the SV-RR procedure that is invoked by a GKL-based singular value solver in step 1 of Algorithm 4.1.

The overall complexities of Algorithms 4.1 and 4.5, as well as of Algorithm 3.1 addressed at the beginning of Section 4, are summarized in Table 4.1.

The main observation drawn from Tables 4.1–4.3 is that, unlike the Zha-Simon approaches, the new updating schemes no longer exhibit the cubic scaling in p. It can be seen that the proposed algorithms scale linearly in the update size, which leads to non-negligible computational savings, especially in context of large text collections, as demonstrated in our TREC8 example in Section 5.

We also note that the new schemes allow one to take advantage of the sparsity of the update, and require less storage due to working with sparse or thinner dense matrices and due to not storing the triangular factors from the QR decompositions. The recent trend of reducing the amount of in-core memory on emerging architectures makes this property particularly important in the context of high performance text mining. In this situation, the execution time may be dominated by out-of-core memory accesses and data movement, which motivates the development of novel memory-efficient algorithms.

If $l = 0$ then the introduced algorithms turn into the early methods in [3, 19]. As seen from Tables 4.1–4.3 after setting $l = 0$, these methods can be extremely fast, but generally yield modest retrieval accuracy. This is not surprising. In the language of the present paper, the schemes [3, 19] represent the SV-RR

Algorithm	Complexity
Alg. 3.3	$(k+p)^3 + (m+n)(p^2 + pk + k^2)$
Alg. 4.3	$(k+l)^3 + (m+n)(l^2 + lk + k^2) + (\texttt{nnz}(C) + \texttt{nnz}(W))(l+k)$
Alg. 4.7	$(k+l)^3 + (m+n)(lk + k^2) + (\texttt{nnz}(C) + \texttt{nnz}(W))(l+k)$

TABLE 4.3
Asymptotic complexity of different updating schemes for correcrting the term weights.

procedures where the left, right or both search subspaces stay unmodified during the matrix updates, e.g., always $\bar{U} = U_k$ or $\bar{V} = V_k$, and are “too small” to provide satisfactory retieval quality.

If $l = p$ then the costs of the proposed schemes resemble those of the Zha-Simon methods [27]. In this case, the search subspaces become sufficiently large to ensure the inclusion of the dominant singular subspaces and, similar to the algorithms in [27], produce the exact triplets of A_D, A_T, or A_{CW}. As has been discussed, this fixes the problem of the deteriorating retrieval accuracy, however, it may result in loss of the efficiency if p is large.

The updating methods of this paper can balance computational expenses by appropriate choices of l and can be placed in between the two extremes that correspond to [3, 19] and [27]. In the next section, we demonstrate that l can be chosen small without sacrificing retireval accuracy.

5. Numerical experiments.

The goal of this section is to demonstrate the differences in cost and accuracy of the discussed updating schemes, and to verify the results of the complexity analysis in subsection 4.3.

In our experiments we apply the updating algorithms to several standard document collections, such as MEDLINE, CRANFIELD, NPL, and TREC8. These datasets have sizes that range from a few thousands of terms and documents to hundreds of thousands, and are commonly used to benchmark the performance of text mining techniques. Each collection is supplied with a set of n_q “canonical” queries and expert-generated lists of relevant documents corresponding to these queries. The listed relevant documents are the ones that should be ideally returned in response to the query, and therefore provide information necessary to evaluate the accuracy of automatic retrieval.

The MEDLINE, CRANFIELD, and NPL term-document matrices are available from ftp://ftp.cs.cornell.edu/pub/smart/. The TMG software [26] has been used to parse the TREC8 dataset and generate the term-document matrix. The standard pre-processing has included stemming and deleting common words according to the TMG-provided stop-list. Terms with no more than 5 occurrences or with appearances in more than 100,000 documents have been removed and 125 empty documents have been ignored. In all tests the weighting schemes for documents and queries have been set to `lxn.bpx` [17, 22].

Due to space limitations we present results only for the case of adding new documents. Our experience with the other update types has lead to similar observations and conclusions.

The tests are organized as follows. Given a term-document matrix A, we fix its first t columns, and compute the k dominant singular triplets Σ_k, U_k, and V_k of the corresponding submatrix of A. This submatrix, denoted using the “MATLAB notation” by $A(:, 1:t)$, represents the initial state of the document collection. To simulate the arrival of new documents, the remaining $n - t$ columns are consecutively added in groups of p columns. Thus, for the first update, we append the submatrix $A(:, t+1 : t+p)$, for the next one $A(:, t+p+1 : t+2p)$, etc. As a result, in our experiments, we nearly double (and sometimes triple) the initial size t of the collection. To track the effects of the update size on the efficiency of the updating schemes, for each dataset we consider several different values of p.

After adding each column group, we update the k singular triplets using the proposed Algorithm 4.1 (“SV”) and Algorithm 4.5 (“GKL”), as well as the existing scheme of Zha and Simon in Algorithm 3.1 (“ZS”). Thereafter, the similarity scores (1.1) with $\alpha = 0$ are evaluated for each “canonical” query, and the average precisions are calculated using the standard 11-point formula, e.g., [13, 17]. Since a total of n_q “canonocal” queries are provided for each dataset, we calculate the *mean* of the corresponding n_q average precisions and plot the resulting quantity against the current number of documents in the collection. We also measure time spent for each update.

All experiments have been performed in MATLAB. Note that, with this testing environment, little or nothing can be deduced about the methods' capabilities in terms of the *absolute* execution times. However, the timing results can provide illustrations for the complexity findings in Tables 4.1–4.3 and give a comparison of the costst of the schemes *relative to each other*. These comparisons are meaningful when the same conditions are used in each case, e.g., the same MATLAB function is used for computing the partial SVD and the same optimization techniques, if any, are applied.

Due to the established unifying framework of the SV-RR procedure, our codes are organized to follow the same execution path and differ only in "localized" subtasks, such as, e.g., computing a few dominant singular triplets in step 1 of Algorithm 4.1 or the GKL vectors in step 1 of Algorithm 4.5 instead of the QR decomposition of $(I - U_k U_k^T)D$ in the original Algorithm 3.1. All the linear algebra operations, such as dense matrix-matrix multiplications, the SVD and QR decompositions, SpMV's, etc., are accomplished by the same MATLAB routines in all of the updating schemes.

In step 1 of Algorithm 4.5, we use our own straightforward MATLAB implementation of the GKL procedure, which is based on Algorithm 4.4. Our singular value solver in step 1 of Algorithm 4.1 is built on top of this GKL procedure by additionally performing the SV-RR computation with respect to the left and right GKL vectors. The latter amounts to the SVD of a bidiagonal matrix followed by the construction of the Ritz singular vectors. The convergence of the singular triplets is achieved, and the solver is stopped, after the difference between the sums of the k dominant singular value approximations on two consecutive iterations becomes smaller than 10^{-1}. The maximum number of iterations is set to p.

The starting vector in the GKL algorithm is always set to the vector of all ones and then normalized to have unit lenght. We have observed that this choice often leads to a higher retrieval accuracy compared to a random vector. The reduced dimensions k are set to (nearly) optimal, in the retrieval accuracy, values as observed in [8, Figures 5-6]. The parameter l that determines the number of singular triplets in step 1 of Algorithm 4.1 or the number of the GKL vectors in step 1 of Algorithm 4.5 is experimentally chosen to provide balance between the computational cost and retrieval accuracy.

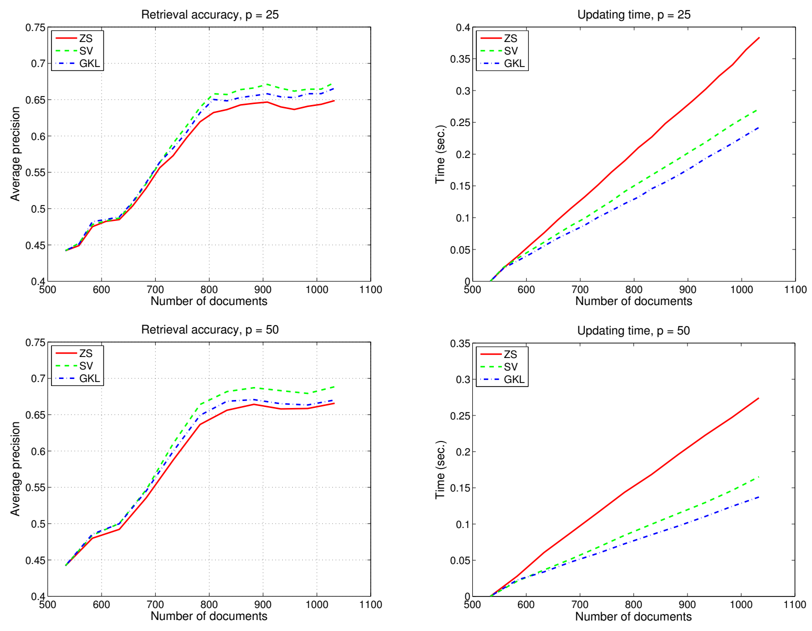

Fig. 5.1. *MEDLINE collection: $m = 7,014$, $n = 1,033$, $k = 75$, $t = 533$, $n_q = 30$. The average precision and time for adding groups of $p = 25$ (top) and $p = 50$ (bottom) documents. The number of singular triplets of $(I - U_k U_k^T)D$ computed by Algorithm 4.1 ("SV") is 2 (top) and 4 (bottom). The number of GKL steps in Algorithm 4.5 ("GKL") is 3 (top) and 5 (bottom). The methods are compared to Algorithm 3.1 ("ZS").*

Figure 5.1 compares different updating schemes for the MEDLINE collection. This collection is known to be small, with the term-document matrix having $m = 7,014$ rows and $n = 1,033$ columns. We fix the initial $t = 533$ columns and add the rest in groups of $p = 25$ (top) and $p = 50$ (bottom). The size k of the reduced subspace is set to 75.

The left-hand side plots demonstrate the differences in the retrieval accuracy. The plots to the right compare the timing results. The horizontal axes represent the number of documents in the collection after consecutive updates. The vertical axes correspond to the mean of the average precisions (left) and the *cumulative* time (right), i.e., the total time spent by the algorithms to perform the current and preceding updates.

The example demonstrates that the number l of the singular and GKL vectors generated by Algorithms 4.1 and 4.5 can be very small. In particular, we construct as few as 2-3 singular triplets and 3-4 GKL vectors. As anticipated from the discussion in subsection 4.3, this fact gives rise to updating schemes that are significantly faster than the Zha-Simon approach in Algorithm 3.1. Figure 5.1 (right) confirms that the new strategies in Algorithms 4.1 and 4.5 are indeed considerably faster than the existing scheme. Remarkably, the gain in the efficiency comes without any loss of the retrieval accuracy. In fact, the contrary is observed in that the new algorithms respond with a higher average precision; see Figure 5.1 (left).

It can be observed from the figure that Algorithm 4.1 gives the most accurate results though it requires slightly more compute time than Algorithm 4.5. This time difference is caused by the overhead inflicted by the singular value solver to ensure the convergence and perform the extraction of the approximate singular triplets in step 1 of Algorithm 4.1. Note that the number of GKL vectors in Algorithm 4.5 is slightly larger than that of the singular triplets in Algorithm 4.1. This represents a "compensation" for the non-optimal choice of the low-rank approximation of $(I - U_k U_k^T)D$ adopted by Algorithm 4.5. Further increase in the number of the GKL vectors may lead to higher retrieval accuracies, but the timing is accordingly affected. The latter observation is true for all of our experiments and is consistent with the relevant results in [8].

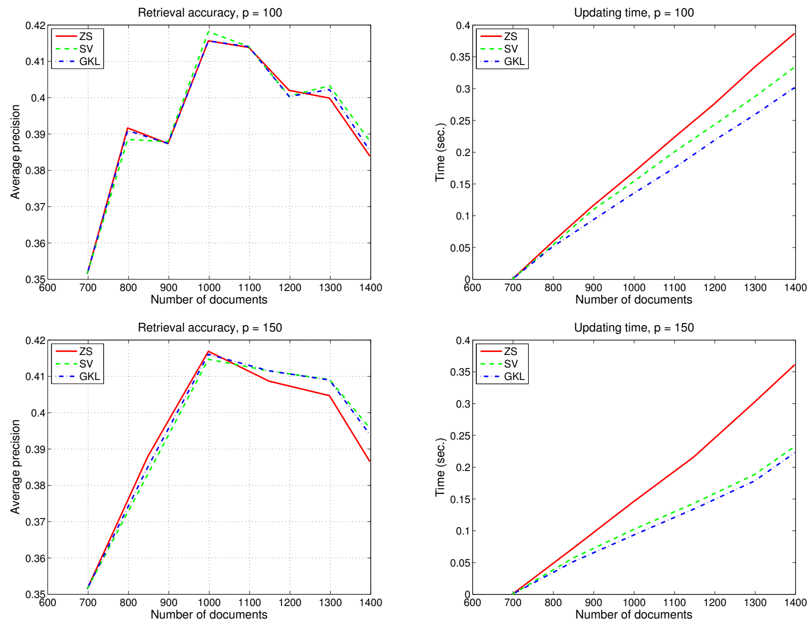

FIG. 5.2. *CRANFIELD collection:* $m = 3,763$, $n = 1,398$, $k = 150$, $t = 698$, $n_q = 225$. *The average precision and time for adding groups of* $p = 100$ *(top) and* $p = 150$ *(bottom) documents. The number of singular triplets of* $(I - U_k U_k^T)D$ *computed by Algorithm 4.1 ("SV") is 25 (top and bottom). The number of GKL steps in Algorithm 4.5 ("GKL") is 51 (top) and 45 (bottom). The methods are compared to Algorithm 3.1 ("ZS").*

Figure 5.2 displays the results for the CRANFIELD collection. Similar to MEDLINE, CRANFIELD represents another example of a small text collection, with the term-document matrix having $m = 3,763$ rows and $n = 1,398$ columns. Following our test framework, we fix the initial $t = 698$ columns and add the rest in groups of $p = 100$ (top) and $p = 150$ (bottom). The dimension k of the reduced subspace is set to 150.

The number l of singular triplets in Algorithm 4.1 is chosen to be 25 for both values of p. The number of GKL steps is set to 51 ($p = 100$) and 45 ($p = 150$). In contrast to the previous example, smaller values of l fail to deliver acceptable retrieval accuracies. Nevertheless, as can be seen in Figure 5.2 (right), the new updating schemes are still noticeably faster than Algorithm 3.1. The retrieval accuracy is comparable for all three approaches, and is slightly higher for the new schemes at the later updates.

In Figure 5.3 we report results for the NPL collection. NPL is a larger text collection with the associated term-document matrix having $m = 7,491$ rows and $n = 11,429$ columns. Note that, in contrast to the

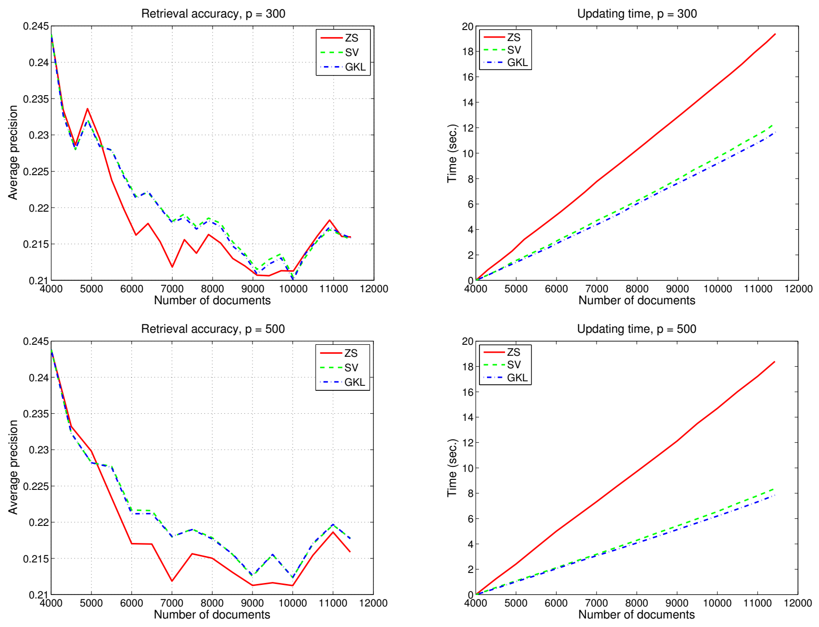

FIG. 5.3. *NPL collection: $m = 7,491$, $n = 11,429$, $k = 550$, $t = 4,000$, $n_q = 93$. The average precision and time for adding groups of $p = 300$ (top) and $p = 500$ (bottom) documents. The number of singular triplets of $(I - U_k U_k^T)D$ computed by Algorithm 4.1 ("SV") is 10 (top and bottom). The number of GKL steps in Algorithm 4.5 ("GKL") is 20 (top and bottom). The methods are compared to Algorithm 3.1 ("ZS").*

previous cases, here the number of rows is smaller than the number of columns. In this sense, NPL provides a more representative example of the real-world large-scale text collections, where the number of terms in the vocabulary is limited while the number of documents, in principle, can become arbitrarily large. For the test purposes we fix $t = 4,000$ initial columns and add the rest in groups of $p = 300$ and $p = 500$; $k = 550$.

Figure 5.3 shows that for the NPL dataset the new updating schemes are also superior to the existing approach both in the retrieval accuracy and compute time. Note that the number l of the singular and GKL vectors is kept reasonably low. In particular, we request 10 singular triplets in step 1 of Algorithm 4.1 and 20 GKL vectors in step 1 of Algorithm 4.5.

Figure 5.4 concerns a larger example given by the TREC8 dataset. This dataset is known to be extensively used for testing new text mining algorithms. It is comprised of four document collections (*Financial Times, Federal Register, Foreign Broadcast Information Service, and Los Angeles Times*) from the TREC CDs 4 & 5 (copyrighted). The queries are from the TREC-8 *ad hoc* task; see `http://trec.nist.gov/data/topics_eng/`. The relevance files are available at `http://trec.nist.gov/data/qrels_eng/`.

After the pre-processing step discussed at the beginning of this section, TREC8 delivers a term-document matrix with total of $m = 138,232$ terms and $n = 528,028$ documents. In our experiment, we fix the initial $t = 90,000$ columns and then incrementally add the new columns in groups of $p = 500$ (top) and $p = 1,000$ (bottom) until their total number reaches 300,000. In both cases, the value of l is relatively small: we use only 10 singular triplets and 20 GKL vectors.

As can be seen in Figure 5.4, the gain in the efficiency presented by the new schemes becomes even more pronounced when the methods are applied to a larger dataset. In particular, our tests show a three-fold speed-up of the overall updating process (420 sequential updates) for $p = 500$ and a five-fold speed-up (210 sequential updates) for $p = 1,000$. Yet, in both cases, the proposed algorithms deliver a higher retrieval accuracy.

In Figure 5.4 we also report the results for schemes [3, 19] ("OB"). [1] As has been previously discussed, these schemes are known to be fast but generally lack accuracy. This is confirmed by our experiment. Remarkably, the methods proposed in this paper are essentially as fast as those in [3, 19], but the accuracy is higher than that of Zha-Simon schemes [27]. Note that in contrast to [3, 19], which can be obtained by

[1]The abbreviation is after the name of the author of [19].

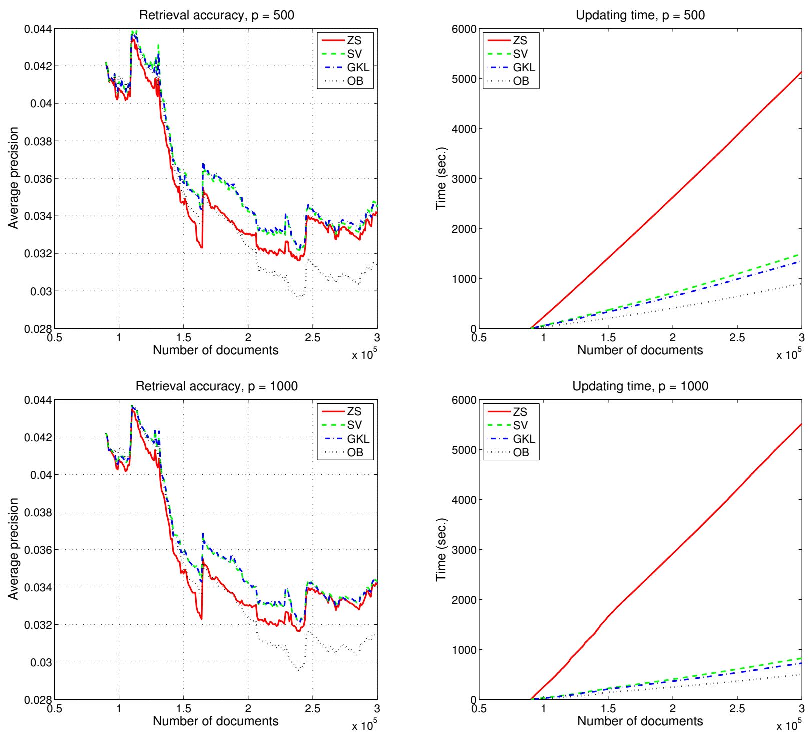

FIG. 5.4. *TREC8 collection: $m = 138,232$, $n = 528,028$, $k = 400$, $t = 90,000$, $n_q = 50$. The average precision and time for adding groups of $p = 500$ (top) and $p = 1000$ (bottom) documents. The number of singular triplets of $(I - U_k U_k^T)D$ computed by Algorithm 4.1 ("SV") is 10. The number of GKL steps in Algorithm 4.5 ("GKL") is 20. The methods are compared to Algorithm 3.1 ("ZS") and the updating scheme* [3, 19] *("OB").*

setting $l = 0$ in our algorithms, the presence of a nonzero number l of singular or GKL vectors is indeed critical for maintaining the accuracy.

6. Conclusion.

This paper introduces several new algorithms for the SVD updating problem in LSI. The proposed schemes are based on classical projection methods applied to the singular value computations. A key ingredient of the new algorithms is the construction of low-dimensional search subspaces. A proper choice of such subspaces leads to fast updating schemes with a modest storage requirement.

In particular, we consider two options for reducing the dimensionality of search subspaces. The first one is based on the use of (approximate) singular vectors. The second options utilizes the GKL vectors. Our tests show that generally a larger number of GKL vectors is needed to obtain comparable retrieval accuracy. However, the case of singular vectors has a slightly higher computational cost.

Note that construction of search subspaces is not restricted only to the two techniques considered in the present work. Due to the established link to a Rayleigh-Ritz procedure, other approaches for generating search subspaces can be investigated within this framework in future research.

Our experiments demonstrate a substantial efficiency improvement over the state-of-the-art updating algorithms [27], with a similar or higher level of retrieval accuracy. Since the new approach scales linearly in p (in contrast to the cubic scaling exhibited by the existing methods [27]), the efficiency gap becomes especially evident as p increases. Because significantly large values of p are likely to be encountered in the context of large-scale datasets, this means that in the future, algorithms such as the ones proposed in this paper, may play a role in reducing the cost of standard SVD-based techniques employed in the related applications.

REFERENCES

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

[2] M. Berry, Z. Drmac, and E. R. Jessup, *Matrices, vector spaces, and information retrieval*, SIAM Review, 41 (1999), pp. 335–362.

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

[4] M. W. Berry, *SVDPACK: A Fortran-77 software library for the sparse singular value decomposition*, tech. rep., Knoxville, TN, USA, 1992.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning*, Information Science and Statistics, Springer, 2006.

[6] K. Blom and A. Ruhe, *A Krylov subspace method for information retrieval*, SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 566–582.

[7] J. Chen and Y. Saad, *Divide and conquer strategies for effective information retrieval*, in SIAM Data Mining Conf. 2009, C. Kamath, ed., 2009, pp. 449–460.

[8] ———, *Lanczos vectors versus singular vectors for effective dimension reduction*, IEEE Trans. on Knowledge and Data Engineering, 21 (2009), pp. 1091–1103.

[9] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman, *Indexing by latent semantic analysis*, J. Soc. Inf. Sci., 41 (1990), pp. 391–407.

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

[11] G. H. Golub and W. M. Kahan, *Calculating the singular values and pseudoinverse of a matrix*, SIAM J. Num. Anal. Ser. B, 2 (1965), pp. 205–224.

[12] G. H. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 3rd ed., 1996.

[13] D. K. Harman, *The 3rd text retrieval conference (trec-3), ed. (1995).* NIST Special Publication 500-225. http://trec.nist.gov.

[14] V. Hernández, J. E. Román, and A. Tomás, *A robust and efficient parallel SVD solver based on restarted Lanczos bidiagonalization*, Electron. Trans. Numer. Anal., 31 (2008), pp. 68–85.

[15] M. E. Hochstenbach, *A Jacobi-Davidson type SVD method*, SIAM Journal on Scientific Computing, 23 (2001), pp. 606–628.

[16] E. Kokiopoulou and Y. Saad, *Polynomial Filtering in Latent Semantic Indexing for Information Retrieval*, in ACM-SIGIR Conference on research and development in information retrieval, Sheffield, UK, July 25th-29th 2004.

[17] T. Kolda and D. O. Leary, *A semi-discrete matrix decomposition for latent semantic indexing in information retrieval*, ACM Transactions on Information Systems, 16 (1998), pp. 322–346.

[18] J. E. Mason and R. J. Spiteri, *A new adaptive folding-up algorithm for information retrieval*, 2007.

[19] G. W. O'Brien, *Information management tools for updating an SVD-encoded indexing scheme*, 1994. Masters Thesis, Department of Computer Science, University of Tennessee, Knoxville, TN.

[20] B. N. Parlett, *The Symmetric Eigenvalue Problem*, no. 20 in Classics in Applied Mathematics, SIAM, Philadelphia, 1998.

[21] Y. Saad, *Numerical Methods for Large Eigenvalue Problems- classics edition*, SIAM, Philadelpha, PA, 2011.

[22] G. Salton, *Automatic Text Processing*, Addison-Wesley, New York, 1989.

[23] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.

[24] H. Simon and H. Zha, *Low rank matrix approximation using the Lanczos bidiagonalization process*, SIAM J. Sci. Stat. Computing, 21 (2000), pp. 2257–2274.

[25] J. E. Tougas and R. J. Spiteri, *Updating the partial singular value decomposition in latent semantic indexing*, Comput. Stat. Data An., 52 (2008), pp. 174–183.

[26] D. Zeimpekis and E. Gallopoulos, *TMG: A MATLAB-based term-document matrix constructor for text collections*, tech. rep., Comp. Sci. dept, University of Patras, December 2003.

[27] H. Zha and H. Simon, *On updating problems in latent semantic indexing*, SIAM Journal on Scientific Computing, 21 (1999), pp. 782–791.

[28] H. Zha and H. Zhang, *Matrices with low-rank-plus-shift structure: partial SVD and latent semantic indexing*, SIAM. J. Matrix Anal. Appl., 21 (2000), pp. 522–536.