PROC. N.S. INST. SCI. (2006)
Volume 43, Part 2, pp. 211-218

**2005 NSIS Graduate Student
Special Prize for High Merit**

A COMPARISON OF METHODS FOR MODIFYING THE PARTIAL SINGULAR VALUE DECOMPOSITION IN LATENT SEMANTIC INDEXING

JANE E. TOUGAS
Faculty of Computer Science
Dalhousie University
6050 University Avenue
Halifax, Nova Scotia B3H 1W5

The tremendous size of the Internet and modern databases has made efficient searching and information retrieval (IR) important. Latent semantic indexing (LSI) is an IR method that represents a dataset as a term-document matrix. LSI uses a matrix factorization method known as the partial singular value decomposition (PSVD). Calculating the PSVD of a large term-document matrix is computationally expensive. In a rapidly expanding environment, a term-document matrix is altered often as new documents and terms are added. Recomputing the PSVD of the term-document matrix each time these slight alterations occur can be prohibitively expensive.

Folding-in is one method of adding new documents or terms to an LSI database; updating the PSVD of the existing LSI database is another. The folding-in method is computationally inexpensive, but may cause deterioration in the accuracy of the PSVD. The PSVD-updating method is computationally more expensive than the folding-in method, but better maintains the accuracy of the PSVD. Folding-up is a new method that combines folding-in and PSVD-updating. Folding-up is faster than either recomputing the PSVD or PSVD-updating, but avoids the degradation in the PSVD that can occur when the folding-in method is used on its own.

La taille incroyable d'Internet et des bases de données modernes a fait en sorte que la recherche efficace d'informations est maintenant importante. L'indexation par sémantique latente (ISL) est une méthode de recherche d'informations qui représente un jeu de données comme une matrice document-terme. L'ISL comprend l'utilisation d'une méthode de factorisation matricielle connue sous le nom de décomposition partielle en valeurs singulières (DPVS). Le calcul de la DPVS d'une grande matrice document-terme est coûteux sur le plan des calculs. Dans un environnement en expansion rapide, une matrice document-terme est souvent modifiée à mesure que de nouveaux documents et termes sont ajoutés. Le recalcul de la DPVS de la matrice document-terme chaque fois qu'une légère modification est apportée peut devenir très coûteux.

L'intégration (folding-in) est une méthode pour ajouter de nouveaux documents ou termes dans une base de donnée ISL, et la mise à jour de la DPVS de la base de données ISL existante en est une autre. La méthode d'intégration est peu coûteuse sur le plan des calculs, mais elle peut entraîner une perte d'exactitude de la DPVS. La méthode de mise à jour de la DPVS est plus coûteuse sur le plan des calculs, mais elle permet de mieux préserver l'exactitude de la DPVS. La méthode d'intégration et de mise à jour (folding-up) est une nouvelle méthode qui combine l'intégration et la mise à jour de la DPVS. Cette méthode est plus rapide que le recalcul ou la mise à jour de la DPVS, mais elle permet d'éviter la perte d'exactitude de la DPVS qui peut survenir quand seule la méthode d'intégration est utilisée.

E-mail: tougas@cs.dal.ca

INTRODUCTION

The expansion of both the Internet and modern databases has sparked an increased interest in methods for the efficient retrieval of information. Latent semantic indexing (LSI) is an information retrieval (IR) method that relies heavily on methods from numerical linear algebra (NLA). LSI thus combines two very different, although equally interesting, fields of study.

The efficient retrieval of textual information is hampered by the fact that many words have more than one meaning (they are *polysemous*). When a polysemous word is used in a search query, irrelevant documents about the word's other meaning(s) may be retrieved. A further complication arises from the fact that many words have similar meanings (they are *synonymous*). When a word that has a synonym is used in a search query, relevant documents containing the synonym, but not the specific word used in the query, may be overlooked. LSI uses a matrix factorization method known as the partial singular value decomposition (PSVD) in an attempt to reduce the retrieval problems caused by polysemy and synonymy.

LSI uses a mathematical approach to examine a document collection as a whole and determine which documents contain many of the same words. The more words documents have in common, the more closely related they are considered to be. When a search query is made, documents containing the words used in the search are returned, but so are those that are closely related to these documents. This allows the retrieval of documents that do not contain all or even any of the words given in the search query (Berry & Browne 1999, Yu et al. 2002).

The mathematical approach LSI uses is known as the vector-space model (Berry & Browne, 1999, Berry et al. 1999, Salton & McGill 1983). This approach creates a *term-document matrix* in which there is a vector for each document, with as many entries as there are semantically significant terms in the documents. The term-document matrix is essentially a table of numbers with, as already noted, a column (vector) for each document, and a row for each term. This matrix is of size t x d, where t denotes the number of terms, and d the number of documents. Refer to this matrix as **A**, and to each entry as a_{ij} , where $1 \leq i \leq t$ and $1 \leq j \leq d$. Each entry indicates the presence or absence of a particular term in a particular document; this may be a weighted frequency that reflects the importance of each term in each document. Search queries are also represented as *t*-dimensional vectors. The query vectors are projected into the term-document matrix using the PSVD. The vectors of documents and queries with many terms in common will be close together in the vector space, whereas those with relatively few terms in common will be far apart. The cosine of the angle between two vectors is commonly used as a measure of the similarity of the vectors. The cosine of the angle between two vectors that are close together will be large, whereas the cosine of the angle between two vectors that are far apart will be smaller.

Problem and motivation

Even using the most advanced NLA methods, calculating the PSVD of a matrix is an extremely expensive computation. In LSI, most of the computation time is spent in calculating the PSVD (Berry et al. 1999, Berry et al. 1995). In a dynamic environment, such as the Internet, the term-document matrix is changed often as new documents and terms are added. Recalculating the PSVD of the matrix each time such changes occur can be prohibitively expensive. Traditionally, LSI uses a process known as folding-in to modify the PSVD. Unfortunately, folding-in results in a trade-off between efficiency and accuracy; although this method is much faster than recomputing the PSVD, it is also much less accurate. Updating the PSVD is a more recent approach (Zha & Simon 1999) that offers a compromise. Updating the PSVD is a slower process than folding-in (although it is much faster than recomputing), but unlike folding-in, it results in little or no loss of accuracy.

The purpose of this paper is to illustrate that updating the PSVD is more accurate than folding-in, and also to introduce a new method, *folding-up*, that is a combination of folding-in and updating the PSVD. Folding-up is an attractive option because it offers a significant improvement in computation time when compared to either recomputing or updating the PSVD, and yet it results in little or no loss of accuracy.

Background

In order to understand the PSVD, it is helpful to understand the singular value decomposition (SVD). The SVD is a matrix factorization that can be used to capture the salient features of a matrix. Given a matrix A of size t x d, its SVD is written as $A = U\Sigma V^T$, where U is a size tx t matrix, Σ is size t x d, and V is size d x d. U and V contain the left and right *singular vectors* of A respectively. When A is a term-document matrix, U represents the term vectors and V represents the document vectors. The matrix Σ has non-zero entries only on the diagonal. These diagonal entries, denoted σ_j for j = 1, 2, ..., min (m,n) and arranged in non-increasing order, are known as the singular values of matrix A. The number of non-zero singular values of a matrix is known as its *rank*, r (see Fig 1).

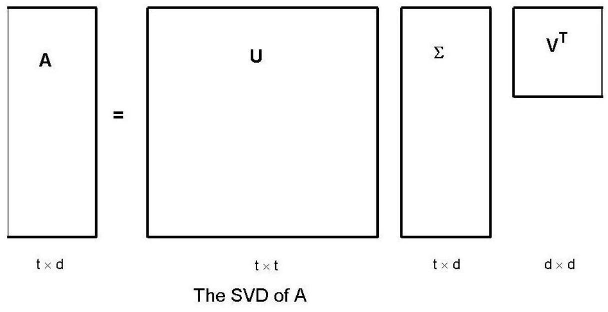

Fig 1 The singular value decomposition (SVD) of **A**.

An alternative way to represent the SVD is as the sum of r rank-one matrices $\mathbf{A} = \Sigma_{j=1}^{r}$ _j $\mathbf{u}_j\mathbf{v}_j^T$, where $\mathbf{u}_j$ and $\mathbf{v}_j$ are the *jth* columns of matrices **U** and **V**, respectively. This representation of the SVD allows the formation of lower-rank approximations of **A**. Let matrices $\mathbf{U}_k$ and $\mathbf{V}_k$ be the first *k* columns of **U** and **V** respectively, and let matrix Σ_k be the leading submatrix of Σ with *k* rows and *k* columns. $\mathbf{A}_k = \mathbf{U}_k\Sigma_k\mathbf{V}_k^T$ is a lower-rank approximation of **A**, where $0 \leq \mathbf{k} < r$. This is the partial SVD (PSVD) of **A**. This approximation can be used to reduce the dimension of the term-document matrix, while eliciting the underlying structure of the data. In LSI, the effect of this dimensional reduction on the data is a muting of the noise caused by synonymy and an enhancing of the latent patterns that indicate semantically similar terms. This means that $\mathbf{A}_k$ can actually be a better representation of the data than the original term-document matrix. The number of terms to keep in the reduced term-document matrix is still open to debate, but experiments indicate that values of *k* between 100 and 300 give the best results (Berry et al. 1995). This tremendous dimensional reduction, given the potentially huge size of the term-document matrix, demonstrates the power of the SVD as a method of data compression (see Fig 2).

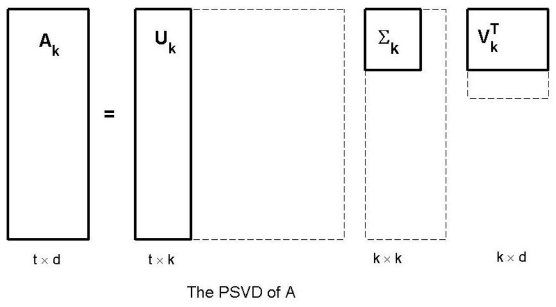

Fig 2 The partial singular value decomposition (PSVD) of **A**.

METHODS

Let $\mathbf{A} = \mathbf{U}_k\Sigma_k\mathbf{V}_k^T$ be the PSVD of the term-document matrix of size *t x d*, where *t* is the number of terms, *d* is the number of documents, and k is the number of dimensions used in the PSVD. Let **D** be the *t x p* term-document matrix containing the document vectors to be appended to **A**, where *p* is the number of new documents.

The folding-in process projects **D** into the *k*-dimensional space with a matrix multiplication: $\mathbf{D}_k = \mathbf{D}^T\mathbf{U}_k\ \Sigma_k^{-1}$ The *p x k* projection $\mathbf{D}_k$ is then folded-in to the existing PSVD of **A** by appending it to the bottom of $\mathbf{V}_k$. This gives the modified matrix$\hat{\mathbf{V}}_k$ of size (d + p) x *k*. It is important to note that $\mathbf{U}_k$ and Σ_k are not modified in any way with this method. As more and more documents are folded-in, the term-document representation of the document collection, $\mathbf{U}_k\Sigma_k\hat{\mathbf{V}}_k$, becomes less and less accurate, because $\mathbf{U}_k$ and Σ_k are not being modified to reflect the addition of the new documents. Folding-in terms follows a similar process.

Updating the PSVD when the term-document matrix changes is a more complicated process than folding-in. Unlike folding-in, which modifies only the matrix Vk, updating modifies each of the matrices $\mathbf{U}_k$, Σ_k, and $\mathbf{V}_k$. The end result (in the absence of rounding off errors) is the exact PSVD of the modified term-document matrix, without the expense of recomputing it from scratch. Although this is computationally more expensive than folding-in, it provides a more accurate representation of the modified document collection. Updating terms rather than documents is done in a similar manner.

Folding-up is new hybrid method that uses a combination of folding-in and updating to modify the PSVD of the term-document matrix when documents (or terms) are added. The idea is to begin by folding-in documents, but then discard the changes made by folding-in, and updating before the accuracy of the representation degrades significantly. The process has the merit of saving the document vectors that are being folded-in between updates; it repays this cost with a saving in computation time, coupled with the precision advantages of updating. If no updates have been made, the current term-document matrix is the initial matrix, otherwise it is the last updated term-document matrix. Once the number of documents that have been folded-in reaches a pre-selected percentage of the current term-document matrix, the vectors that have been appended to Vk during folding-in are discarded. The PSVD is then updated to reflect the addition of all of the document vectors that have been folded-in since the last update. These document vectors are then discarded. The process then continues, folding-in until the next update.

Experiments.

The experiments are run using *Matlab* Release 13 on an Ultra3 SunFire V880 (Solaris 8 operating system). The MEDLINE text collection (Cornell SMART System [ftp://cs.cornell.edu/pub/smart]) containing 1033 documents and 30 queries is used. Removing semantically insignificant terms gives a term-document matrix A_{med} of size 5735 _ 1033. The measure of similarity is the cosine of the angle between query and document vectors. For each example, the term-document matrix is incrementally updated with document vectors until its column space has approximately doubled. The average precision for the methods used in each example are compared. In each case, the average precision for each of the queries at 11 standard recall levels (0%, 10%, ···, 100%) is averaged to produce the overall average precision at each increment of each experiment. For each method used, the average precision is plotted at each of these increments, starting with the initial term-document matrix. All PSVDs are computed using the *Matlab* function svds, with k = 125 for the MEDLINE collection, where k is the number of singular values and corresponding left and right singular vectors computed. The updating method is based on a method introduced by Zha and Simon (1999). For the sake of brevity, the experiments described use only the document updating method. Note that similar results are produced using the term updating method.

RESULTS

Example 1

A$_{med}$, size 5735 _ 1033, is partitioned such that the first 533 columns are used as the initial term-document matrix, and the remaining columns are added incrementally, in groups of size 10. The average precision is compared for four methods at each increment: recomputing the PSVD, folding-in, updating, and folding-up. For the folding-up method in this example, updates occur when the number of documents folded-in reaches approximately 8% of the size of the initial matrix for the first update, and of the updated matrix thereafter. As expected, Fig 3 shows that the average precision for

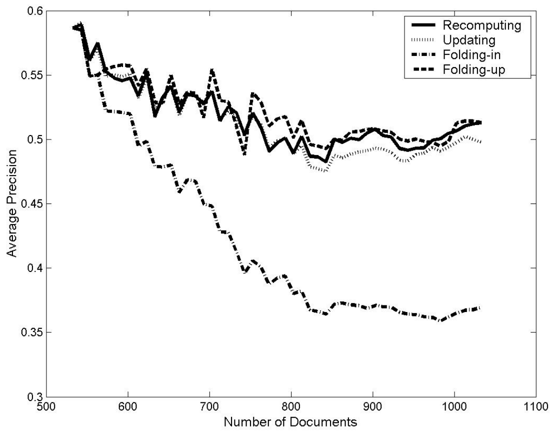

Fig 3 Comparison of the average precision of four methods for the MEDLINE collection, with 500 documents added in 50 groups of 10.

folding-in deteriorates rapidly. The average precision for updating does not begin to deteriorate until the initial matrix is more than one and a half times its original size, and the increments are less than 1.25% of the size of the matrix. Although the deterioration is slight, it does indicate that doing many updates that are very small relative to the size of the matrix may eventually have a negative affect on the average precision. However, the savings in computation time compared to recomputing, shown in Table 1, may more than compensate for this small deficiency; in this case, updating is more than 100 times faster than recomputing. Fig 3 shows that in this example folding-up actually outperforms the other methods for much of the graph, even though it is faster than either recomputing or just updating. See Table 1 for a comparison of CPU times.

Table 1 Comparison of total CPU times (seconds) for the MEDLINE collection with 500 documents added in groups of 10 and in groups of 25.

Method	**CPU time**	**CPU time**
	Increments of 10	**Increments of 25**
Recomputing	5001.60	2045.80
Updating	43.07	22.33
Folding-in	1.35	0.75
Folding-up	15.76	13.14

Example 2

A$_{med}$, size 5735 _ 1033, is partitioned such that the first 533 columns are used as the initial term-document matrix, and the remaining columns are added incrementally, in groups of size 25. The average precision is compared for four methods at each increment: recomputing the PSVD, folding-in, updating, and folding-up. For the folding-up method in this example, updates

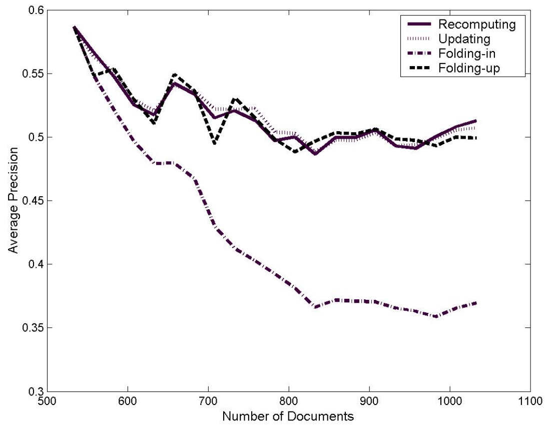

Fig 4 Comparison of the average precision of four methods for the MEDLINE collection, with 500 documents added in 20 groups of 25.

occur when the number of documents folded-in reaches approximately 14% of the size of the initial matrix for the first update, and of the updated matrix thereafter. As in Fig 3, Fig 4 shows that the average precision for folding-in deteriorates rapidly. The average precision for updating does not deteriorate, and is at times even slightly better than that for recomputing the PSVD. These results suggest that updating in larger increments, relative to the size of the matrix, can give optimal average precision. Folding-up again outperforms the other methods for some parts of the graph, while taking less computation time than either recomputing the PSVD or simply updating it. Table 1 gives a comparison of the CPU times for the methods.

CONCLUSIONS

LSI relies heavily on the PSVD of the term-document matrix representation of a document collection. Calculating the PSVD of large term document matrices is computationally expensive, therefore when documents (or terms) are added to an existing dataset, it is beneficial to update the existing PSVD to reflect these changes. Examples 1 and 2 illustrate that updating the PSVD of the term-document matrix each time changes are made to the document collection is not only much faster than recomputing the PSVD, but also gives better average precision than the traditional method of folding-in documents. Folding-up, a new method that is a combination of folding-in and updating, gives better average precision that folding-in, with less computation time than the updating method.

Acknowledgements. The work for this paper by JET was supported by NSERC Canada and the Killam Trust. The research for this paper was done in collaboration with Henry Stern, and supervised by Dr. Raymond J. Spiteri.

REFERENCES

Berry MW, Browne M (1999) Understanding search engines: mathematical modeling and text retrieval. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA

Berry MW, Drmac Z, Jessup ER (1999) Matrices, vector spaces, and information retrieval. SIAM Rev 41(2):335–362(electronic)

Berry MW, Dumais ST, O'Brien GW (1995) Using linear algebra for intelligent information retrieval. SIAM Rev 37(4):573–595

Salton G, McGill MJ (1983) Introduction to modern information retrieval. McGraw-Hill, New York, NY

Yu C, Cuadrado J, Ceglowski M, Payne JS (2002) Patterns in unstructured data: discovery, aggregation, and visualization. Presentation to the Andrew W. Mellon **Foundation, New York, NY**

Zha H, Simon HD (1999) On updating problems in latent semantic indexing. SIAM J Sci Comput 21(2):782–791