Hindawi
Scientific Programming
Volume 2019, Article ID 1095643, 12 pages
https://doi.org/10.1155/2019/1095643

Research Article

Alternate Low-Rank Matrix Approximation in Latent Semantic Analysis

Fahrettin Horasan, Hasan Erbay, Fatih Varçın, and Emre Deniz

Computer Engineering Department, Engineering Faculty, Kırıkkale University, Yahşihan, 71450 Kırıkkale, Turkey

Correspondence should be addressed to Fatih Varçın; fatihvarcin@kku.edu.tr

Received 18 June 2018; Accepted 29 November 2018; Published 3 February 2019

Academic Editor: Danilo Pianini

The latent semantic analysis (LSA) is a mathematical/statistical way of discovering hidden concepts between terms and documents or within a document collection (i.e., a large corpus of text). Each document of the corpus and terms are expressed as a vector with elements corresponding to these concepts to form a term-document matrix. Then, the LSA uses a low-rank approximation to the term-document matrix in order to remove irrelevant information, to extract more important relations, and to reduce the computational time. The irrelevant information is called as "noise" and does not have a noteworthy effect on the meaning of the document collection. This is an essential step in the LSA. The singular value decomposition (SVD) has been the main tool obtaining the low-rank approximation in the LSA. Since the document collection is dynamic (i.e., the term-document matrix is subject to repeated updates), we need to renew the approximation. This can be done via recomputing the SVD or updating the SVD. However, the computational time of recomputing or updating the SVD of the term-document matrix is very high when adding new terms and/or documents to preexisting document collection. Therefore, this issue opened the door of using other matrix decompositions for the LSA as ULV- and URV-based decompositions. This study shows that the truncated ULV decomposition (TULVD) is a good alternative to the SVD in the LSA modeling.

1. Introduction

The latent semantic analysis (LSA) is a mathematical/statistical method which is used for discovering the existing latent relationships between terms and documents or within a collection of documents (i.e., a large corpus of text) [1]. Although the LSA works especially well on textual data, it has been very popular in the academic community recently because of its wide variety of practices in the content information [2], sociological discourse analysis [3], image retrieval systems [4], human cognition, and human learning [5]. The LSA can be applied to any collection of documents that is cleaned from the syntactical and grammatical structure. If the collection of documents contains m terms and n documents, it is represented by using a matrix A of dimension $m \times n$ and called the term-document matrix.

The LSA uses a low-rank approximation to the term-document matrix in order to remove irrelevant information, to extract more important relations, and to reduce the computational time. The irrelevant information is called as "noise" and does not have a noteworthy effect on the meaning of the document collection [6].

The low-rank approximation of the term-document matrix A is, for a positive constant $k \ll \min(m, n)$, the matrix A_k that satisfies

$$\begin{aligned} \min \quad & \left\| A - A_k \right\|, \\ \text{s.t.} \quad & \operatorname{rank} A_k = k, \end{aligned} \tag{1}$$

where $\| \cdots \|$ represents either two-norm or *Frobenius-norm*. The existence of such a matrix follows from the singular value decomposition (SVD) of A. Moreover, with no doubt, the truncated singular value decomposition is the main tool for solving the minimization problem given by (1). However, in the LSA where document collections are dynamic over time, i.e., the term-document matrix is subject to repeated updates, the SVD becomes prohibitive due to the high computational expense. Thus, alternative decompositions have been proposed for these applications such as low-rank ULV/URV decompositions [7] and truncated ULV decomposition (TULVD) [8]. Recall that, the initial computing cost for the low-rank ULV/URV decompositions and the TULVD lower is than the SVD [7].

The manuscript demonstrates that the TULVD is a good substitute for the SVD in the LSA modeling.

The rest of the manuscript is organized as follows. In Section 2, we introduce some notations and cover critical background materials in numerical linear algebra. In Section 3, we give the main steps of our LSA modeling. Then in Section 4, we test our model using some commonly used test collections and present some simulation results. In Section 5, we comment on simulation results.

2. Notations and Background

2.1. Notations. Throughout the paper, uppercase letters such as A denote matrices. The $n \times n$ identity matrix is denoted by I_n. Moreover, the norm $\|\cdots\|$ denotes the spectral norm, and $\|\cdots\|_F$ denotes the Frobenius norm. The notation $\mathbb{R}^{m\times n}$ represents the set of $m \times n$ real matrices. An $m \times n$ dimensional matrix A is represented as $A = [a_{ij}]$ where a_{ij} is the entry of A at i row and j column with $1 \le i \le m$ and $1 \le j \le n$.

2.2. Orthogonal Matrix Decompositions

Definition 1 (the singular value decomposition). For a matrix $A \in \mathbb{R}^{m\times n}$ with $m \ge n$, the singular value decomposition (SVD) is

$$A = W \begin{pmatrix} \sum \\ 0 \end{pmatrix} Y^T, \tag{2}$$

where the left and right singular matrices W and Y are orthogonal matrices and where $\sum = \operatorname{diag}(\sigma_1, \dots, \sigma_n)$ is a diagonal matrix with the following order:

$$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n \ge 0. \tag{3}$$

The diagonal entries of $\sum$ are called the singular values of A.

For a given positive integer $k \ll n \ll m$, we block partition the SVD in (2) as

$$A = \begin{pmatrix} W_k & W_0 & W_\perp \end{pmatrix} \begin{pmatrix} \sum_k & 0 \\ 0 & \sum_0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Y_k & Y_0 \end{pmatrix}^T, \tag{4}$$

where $\sum_k = \operatorname{diag}(\sigma_1, \dots, \sigma_k)$ and $\sum_0 = \operatorname{diag}(\sigma_{k+1}, \dots, \sigma_n)$ are diagonal matrices containing the k largest and $n-k$ smallest singular values of A, respectively, (Figure 1). The matrix A_k defined by

$$A_k = W_k \sum_k Y_k^T, \tag{5}$$

is called rank k matrix approximation to A. For some tolerance, ϵ_M is proportional to the machine unit, and if the singular values satisfy

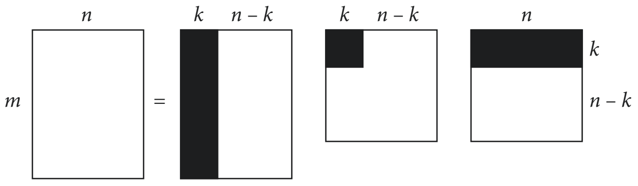


Figure 1: Low-rank decomposition.

$$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_k \gg \varepsilon_M \ge \sigma_{k+1} \ge \cdots \sigma_n \ge 0, \tag{6}$$

then the value k is called the numerical rank of the matrix A. However, we are aware that the determination of the numerical rank is a sensitive computation, especially when there is no well-defined gap between singular values [9, 10]. Moreover, in some situations, like the example in Section 5.4.1 of [10], the tolerance ϵ_M is chosen slightly bigger. The time complexity of obtaining A_k is $6mn^2 + 20n^3$ [7].

Even though the SVD provides accurate subspaces, as we mentioned above, it is not suitable for dynamic problems where data changes (i.e., update and/or downdate) due to high computational demand in both dense [11] or sparse matrices [12]. Herein, we consider ULV-based TULVD for approximating the matrix subspaces.

Definition 2 (the truncated ULV decomposition). For a matrix $A \in \mathbb{R}^{m\times n}$ with the numerical rank $k \ll n \ll m$, the TULVD is

$$A = ULV^T + E, \tag{7}$$

where $L \in \mathbb{R}^{k\times k}$ is nonsingular lower triangular matrix, $U_1 \in \mathbb{R}^{m\times k}$ and $V \in \mathbb{R}^{n\times k}$ are left orthogonal matrices (i.e., $U^TU = V^TV = I_k$), and $E \in \mathbb{R}^{m\times n}$ is an error matrix.

The matrices L and E satisfy

$$\begin{aligned} \min \quad & \|L^{-1}\|_F, \\ \text{s.t.} \quad & \|E_F\| < \epsilon_M, \\ & U^T E = 0. \end{aligned} \tag{8}$$

To meet these conditions, instead of minimizing $\|L^{-1}\|_F$, keep it as small as feasible and always enforce the constraint on $\|E\|_F$ thereby biasing the algorithm to favor a small approximation error over ideal conditioning of L.

The TULVD provides both rank and good approximate subspaces of the matrix [8, 13] and differs from the ULV decompositions in [14, 15] in two significant respects:

(1) The matrix E is not stored, instead, L_F^{-1} is maintained. However, by equations in (10) and (11), we are able to compute the projections Ev or E^Tu using A and U; in addition, computational tools for computing them are provided in Section 2 of [8].

(2) A is either sparse or structured, the matrix-vector multiplications Av or A^Tu require less than $O(mn)$. For example, when A is sparse, the computational complexity of the matrix-vector product is $O(m + n)$.

Proposition 1. *Let $A = ULV^T + E$ be a TULVD of the matrix $A \in \mathbb{R}^{m\times n}$ with rank k. Then*

$$E = PA, \quad P = I_m - UU^{+}, \tag{9}$$

where U^{+} is the pseudoinverse of U.

Proof. See [8].

Then, it follows that

$$Ev = (I_m - UU^{+})Av, \tag{10}$$

$$E^T u = A^T (I_m - UU^{+})u. \tag{11}$$

The computation of TULVD of the matrix A requires $12mnk + 3(m+n)k + 4n^2k + (k+1)W(A)$ operations where $W(A)$ represents the average work to estimate the principal singular triplets of A and takes $O(mn)$ operations [7]. □

3. LSA Modeling

The LSA relies on some existing latent structure in word usage in the corpus. It uses statistically derived conceptual indices instead of individual words for retrieval. Thus, it overcomes the problems of synonymy and polysemy in lexical matching retrieval methods [16]. Note that the LSA is an unsupervised learning method.

The main steps of the LSA modeling is outlined in Algorithm 1. The LSA modeling algorithm inputs the corpus of at least two or more monolingual textual documents. The documents may be of different types such as medical, educational, computer science, and social science. Moreover, the corpus may be in any language.

In the LSA modeling, a vector space, so-called "semantic space," representation of the document collection is typically computed and then the inner product or cosine between the user query vector and/or document vectors is used as a measure of similarity between the documents. We note that the similarity estimates derived by the LSA are not simple contiguity frequencies, co-occurrence counts, or correlations in usage but depend on a powerful mathematical/statistical analysis that is capable of correctly inferring much deeper relations [17].

In the following, the main steps of LSA modeling in Algorithm 1 are explained.

In obtaining the term-document matrix step, first, each document in the corpus is cleaned from the syntactical and grammatical structure in order to improve the LSA performance both effectively and efficiently. In the term-document matrix, each row stands for a unique word and each column stands for a document in the corpus. Each entry expresses both the word's importance in the particular document and the degree to which the word carries information in the corpus in general. Mathematically, the value of the entry a_{ij} at the ith row and jth column of the document-term matrix A is given by

$$a_{ij} = L(i,j) \times G(i), \tag{12}$$

where $L(i,j)$ represents the local weight of the word i in the document j and $G(i)$ represents the global weight of the word i. There are different local and general weighting methods defined in the literature; however, in this study, term frequency (tf) for local weighting and inverse document frequency (idf) for overall weighting methods are used to calculate the element $A[i,j]$ of the term-document matrix A. Other weighting schemes [18, 19] can be applied to increase/decrease the importance of terms within and/or among documents. Here we have to note that equation (12) does not take the word order into account.

The main step in LSA modeling is to obtain the low-rank approximation of the term-document matrix A. It is computed via both the SVD and the truncated ULV decomposition. In the former case, the rank k low-dimensional approximation is the matrix A_k given in equation (5), whereas in the latter case, it is derived from equation (7). To be more precise, the singular values L are close to the k-largest singular values of A, while the columns of matrices U and V are the corresponding approximate left and right singular vectors of A, respectively [8]. Recall that the LSA attempts to retrieve a small number of concepts are important for representing the corpus. The matrix L indicates these concepts as good as $\sum_k$, especially when L is diagonally dominant [20]. On the contrary, in the LSA applications, the number of important concepts is much smaller than both the number of terms and documents, i.e., $k \ll \min(m,n)$.

Note that in reduced k-dimensional semantic space, the rows of the matrix $W_k\sum_k$ and UL represent the terms, but the columns of the matrix $\sum_k Y_k^T$ and LV^T represent the documents in the corpus.

3.1. Query. A query consists of words and is considered as a document and represented in vector space. In other words, the query composed of the words entered by the user is translated to an $m \times 1$ dimensional vector q by using the same weighting process used to construct the term-document matrix. Then, the query vector q is represented in the SVD-based vector space given by equation (5) as

$$\hat{q} = q^T W_k \sum_k, \tag{13}$$

and the TULVD-based vector space given by equation (7) as

$$\hat{\hat{q}} = q^T UL. \tag{14}$$

By representing the query vector in the corresponding vector space, all document vectors existing in the vector space can be compared with the query vector and sorted by the similarity rank.

The determination of the exact similarity measurement method is very important in terms of classification of documents and performance of information retrieval [21]. In literature, different similarity measures are defined such as Euclidean distance, cosine similarity, Jaccard coefficient, Pearson correlation coefficient, and averaged Kullback–Leibler divergence [22]. In this study, we prefer cosine similarity.

```
% input:
  % monolingual textual corpus
% output:
  % graph of terms and documents
% read the corpus, parse it, and execute the morphological step,
% weight the terms and obtain the term-document matrix
doc_file = ‘files/document.txt’;
stopword_file = ‘files/stopword_en.txt’;
A = doc_term_mat (doc_file, stopword_file, weight);
% apply the SVD decomposition
[W, Sigma, Y] = svd(A);
% obtain the components of A_k by the SVD
W_k = W (:, 1 : k);
Sigma_k = Sigma (1 : k, 1 : k);
Y_k = Y(:, 1 : k);
% apply the TULVD
[U, L, V] = TULV(A);
% obtain the components of A_k by the TULVD
U_k = U; L_k = L; V_k = V;
% find term and document vectors in k-space
term_vec = U_k * L_k
doc_vec = L_k * transpose (V_k)
% represent the query q in k-space
query_vec = query (q, U_k, L_k)
% find semantic relationship in the corpus using cosine similarity
semantic_sim = cosine_sim (query_vec, doc_vec)
```

Algorithm 1: The LSA modeling.

4. Application

To test our LSA model, we make use of three commonly used information retrieval test collections. These collections are American Documentation Institute Reports (ADI), a collection of articles published in Time magazine (TIME), and a collection of medline articles (MED). Each of these test collections contain a set of short articles, a set of queries, and a list indicating which documents are relevant to which queries. The performance of the model is evaluated on the basis of this list.

For each collection, stopwords in the documents are cleaned and stemming is applied before the term-document matrix of the collection is created. Table 1 presents some statistics before and after the preprocessing along with the number of queries.

Thus, the term-document matrix A is of size 986×82 for the ADI, 14774×424 for the TIME, and 9477×1033 for the MED, and its entries are obtained by using equation (12). The local weight $L(i, j)$ of the ith word in the jth document is obtained by tf, and the overall weighting $G(i)$ of the ith word is by idf, that is,

$$L(i, j) = \mathrm{tf}(i, j), \tag{15}$$

where $\mathrm{tf}(i, j)$ is the frequency of the word i within document j; on the contrary,

$$G(i) = \log\left(\frac{n}{\mathrm{df}(i)}\right), \tag{16}$$

where $\mathrm{df}(i)$ is the frequency of documents in which the word i appears at least once.

Table 1: Numbers of documents, queries, and terms in the test collections.

Collection	Number of documents	Number of terms	Number of terms before stemming	Number of queries
ADI	82	986	1308	35
TIME	424	14774	20853	83
MED	1033	9477	12931	30

The term-document matrix for the MED collection is given in Table 2. Note that as mentioned above, this matrix is a sparse matrix.

The distribution of terms in the data set using the SVD is given in Figure 2 and using the TULVD in Figure 3. On the contrary, the distribution of documents using the SVD is given in Figure 4 and using the TULVD in Figure 5.

When the distributions of terms and documents for both algorithms are visually examined, it is seen that those realized by the TULVD are distributed over a wider area. However, when analyzed as an angular perspective, it is observed that the distributions given for both algorithms are not identical but show a similar distribution.

Now, we quantitatively compare the performance of the SVD-based information retrieval and the TULVD-based information retrieval. The standard metrics are “recall” and “precision.” Before we remind their mathematical definitions, we define some variables. We let A be the set of documents returned as a result of the query information extraction and relevant to query. Moreover, let B represent all the documents accessed in the query result, and C represent all the documents related to the query in the corpus. Then, the recall is defined as

Table 2: Term-document matrix.

Document ID	Term ID	Weight
1	1	5.095
1	2	10.278
1	3	10.278
⋮	⋮	⋮
2	3	1.713
2	5	0.858
2	9	0.001
⋮	⋮	⋮
3	3	6.852
3	14	1.868
3	70	6.333
⋮	⋮	⋮
1033	9477	3.014

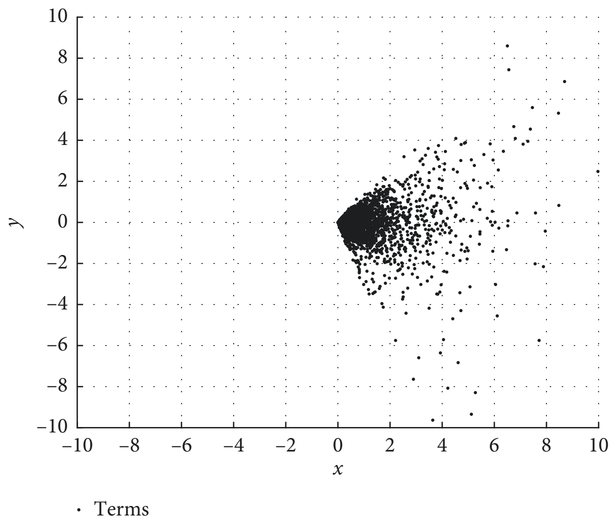


Figure 2: The distribution of terms of the MED collection using the SVD.

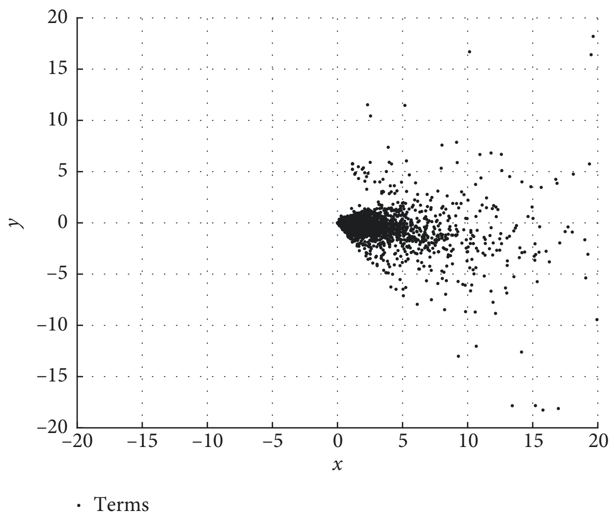


Figure 3: The distribution of terms of the MED collection using the TULVD.

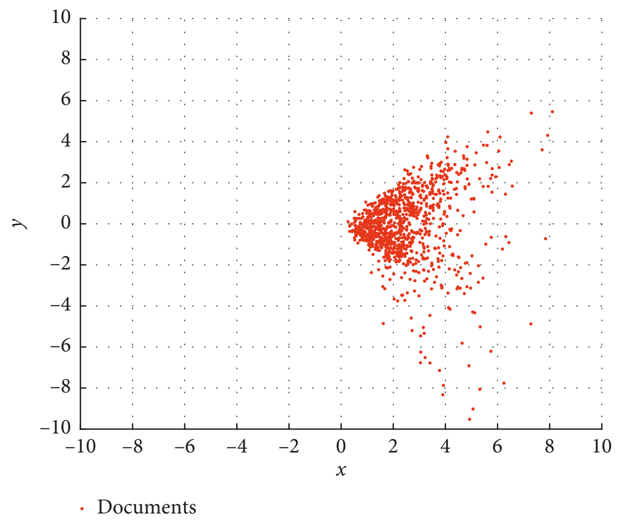


Figure 4: The distribution of documents of the MED collection using the SVD.

$$r = \frac{n(A)}{n(C)}, \quad (17)$$

and the precision is defined as

$$p = \frac{n(A)}{n(B)}. \quad (18)$$

A detailed explanation of these quantitative metrics can be found in [23–25].

Tables 3, 4, and 5 show the results of the SVD and TULVD methods for the ADI, the MED, and the TIME collections, respectively, according to different k values. In the tables, instead of taking all the returned documents after the query, we only use 10% and 50% to obtain quantitative metrics. Precision shows the average success of indexed documents for all queries in these slices. In addition, Min Cosine Similarity Value shows the average minimum cosine similarity value of the documents listed in the query result.

Table 3 shows the performance of the ADI collection according to the rank k value. For $k < 40$, the performance of both methods is poor. Indexing accuracy increases until k is between 50 and 60. However, when k is greater than 70, the indexing accuracy begins to decrease. Similarly, Table 4 shows the performance of the MED collection according to the k value, and it is seen that the performance of both methods is poor for $k < 10$, increases for k between 20 and 150 and decreases for $k > 300$. Finally, the performance analysis of the TIME collection is given in Table 5. The performance of both methods is poor for $k < 50$, increases when k is between 100 and 200, and decreases when k is greater than 200.

Tables 6, 7, and 8 show the success of the SVD and the TULVD methods applied to the ADI, the MED, and the TIME collections, respectively, according to the similarity threshold value. The results in the tables are obtained by taking the average of the achievement of the documents listed for all the

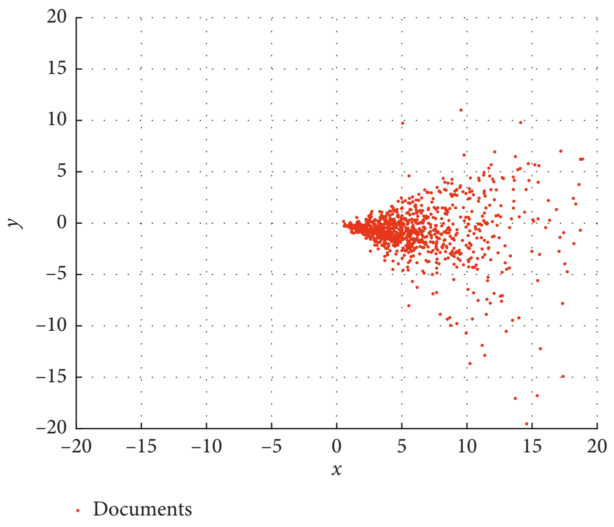


Figure 5: The distribution of documents of the MED collection using the TULVD decomposition.

Table 3: The accuracy of the returned documents by the SVD and the TULVD for the ADI collection.

k	SVD			TULVD		
	Min Cosine Similarity	Precision (10%)	Precision (50%)	Min Cosine Similarity	Precision (10%)	Precision (50%)
5	0.162	22.3	12.6	0.0884	21.3	14.1
10	0.119	31.1	18.6	0.0225	31.5	17.3
20	0.105	39.4	23.8	0.0101	34.7	19.9
30	0.103	55.3	29.0	0.0086	46.6	20.2
40	0.102	58.9	32.8	0.0071	69.3	32.1
50	0.101	73.5	33.1	0.0034	70.2	37.6
60	0.101	71.2	39.7	0.0008	65.3	37.9
70	0.100	69.2	39.3	0.0007	53.8	36.2
82	0.100	50.0	41.2	0.0003	49.3	29.1

Table 4: The accuracy of the returned documents by the SVD and the TULVD for the MED collection.

k	SVD			TULVD		
	Min Cosine Similarity	Precision (10%)	Precision (50%)	Min Cosine Similarity	Precision (10%)	Precision (50%)
5	0.618	22.2	12.6	0.786	29.8	11.7
10	0.289	34.2	15.2	0.547	53.0	17.2
20	0.069	73.1	19.3	0.148	66.5	18.8
50	0.035	77.9	19.4	0.091	77.5	24.2
100	0.025	85.2	34.4	0.035	83.2	45.6
150	0.022	83.3	27.1	0.021	85.8	46.0
300	0.015	66.4	18.1	0.01	77.2	38.3
600	0.007	56.2	16.5	0.006	52.0	16.0
900	0.005	51.6	15.3	0.003	47.7	15.1
1033	0.003	49.6	15.5	0.001	46.9	15.2

test questions in the collections. All of the returned documents are taken into account while calculating the results.

In these tables, performance values corresponding to a certain cosine threshold value are shown for both methods. As the cosine threshold increases, in general, the recall decreases, but the precision increases. Moreover, decreasing cosine threshold value increases the number of documents returned as a query result. However, the proportion of related documents in these documents increases.

The coordinates of the documents in the vector space obtained in the LSA process are used to list those that are

Table 5: The accuracy of the returned documents by the SVD and the TULVD for the TIME collection.

k	SVD			TULVD		
	Min Cosine Similarity	Precision (10%)	Precision (50%)	Min Cosine Similarity	Precision (10%)	Precision (50%)
5	0.6290	30.4	13.2	0.70	31.3	12.4
10	0.4840	49.1	17.0	0.56	44.4	15.5
20	0.2140	58.9	17.4	0.51	49.8	16.1
50	0.1001	59.7	18.5	0.17	59.6	34.7
100	0.1013	70.8	22.8	0.09	65.8	35.2
150	0.1009	72.6	25.1	0.06	73.4	38.3
200	0.1004	75.6	33.1	0.03	74.1	40.8
300	0.1026	73.5	40.9	0.02	63.1	44.8
400	0.0823	72.8	47.1	0.02	63.1	46.1
423	0.0812	68.8	45.1	0.01	56.7	32.4

Table 6: Average precision, recall, and total number of documents returned for various relative cosine threshold values for the ADI collection.

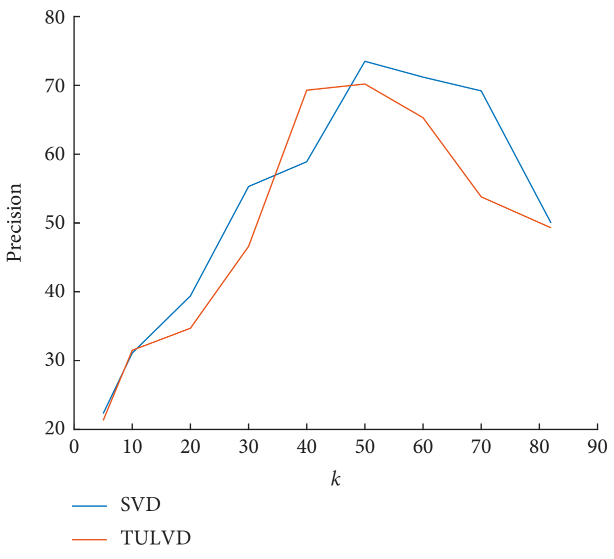
Cosine threshold	SVD			TULVD		
	Recall	Precision	Average number of documents returned	Recall	Precision	Number of documents returned
0.1	82.1	11.4	34.4	76.7	18.1	35.3
0.2	71.2	17.2	19.7	62.2	23.4	21.3
0.3	60.5	24.7	11.5	49.8	35.7	11.3
0.4	46.7	33.9	6.3	44.9	51.9	6.5
0.5	36.3	47.4	3.6	31.1	60.5	3.7
0.6	32.8	59.6	2.3	18.6	58.7	2.1
0.7	25.8	71.0	2.0	13.0	72.2	1.5
0.8	15.2	73.5	1.3	0.0	0.0	0.0
0.9	0.0	0.0	0.0	0.0	0.0	0.0

Table 7: Average precision, recall, and total number of documents returned for various relative cosine threshold values for the MED collection.

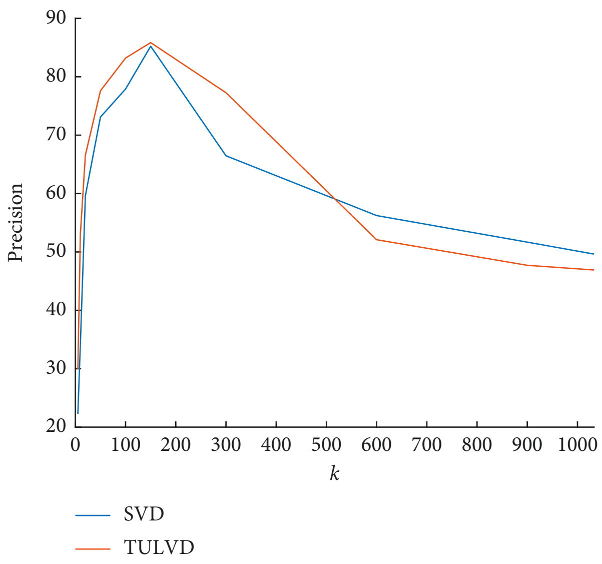
Cosine threshold	SVD			TULVD		
	Recall	Precision	Number of documents returned	Recall	Precision	Number of documents returned
0.1	97.4	21.3	135.7	97.9	17.2	165.8
0.2	94.1	43.1	64.1	94.5	38.0	73.4
0.3	89.7	59.4	41.1	87.1	53.4	44.7
0.4	83.3	70.4	30.2	81.9	66.3	32.4
0.5	66.3	76.8	21.7	68.4	75.1	23.2
0.6	51.7	86.8	15.1	51.8	84.2	15.2
0.7	30.0	91.7	18.4	33.0	90.2	9.3
0.8	18.6	97.1	4.5	20.9	90.7	5.5
0.9	7.9	100.0	1.5	6.1	100	1.3

Table 8: Average precision, recall, and total numbers of documents returned for various relative cosine threshold values for the TIME collection.

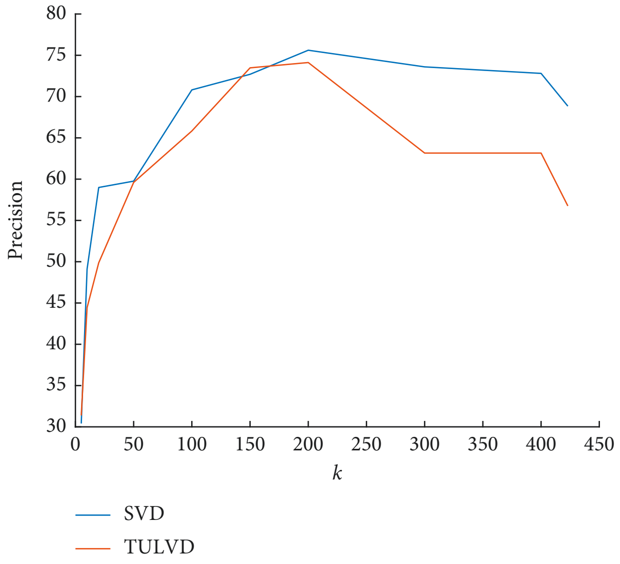
Cosine threshold	SVD			TULVD		
	Recall	Precision	Number of documents returned	Recall	Precision	Number of documents returned
0.1	88.6	12.4	58.4	82.5	22.5	37.5
0.2	82.4	17.2	46.6	71.8	34.4	20.2
0.3	73.5	26.9	25.6	59.1	40.1	12.8
0.4	68.3	37.7	16.3	49.9	51.7	8.1
0.5	63.2	49.8	10.8	40.0	56.8	5.5
0.6	54.1	63.2	7.1	32.8	66.3	4.8
0.7	43.8	72.3	4.8	32.7	67.5	3.0
0.8	40.6	76.6	4.0	27.6	71.6	2.5
0.9	0.0	0.0	0.0	0.0	0.0	0.0



(a)



(b)



(c)

Figure 6: Precision results for different k values using the SVD and the TULVD: (a) ADI; (b) MED; (c) TIME.

similar to the query clauses. Figure 6 shows the successful indexing of documents in vector space created using the SVD and the TULVD for the ADI, the MED, and the TIME collections. All documents returned for each query are listed in the decreasing order of similarity. The accuracy of these documents is calculated in percentiles. Precision decreases as percentile slice rate increases, but retrieval to related documents is increasing. As with other results, Figure 6 also reflects the average of the performance metrics of the documents listed for all of the queries in each collection.

Figure 7 compares the SVD and TULVD methods for the three collections used in the testing process with the average minimum similarity values of the documents listed at the end of the indexing process with different rank values. The results for the MED and the TIME collections seem to be similar. In the ADI collection, it is seen that the similarity change rate is almost the same, although it has different values according to the increasing rank values. This difference can be attributed to the fact that the number of documents in the MED and the TIME collections is much higher than the ADI collection.

Figure 7 shows the minimum cosine similarity value change for the ADI, the MED, and the TIME collections with respect to k. Note that as the number k increases for the three collections, the rate of change of the minimum

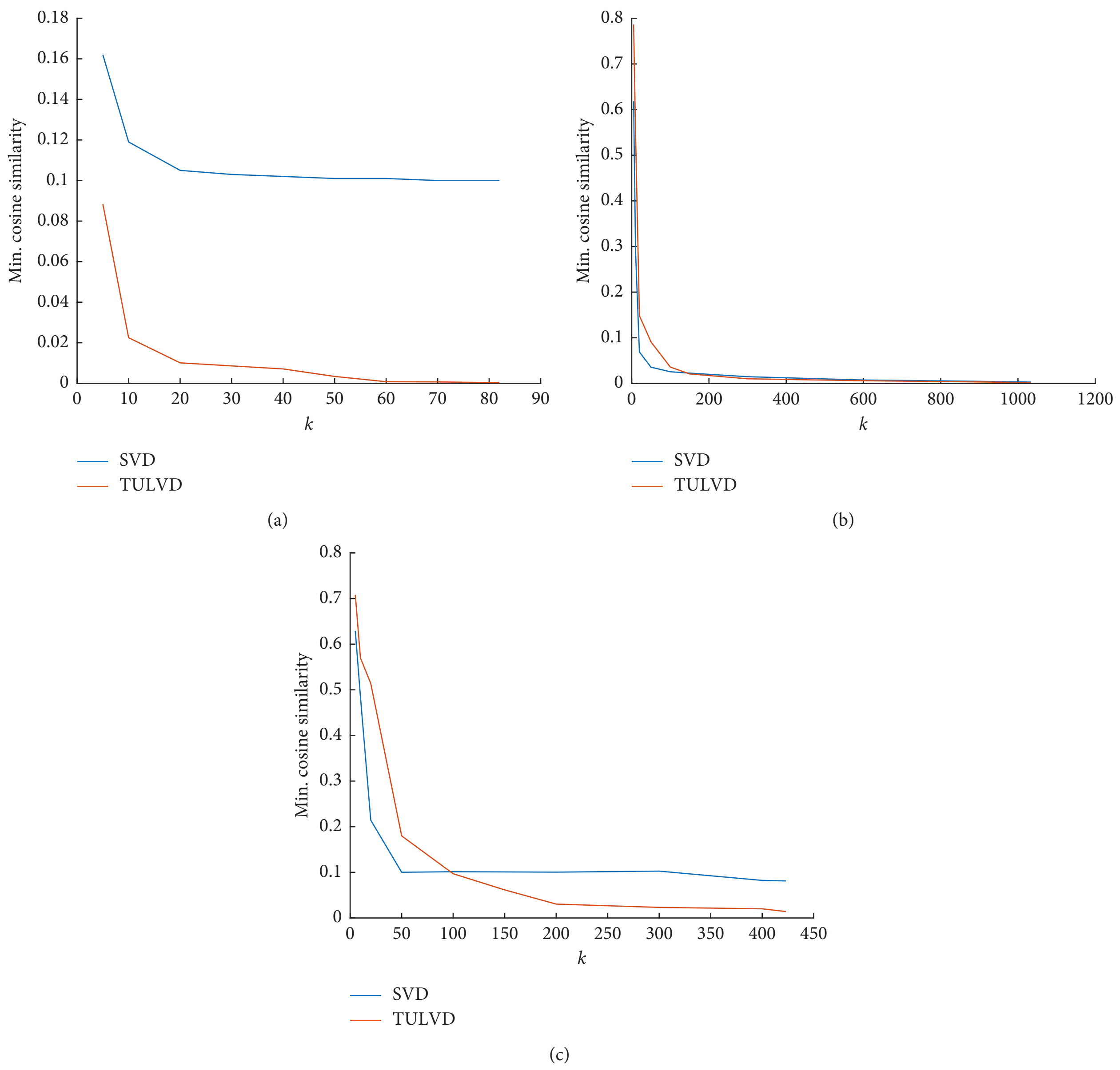


Figure 7: A plot of the minimum cosine similarity values for different k values using the SVD and the TULVD: (a) ADI; (b) MED; (c) TIME.

cosine similarity for successive steps decrease differently. Thus, it makes difficult to determine the similarity threshold used in document indexing and prevents successful indexing. For this reason, it is recommended to obtain values in which the rate of change of the value of k is high and the success of document indexing is good in order to increase the performance of the process and to retrieve the correct documents. In this case, k should be 50 for the ADI collection, 150 for the MED collection, and 150 for the TIME collection.

On the contrary, Figures 8, 9, and 10 illustrate the indexing success of the semantic vector space generated by both methods for the ADI, the MED, and the TIME collections, respectively, according to different k values.

5. Conclusion

According to the visual observation of the simulations presented in Figures 2, 3, 4, and 5 as well as the quantitative measurement of the recall and the precision given in Tables 3, 4, and 5, the TULVD can be a good substitute for the SVD to find the rank k approximation in the LSA modeling. To support our claim, in addition, when we examine Tables 6, 7, and 8, we observe that the TULVD-based LSA

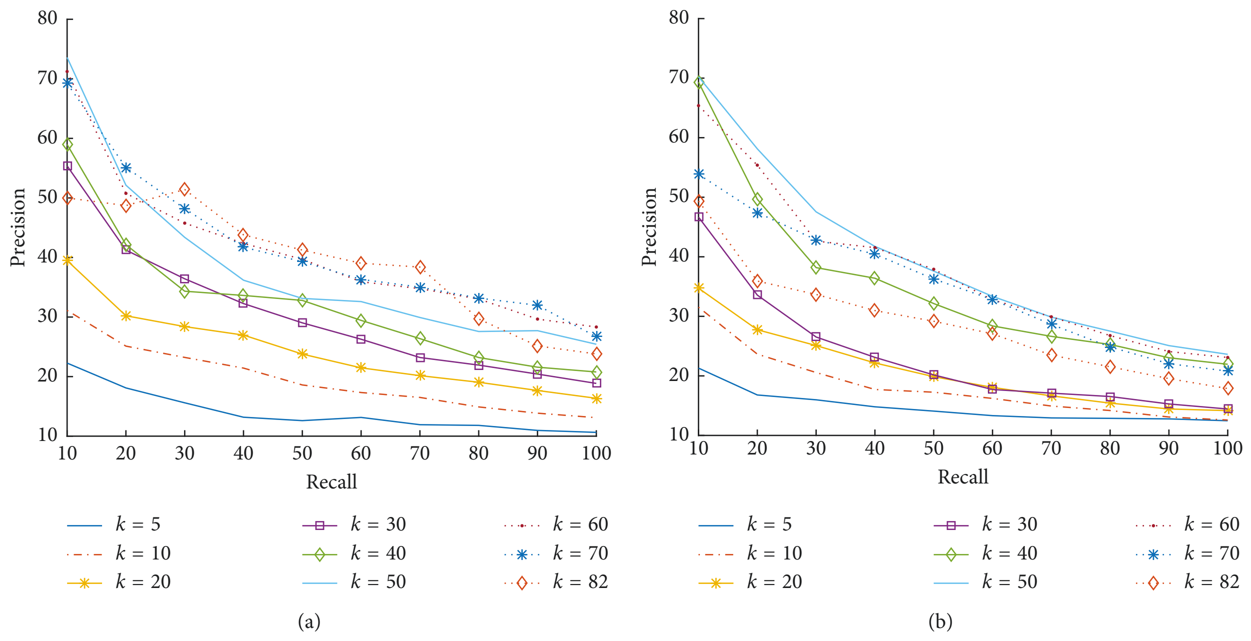


FIGURE 8: Precision vs. recall results for different *k* values using the SVD and the TULVD for the ADI collection: (a) SVD; (b) T-ULVD.

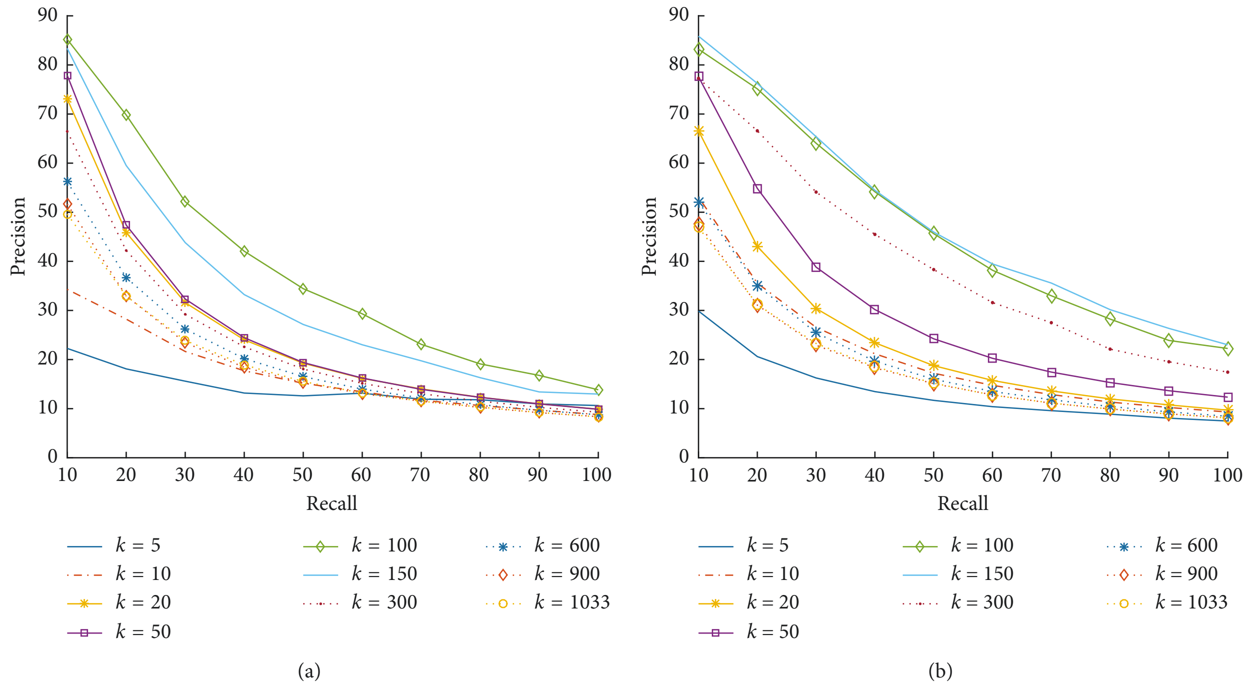
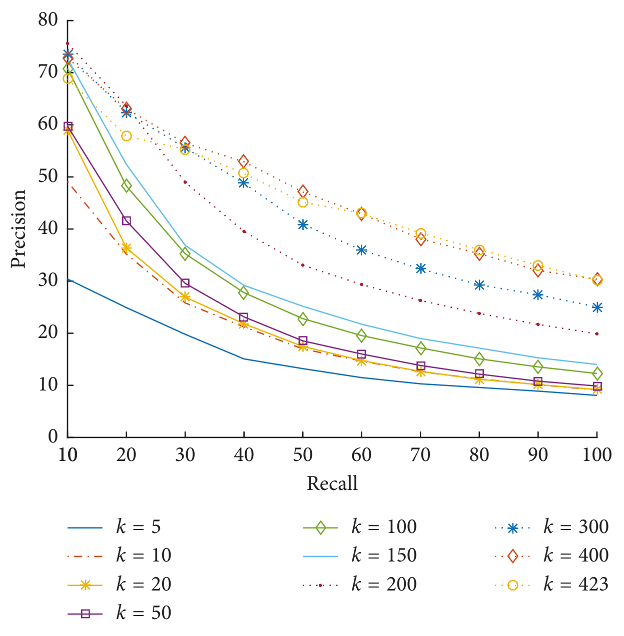


FIGURE 9: Precision vs. recall results for different *k* values using the SVD and the TULVD for the MED collection: (a) SVD; (b) T-ULVD.

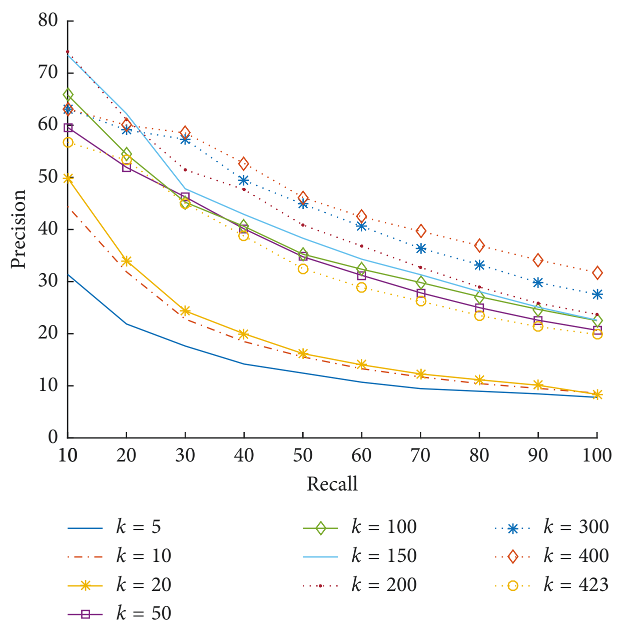
model produces results similar to the SVD-based LSA model for different cosine threshold values in these three collections. Moreover, Figures 8, 9, and 10 are examined, and the best *k* value for the SVD and TULVD application in the ADI collection is 50, in the MED collection 100, and in the TIME collection 200.

In result, the TULVD is as good as the SVD for retrieving the semantic structure of the textual document in the LSA modeling. The main advantage of the TULVD over the SVD is the efficient computation of the initial low-rank approximation as well as the efficient computation of the low-rank approximation when adding a new document and/or term to the existing LSA generated database, i.e., "updating".

As a result of our experience in this study, it is thought that TULVD can be used as an alternative method in many



(a)



(b)

Figure 10: Precision vs. recall results for different k values using the SVD and the TULVD for the TIME collection: (a) SVD; (b) T-ULVD.

areas of SVD used such as data compression, missing data completion, image processing, sound processing, noisy data cleaning, and especially signal processing. In addition, this study can be extended to cover fields such as text summarization, text similarity, keyword extraction, author detection, and text classification.

Data Availability

The ATI, TIME, and MEDLINE collections, which are well-known datasets, used to support the findings of this study have been obtained from the Glasgow Repository (http://ir.dcs.gla.ac.uk/resources/test_collections/).

Disclosure

Initial results of this study were presented in the 21st International Conference Mathematical Modeling and Analysis.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported with project 2016/150 by Kırıkkale University Scientific Research Projects (BAP).

References

[1] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[2] N. Seco, T. Veale, and J. Hayes, "An intrinsic information content metric for semantic similarity in wordnet," in *Proceedings of the 16th European Conference on Artificial Intelligence*, pp. 1089-1090, IOS Press, Valencia, Spain, August 2004.

[3] J. R. Ruiz, "Sociological discourse analysis: methods and logic," *Forum Qualitative Sozialforschung/Forum: Qualitative Social Research*, vol. 10, no. 2, pp. 1–22, 2009.

[4] M. Hanselman, M. Kirchner, B. Renard et al., "Concise representation of mass spectrometry images by probabilistic latent semantic analysis," *Analytical Chemistry*, vol. 80, no. 24, pp. 9649–9658, 2008.

[5] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[6] W. Song, J. Z. Liang, X. L. He, and P. Chen, "Taking advantage of improved resource allocating network and latent semantic feature selection approach for automated text categorization," *Applied Soft Computing*, vol. 21, pp. 210–220, 2014.

[7] R. D. Fierro and P. C. Hansen, "Low-rank revealing UTV decompositions," *Numerical Algorithms*, vol. 15, no. 1, pp. 37–55, 1997.

[8] J. L. Barlow and H. Erbay, "Modifiable low-rank approximation to a matrix," *Numerical Linear Algebra with Applications*, vol. 16, no. 10, pp. 833–860, 2009.

[9] D. Watkins, *Fundamentals of Matrix Computations*, John Wiley and Sons, Hoboken, NJ, USA, 2002.

[10] G. Golub and C. V. Loan, *Matrix Computations*, The John Hopkins Press, Baltimore, MD, USA, 2013.

[11] J. R. Bunch and C. P. Nielsen, "Updating the singular value decomposition," *Numerische Mathematik*, vol. 31, no. 2, pp. 111–129, 1978.

[12] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "The computational complexity of alternative updating approaches for an svd-encoded indexing scheme," in *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, pp. 39–44, San Francisco, CA, USA, February 1995.

[13] H. Erbay, J. L. Barlow, and Z. Zhang, "A modified Gram-Schmidt-based downdating technique for ULV

decompositions with applications to recursive TLS problems," *Computational Statistics & Data Analysis*, vol. 41, no. 1, pp. 195–209, 2002.

[14] G. W. Stewart, "An updating algorithm for subspace tracking," *IEEE Transactions on Signal Processing*, vol. 40, no. 6, pp. 1535–1541, 1992.

[15] J. L. Barlow, "Modification and maintenance of ULV decompositions," in *Applied Mathematics and Scientific Computing*, pp. 31–62, Springer, Berlin, Germany, 2002.

[16] J. E. Tougas and R. J. Spiteri, "Updating the partial singular value decomposition in latent semantic indexing," *Computational Statistics & Data Analysis*, vol. 52, no. 1, pp. 174–183, 2007.

[17] T. A. Letsche and M. W. Berry, "Large-scale information retrieval with latent semantic indexing," *Information sciences*, vol. 100, no. 1-4, pp. 105–137, 1997.

[18] S. T. Dumais, "Improving the retrieval of information from external sources," *Behavior Research Methods, Instruments, & Computers*, vol. 23, no. 2, pp. 229–236, 1991.

[19] M. G. Ozsoy, I. Cicekli, and F. N. Alpaslan, "Text summarization of Turkish texts using latent semantic analysis," in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 869–876, Association for Computational Linguistics, Beijing, China, August 2010.

[20] M. W. Berry and R. D. Fierro, "Low-rank orthogonal decompositions for information retrieval applications," *Numerical Linear Algebra with Applications*, vol. 3, no. 4, pp. 301–327, 1996.

[21] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, no. 4, pp. 573–595, 1995.

[22] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC'2008)*, pp. 49–56, Christchurch, New Zealand, April 2008.

[23] E. Jessup and J. Martin, "Taking a new look at the latent semantic analysis approach to information retrieval," in *Computational Information Retrieval*, pp. 121–144, SIAM, Philadelphia, PA, USA, 2001.

[24] B. Kang, D. Kim, and S. Lee, "Exploiting concept clusters for content-based information retrieval," *Information sciences*, vol. 170, no. 2, pp. 443–462, 2005.

[25] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation, Lecture Notes in Computer Science," in *Proceedings of Australasian Joint Conference on Artificial Intelligence*, pp. 1015–1021, Springer, Hobart, Australia, December 2006.