JKAU: Comp. IT., **Vol. 1** No. 2, pp: 69-103 (2012 A.D./ 1434 A.H.)
DOI: 10.4197 / Comp. 1-2.4

Lexical Noise Analysis and Removal in Intelligent Search Engines

Tareq Jaber
Faculty of Computing and Information Technology
King Abdulaziz University, Jeddah, KSA
tjaber1@kau.edu.sa

Abstract. In the field of intelligent information retrieval (IR), latent semantic indexing (LSI) is a popular technique used to retrieve information related more in meaning than in lexical matching. This technique overcomes the problems associated with synonymy and polysemy (common causes of inaccuracy in matching algorithms). A core component in the process is the use of the singular value decomposition (SVD) which acts as a mathematical model for the lexical noise in the term document matrix (TDM). This paper investigates various aspects of LSI from the viewpoint of noise modeling and removal in image processing. A discussion about and an investigation into, mathematical modeling for lexical noise in the TDM is presented. The work addresses a definition for *noise* in text processing and seeks to determine the best structure of the TDM. In other words, the structure of the TDM that would facilitate efficient searching within the LSI.

Keywords: latent semantic indexing, information retrieval, singular value decomposition, image processing, noise modeling and removal.

1 Introduction

The world is becoming increasingly digitized and there has been a massive expansion in the volume of data that is available online in a variety of forms. In addition, due to the so-called internet revolution, anyone can access millions of pages on the web. In 1998 researchers estimated that there were about 300 million web pages on the internet[1,2]

and now, ten years later, this estimate will have grown significantly. Having this information is useless unless there is some efficient way of retrieving appropriate or relevant information[3]. It can be argued that there is a clear need to provide new approaches that enhance the data retrieval processes and this is one of the driving forces behind the continued interest in information retrieval (IR) systems.

Traditional approaches to IR, such as those employed by the major search engines, are based on keyword matching. However, this method of IR presents major problems in terms of handling the increasingly large volumes of data, and in the accuracy of the results obtained. It has been suggested that individual terms and keywords are not adequate discriminators of the semantic content of the documents and queries[4].

Latent semantic indexing (LSI) is a well-known technique used in IR[3]. LSI has proved popular in IR as the technique can cope with the problems and inaccuracies associated with both synonymy and polysemy.

Synonymy is the situation where there are many ways to express a given concept, such as *cellphone*, *mobile* and *cellular*. Therefore the literal terms in a user′s query may not match those of relevant documents. Synonymy results in low *recall*, where *recall* is defined as the ratio of the number of the relevant documents retrieved to the number of relevant documents in the database[3].

Polysemy refers to single words that have more than one meaning, *e.g. plane* could refer to an aeroplane, or a flat surface. As a consequence, polysematic terms in queries can lead to matches with irrelevant documents - in effect erroneous matches. Polysemy results in low *precision*, which is defined as the ratio of the number of relevant documents retrieved to the total number of documents retrieved in a query.

In other words the problem is to determine whether the system has to deal with the same word with different meanings or different words that happen to look, or sound, the same[3].

LSI is based on the assumption that words used in a document have some latent semantic correlation. Hence, when these words are compared across documents, certain sets of words will be shared among many documents and absent in others. These words, and the documents they

share, are said to be semantically close to each other[3,5]. A typical LSI system is implemented in several stages[3,5] which are illustrated later in the paper.

We proposed in[6,7] a new approach to the LSI process based on the possibility of using image processing techniques in text document retrieval. This research will focus on different aspects of LSI *e.g*. lexical noise analysis, modeling and removal, and the best database structure which facilitates efficient searching. In[8] this work was presented in a concise manner, and this paper proposes broad investigation in tandem with full analysis. Experimentation is based upon the generation and testing of 'random' TDMs which exhibit different structures, degrees of sparsity and distribution. Moreover, a simple and clear illustration for the decomposition step in the LSI system is presented, which represents the SVD algorithm with different $k - values$. In addition, a greater understanding of the functionality of LSI as a technique used for conceptual and semantic retrieving is provided.

This paper is organised in the following sections. Section 2 introduces the existing work. The LSI components are illustrated in section 3. This gives an overview of the LSI system and the processes involved, along with the necessary mathematical background theory. Section 4 provides a review of *noise models* in image processing applications and the most common filters used to remove the noise. The detailed investigation is presented in section 5. Concluding remarks are given in section 6.

2 Existing Work

It can be argued that there is a clear need to provide new approaches that enhance the data retrieval processes and this is one of the driving forces behind the continued interest in IR systems. In particular, the large body of research in LSI systems reflects this interest. Within LSI systems attention has focused on those elements that are viewed as being computationally intensive and corresponding slow, *e.g.* preprocessing steps or matrix decomposition algorithms. For example, research into the preprocessing stage looks at how to determine what constitutes the keywords in a database, which describe the database and are used as references to the documents titles. The parsing rule used by most researchers[3,5] requires that keywords appear in more than one

documents but do not appear in all documents. The stop words list constructed by Fox[9] has been widely accepted as the norm for identifying the non-meaningful words that can be eliminated from a keywords list.

A considerable amount of work has been devoted as well to producing efficient stemming algorithms for IR[10,11]. A stemming algorithm 'breaks down' words into stems, *e.g.* the keywords " *read* ", "*reader*", "*reading*", can all be 'broken down' into the stem "*read*". This stem can then be used as one keyword rather than having to store the original three separate keywords.

As with stemming algorithms, there exists a considerable body of work on term weighting[3,12]. Term weighting is one of the common methods for improving retrieval performance. It functions by giving terms in the TDM different *weights*. In practice, local and global weights are applied to increase, or decrease, the importance of terms within or among documents in the TDM.

Research has moved beyond the basics of the LSI process, and several alternative decomposition algorithms to SVD have been suggested, including QR factorisation[3] and semi discrete matrix decomposition (SDD)[13]. In Unitary Operators on the Document Space[14], Hoenkamp shows that the decomposition underlying LSI is an example of a unitary operator. Hoenkamp proposed the use of the Haar wavelet transform (HWT) as an alternative as this transpose shares the unitary property and has a much reduced computational cost. This line of research showed some promising initial results. Furthermore, the concept of representing the TDM as a gray scale image, as illustrated in Fig. 1 (The Cochrane database contains titles of medical studies[15]) was postulated. In such a model the white dots in the image (non-zero values) represent the keywords in the document sets. In addition, it has been argued that using the HWT to remove noise from an image is equivalent to using the HWT to remove lexical noise from the TDM.

There are several studies into using LSI in tandem with other techniques, such as document clustering[16]. The results reported in[16] show that the accuracy of the LSI technique may be improved when retrieving from clustered subsets. However the number of clusters to choose remains an unsolved problem, which affects the performance of a

clustered SVD retrieval system. One of the most recent works has emphasized on dimension reduction in the LSI system[17].

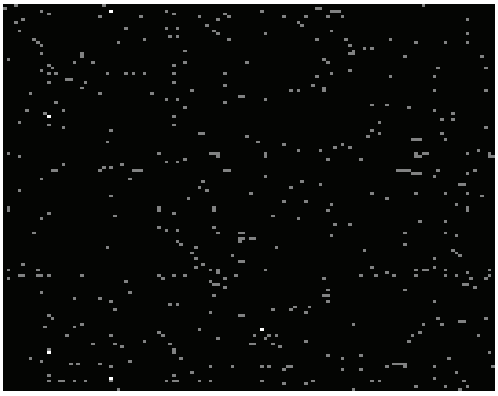

Fig. 1 Cochrane's TDM represented as a grey scale image.

Perhaps the most surprising applications of LSI research have been in fields other than IR. The principles underlying LSI have been applied to imbue machines with human-like learning capabilities[18,19]. In[20] the use of both keywords and image features to represent documents has been presented in order to improve the retrieval performance. Other researchers have used LSI in the field of image retrieval[21].

3 LSI System Components

In this section the different components of the standard LSI are illustrated.

3.1 Document Preprocessing Description

As mentioned before, the database needs to be converted to a TDM. Before this can be achieved, preprocessing has to be carried out on the document set. Punctuation and meaningless words need to be removed, and the keywords necessary for construction have to be extracted for comparison with the user′s query[3].

Initially all the text in each entry of the database is extracted, and joined together to form a large collection of the terms that appear in each of the documents. This list is then processed by:

• Removing punctuation characters "0123456789*:;*:;′ ()[] *etc:*", as these do not contribute to the meaning of terms in the documents.

• Removing "stop words". These are words that have no meaning and so do not represent any of the semantic structure of the documents. Examples of "stop words" are *the, probably, however, etc*.

• The next step in preprocessing involves removing duplicate words in the list of keywords. Because all the words were extracted from each entry in the table, many words will appear more than once and have to be removed. This is achieved by sorting all the terms in the keywords list alphabetically, thus all duplicate words are adjacent in the list. A recursive function can be used to compare each adjacent pair of words in the keywords list, and if they are the same, the duplicate term is deleted.

• Finally, a list that comprises all keywords in the documents set is obtained, along with a list of keywords in each individual document.

3.2 Memos Database Example

To illustrate the preprocessing step, let consider the *Memos* Database as an example[5]. The titles in *Memos* database are presented in Table 1.

Table 1. Memos Document Set[5].

B1	human computer interface for ABC computer applications
B2	a survey of user opinion of computer system response time
B3	the EPS user interface management system
B4	system and human system engineering testing of EPS
B5	relation of user perceived response time to error measurement
B6	the generation of random, binary and ordered trees
B7	the intersection of paths in trees
B8	graph minors IV: widths of trees and well-quasi ordering
B9	graph minors: a survey

Preprocessing produces the following set of unique keywords (in Table shown in bold):

{human, computer, interface, survey, user, system, response, time, EPS, trees, graph, minors}

3.3 Term Document Matrix

Once preprocessing is complete, the TDM is constructed from a list of terms that characterizes the structure of all the documents and the keyword list for each document that was generated in the previous step. Each row of the matrix is assigned to a term, and each column of the matrix is assigned to a document. The value that appears in position (i, j) of the matrix is the number of times that the keyword assigned to the i^{th} row appears in the document assigned to the j^{th} column. Most values in the matrix are zero, as only a subset of keywords appears in any given document. The TDM generated for the *Memos* example is shown in Table 2.

Table 2. TDM for Memos Example[5].

	B1	B2	B3	B4	B5	B6	B7	B8	B9
Computer	2	1	0	0	0	0	0	0	0
Eps	0	0	1	1	0	0	0	0	0
Graph	0	0	0	0	0	0	0	1	1
Human	1	0	0	1	0	0	0	0	0
Interface	1	0	1	0	0	0	0	0	0
Minors	0	0	0	0	0	0	0	1	1
Response	0	1	0	0	1	0	0	0	0
Survey	0	1	0	0	0	0	0	0	1
System	0	1	1	2	0	0	0	0	0
Time	0	1	0	0	1	0	0	0	0
Trees	0	0	0	0	0	1	1	1	0
User	0	1	1	0	1	0	0	0	0

Each column in the database can be considered as a vector describing the document it represents, each row can be considered as a vector describing the term that it represents. There is undoubtedly a great deal of lexical noise in this process, as illustrated by the redundancy in the matrix. The LSI process seeks to eliminate this redundancy by decomposing the TDM using the SVD algorithm.

3.4 Query Vector

In order for searches to be carried out, queries must also be represented in vector form. This is achieved by the same process that is used to convert documents into columns in the TDM. Keywords are extracted from the query, and if a keyword also appears in the document set then the number of times it appears in the query is recorded using the same format as one of the document vectors in the TDM. For example the query 'response time' would be converted to the form (0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0) as 'response' corresponds to the seventh row of the TDM, and 'time' corresponds to the tenth row, and each word appears once in the query. In effect, the query is a pseudo-document.

3.5 Singular Value Decomposition (SVD)

A matrix M can be decomposed into an approximate, reduced, form as:

$$M = U*S*V^T \quad (1)$$

Where U is the singular row vectors of M, S is a diagonal matrix holding the singular values of M in ascending order and V^T is the transpose of the singular value column vectors of M[3]. The diagonal elements in S are stored in ascending order[3]. The higher order values are larger and this means that they represent more of the semantic content of M (Fig. 2). By contrast, the lower order values are small and can be viewed as "lexical noise"[14].

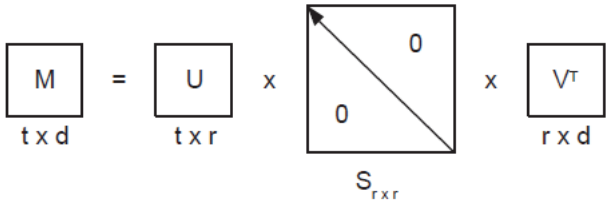

Fig. 2. SVD decomposition of (t x d) TDM.

In Fig. 2, t = number of terms in the TDM, d = number of documents in the TDM, and r = the rank of M.

At the heart of LSI is that the latent semantic structure of the document set is identified by the matrix of diagonal values (Fig. 3).

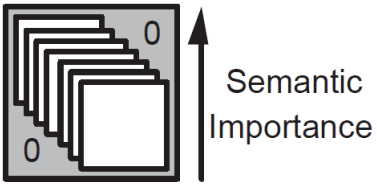

Fig. 3. Diagonal Matrix S. The inner boxes represent singular values.

The original TDM can be approximated by multiplying the three matrices U, S and V. However, if the lowest singular values of S are discarded, then the TDM can be approximated by:

$$M_a = U_k * S_k * V_k^T \quad (2)$$

Where $k < r$, M_a = approximated TDM, U_k = first k columns of U, S_k = the new matrix of singular values, and V_k^T = transpose of the first k columns of V[3] as illustrated in Fig. 4.

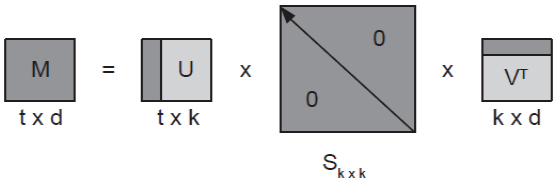

Fig. 4. Approximated TDM.

The resultant approximated matrix has the same dimensions as the original TDM and represents the best k-rank approximation of M in terms of the Frobenius Norm and p-Norm[22]. The query can then be compared to each document in the new approximated matrix. With "lexical noise" removed, this should lead to improved results when the query is compared to the approximated documents.

3.6 Metrics Methodology

Each column of the TDM represents a document in the original document set in vector form as shown in Table 3. This is also true for the approximated TDM. The query is a row vector constructed such that its transpose can be considered equivalent to a document vector containing

only the words that appear in the query. In effect, the query is a pseudo-document. For example the query (0 1 0) is a 3 dimensional row vector.

Table 3. Each column represents a document as a three dimensional column.

0	0	1	1
0	1	1	0
1	0	0	0

Each document vector in the approximated TDM can then be compared to the query by calculating the cosine between them. The cosine is calculated from the following equation:

$$\cos \Phi = \frac{a_j^T q}{\|a_j^T\| \|q\|} \quad (3)$$

Where a^T_j is the transpose of the j^{th} document vector in the approximated matrix a, q is the query vector, $||a^T_j||$ is the modulus of a^T_j , $||q||$ is the modulus of q.

The modulus is equivalent to the Euclidean norm: $||q|| = \sqrt{} (q_1^2 + q_2^2 + q_3^2 \ldots + q_{n-1}^2 + q_n^2)$.

A cosine value of 1 means that both vectors exist in exactly the same dimensional space. Below this value the vectors become steadily less similar. In order to determine which documents are similar enough to be returned in response to a user's query, a threshold of 0.5 is set which it is usually selected by the most researchers in this area. A suitable threshold value could be determined based on certain heuristics and experiments, which could be an interesting topic for future study[16], [23].

4 Noise Modeling and Removal in Image Processing

This section provides a study of the mathematical representation and removal of noise in image processing. This brief understanding will be used to address some important issues in LSI systems. In particular, modeling the lexical noise in the TDM, that represents defining the nature of the noise in the text processing. Furthermore, to provide a

comparison between the noise in both LSI and image processing areas, in order to evaluate the use of image processing techniques for lexical noise removal in text processing.

Noise is any undesired information that contaminates an image. The aim of image processing is to improve or enhance the quality of an image by removing the noise from the image. Most of the techniques employed in processing an image look at a pixel and its immediate neighbors and make some sort of calculation involving these values. Noise can usually take the form of extreme intensity values. However, in some cases, the high intensity values represent important information that should not be lost. Clearly, addressing the nature of noise is an important issue for noise filtering applications.

The next two sections describe some of the most common noise models that appear in images followed by methods used to remove it.

4.1 Various Noise Models

Noise applied to a noise free image can degrade the image to such a point that important features are no longer observable. Noise can arise from many places such as sensor problems in a camera, dust covering the optics or can be picked up during a point to point transmission of an image[24]. The classification of noise is based upon the shape of the probability density function (PDF) or histogram for the noise.

In typical images the noise can be modeled with either a *salt − and − pepper* (*impulse*), *uniform*, or *gaussian* (*normal*) distribution[24-26]. The three models are clarified below.

4.1.1 Salt-and-Pepper Noise

This type of noise commonly appears in images. This noise is named for the *salt − and − pepper* appearance for the infected image. With this noise, the corrupted pixels are either set to the maximum value or to zero. Unaffected pixels always remain unchanged.

The PDF of *salt − and − pepper* noise is given by the following equation:

$$P(z)=\begin{cases} \mathrm{p_a} & for \quad \mathrm{z = a} \\ \mathrm{p_b} & for \quad z = b \\ 0 & otherwise \end{cases} \tag{4}$$

Where *a* and *b* represent two separate gray-levels. If $b > a$, gray-level *b* will appear as a light dot in the image. Conversely, level *a* will appear as a dark dot. If neither probability is zero, and especially if they are approximately equal, impulse noise values will resemble *salt − and − pepper* dots randomly distributed over the image[25].

4.1.2 Uniform Noise

Here, the noise gives a diffused look to the picture. *Uniform* noise is useful because it can be used to generate any other type of noise distribution and is often used to degrade images for the evaluation of image restoration algorithms because it provides the most unbiased or neutral noise model[26].

The PDF of uniform noise is given by:

$$P(z) = \begin{cases} \frac{1}{b-a} & if \quad a \le z \le b \\ 0 & otherwise \end{cases} \tag{5}$$

Where *z* represents gray-level.

4.1.3 Gaussian Noise

This noise blurs the image and hides the detail of the image. It is characterized by two parameters, mean and variance.

The PDF of Gaussian noise is given by:

$$P(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(z-\mu)^2/2\sigma^2} \tag{6}$$

Where *z* represents gray-level, μ is the mean of average value of *z*, and σ is its standard deviation. The standard deviation squared, σ^2, is called the variance of *z*. When *z* is described by the PDF above, approximately 70% of its values will be in the range $[(\mu - \sigma) , (\mu + \sigma)]$, and about 95% will be in the range $[(\mu - 2\sigma) , (\mu + 2\sigma)]$ [25].

4.2 Noise Removal Filters

The reduction of noise present in images is an important aspect of image processing. It is generally recognized that noise removal should be the first step of any image processing method. This section presents a review of the most common spatial filters used to remove the noise.

4.2.1 Average Filter

Averaging or *mean* filtering is a simple, intuitive and easy to implement method of smoothing images, *i.e.* reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images. The *average* filter is the simplest linear spatial filter. This is a lowpass filter, which removes high spatial frequencies from an image. It works by passing a mask (usually a 3x3 grid) over the image, calculating the *mean* or *average* intensity for the mask, and setting the central pixel to this value. For example, Fig. 5 shows a mask to be used on the left, and the group of pixels it will be applied to on the right[24].

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

2	3	3
2	3	18
3	19	20

Fig. 5. Average filter example.

4.2.2 Gaussian Filter

Another common linear spatial filter is the *gaussian filter*. This filter gives a higher weighting to values which are closer (and thus should be more influential) to the central pixel. It is used to blur or smooth images and remove detail and noise, and in this sense it is similar to the *average* filter, but will preserve edges better than the more basic *average* filter. An example *gaussian* filter is given in Fig. 6.

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

Fig. 6. Gaussian filter example.

This is also a lowpass filter, which attenuates high spatial frequencies present in an image. This filter works very well in the presence of additive *Gaussian* noise.

4.2.3 Median Filter

The *median* filter is a nonlinear spatial filter that is good at removing *salt − and − pepper* noise from any kind of image. The strength of the function is controlled by specifying the size of the neighborhood (the surrounding pixels used for calculating the median value). This function causes minimal blurring of the image. It also does a better job than the *average* filter at preserving edges within an image.

The *median* filter considers each pixel in the image in turn and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. It replaces the pixel with the *median* of those nearby neighbors. Fig. 7 illustrates an example calculation, where a value that is clearly wrong gets filtered out[24].

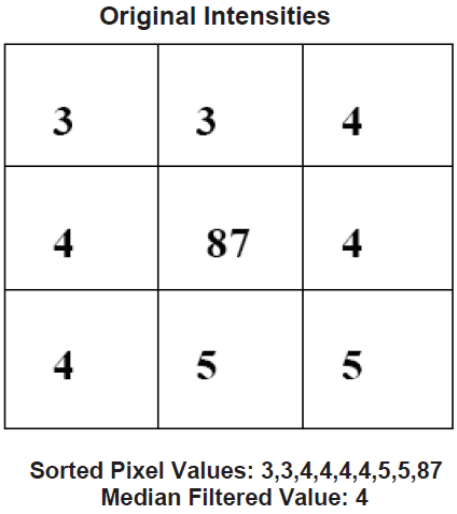

Fig. 7. Median filter example.

The noisy pixel with intensity 87 would be filtered out and replaced with 4, a much better approximation to the value.

However, there are some obvious problems with these filters. The *average* filters are good at removing *gaussian* noise but blur the edges and *median* filters will always filter in favour of the more dominant region, and thus work very well with *salt − and − pepper* noise. *Mean*

(*average*) filters average the central pixel with neighbouring pixels, which is fine when all of the pixels are part of the same region. When the filter is near an edge however, it starts to include pixels from what could possibly be a variety of different regions.

5 Investigation Method

As mentioned before, the bulk of research into LSI systems has focused on:

• the preprocessing step on the database.

• the decomposition step, by testing different alternatives for the SVD.

• the clustering technique to deal with large databases.

• the determination of the best $k - value$ in the SVD algorithm.

However, there is a lack of research into mathematical models for the noise in the LSI system. As shown, in many seminal works on LSI, the mathematical perspectives of the LSI system components have been presented, *e.g.* TDM decomposition, which represents the lexical noise removal step, and similarity measure (cosine) to rank documents according to their similarity with a query. There is not much work investigating the nature of the noise in the TDM which represents a database.

The preceding section presented a brief overview of noise in image processing and the various techniques used to remove such noise from images. Using this information as a base some important issues in LSI system will be addressed. In particular, attention will focus on modeling the lexical noise in the TDM, which represents defining the nature of the noise in text processing, and the different types of this noise which can be specified by looking to the sources of each noise.

In[14], the TDM was visualized and treated as an image. It was suggested that image processing techniques could be applied to this visualization in the LSI system as a first step to assist in eliminating noise from the TDM. The intention was to improve the quality of the LSI application in terms of computation cost. This investigation can explain the suggestion of applying these techniques to the concept of ’noise’ in the TDM visualization.

5.1 Lexical Noise

After the creation of the TDM, that is, a two-dimensional matrix that represents the number of times a keyword appears in each document in the database, the resulting matrix will be sparse, that is a large proportion of elements will be zero, as each keyword will only appear in a relatively small number of documents. The zeros in the matrix represent lexical noise or redundancy in the sparse matrix as clearly illustrated in our previous work[27].

It is suggested that there are (at least) three views, or types, of noise in the context of LSI as the following:

• Traditional noise as viewed in LSI - *e.g.* the stop words (a, an, of, the, *etc*.). This type of noise is usually processed and removed at the preprocessing step for the database.

• Noise generated by poor structure of the database or query model being used. Longer document descriptions increase the number of the keywords and the distribution of the non-zero values in the TDM, which in turn helps in improving the semantic meaning among the documents in the database. While on the other hand, the shorter document descriptions represent poor database structure that does not support LSI searching, as the sparseness in the TDM increases for such types of structure. More investigation on this type or view of noise is presented in the next section.

• New types of noise being generated by *spammers* or others trying to avoid filtering systems on mailers. This view of noise can be explained by giving an example. Let consider a company is called *Silver*, and the owners want to have this appear as many times as possible in a sentence. Some advertisers will not allow this so instead of saying

"Buy from *Silver* as *Silver* is the best trader available and *Silver* will always give good value and *Silver* will never let you down".

They will reword the line by writing

"Buy from Silver as Sil ∗ ver is the best trader available and S ∗ ilver will always give good value and Silve ∗ r will never let you down".

This type of restructuring makes it difficult for a simple filter to detect the occurrence of *Silver* as the structure of the word has been

adjusted (by the inclusion of a simple 'buzz' letter). However, a human reader can still detect the word with ease. It is a feature of the way that people read information. Many examples of this type of noise occur in e-mails, as *spammers* try to avoid spam filters with this type of approach. Such structures can be viewed as noise, but how this noise can be represented, detected and removed from a message is an interesting question, and suitable for research in future work.

The noise in text processing context is quite subtle and complex. A typical TDM is visualized as a typical grey-scale image. An 8-bit grey-scale image will have values between 0 and 255, and is likely to have a good spread of values in this range. In a typical TDM, most values are 0 or 1. Hence, noise reduction using the conventional image processing filters mentioned above will not be able to remove the redundancy in the TDM e.g. by passing a mask over the TDM visualization. Even before applying these filters, most values in a TDM are already close to zero. The difference between redundant information and important information may simply be less clear cut. Candidates can be found in image processing applications, which have similar aims as LSI, but under the guise of lossless and lossy compression[14]. The choice of such techniques can be explained by the following:

• One of the advantages of LSI is that it seems to remove (lexical) noise from the TDM by dimension reduction. In SVD, dimension reduction is achieved by zeroing the smallest eigen values in the diagonal matrix *S*.

• The analog for the image processing transform is to zero the smallest coefficients.

• In both cases this is interpreted as removing (lexical) noise.

The next section presents an investigation into the structure of the TDM in the LSI system. Test TDMs that exhibit certain features are generated and tested in the LSI system, with the aim of determining the best database structure which will lead to an effective LSI searching.

5.2 Term Document Matrix (TDM) Modeling

This section presents different investigations for the LSI system, more precisely on the TDM, and in addition provides a simple illustration for the SVD algorithm at the decomposition stage in the LSI process. A

number of TDMs and query vectors, with different structures and degrees of sparsity, are generated and tested in the LSI system. The aim behind this analysis is to study the effect of structure, degree of sparsity, distribution and any other attributes of the TDM on the search results with the obvious goal of determining the best characteristics of TDM that will give the best results.

The LSI search is carried out for the different random TDMs which are presented in the following sections. A number of figures for the TDMs and the queries are generated to present clear illustrations.

5.2.1 The TDM Diagonal

This first group of experiments considers the effect on the LSI system of nonzero values in the diagonal of the TDM, and varying the number of non-zero elements in the query vector. The fact that the singular values of the S diagonal matrix of the decomposed TDM are in descending order is important. By keeping k singular values, the k highest singular values are returned, which represents the lexical noise removing step. Let us consider that the original TDM is a diagonal matrix of ones, it is expected that the resultant three matrices of the decomposition step are diagonal as well. Ignoring a number of singular values in the S diagonal matrix will result in removing the same values in the original TDM. With this database structure, the standard vector space model (VSM), which is the model underpinning LSI but without the SVD algorithm or the dimension reduction step, is predicted to outperform the LSI system.

In other words, the LSI will perform inefficiently for such database structures. The details of the investigation are presented in the following examples:

• **Example one:** In this example a 15x15 diagonal matrix, where the diagonal elements are ones and non-diagonal elements are zeros, is generated to test in the LSI system, a pseudo query vector is also created. For Fig. 8, the results of the search always return one document at different $k-values$ for the SVD algorithm in the LSI process, and it is the first document in the TDM. At the decomposition step in the LSI system, the SVD decomposes the TDM into three matrices, one of them the diagonal matrix *S*, that holds the singular values of the original TDM in

ascending order, in order to apply dimension reduction to this matrix to remove the small singular values which represent the noise.

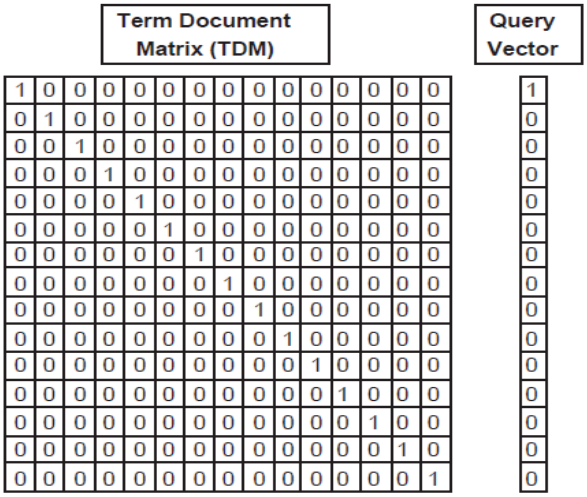

Fig. 8. Diagonal TDM of 1's and one term query vector.

The same procedure is applied in the above example, where the TDM is a diagonal matrix of ones, and it is decomposed into three diagonal matrices. Since there is no change to the TDM, the singular diagonal values of the *S* matrix are still ones, and applying the dimension reduction at different $k - values$ on the matrix will result in ignoring the same diagonal values in the original TDM. Fig. 9 clarifies this point.

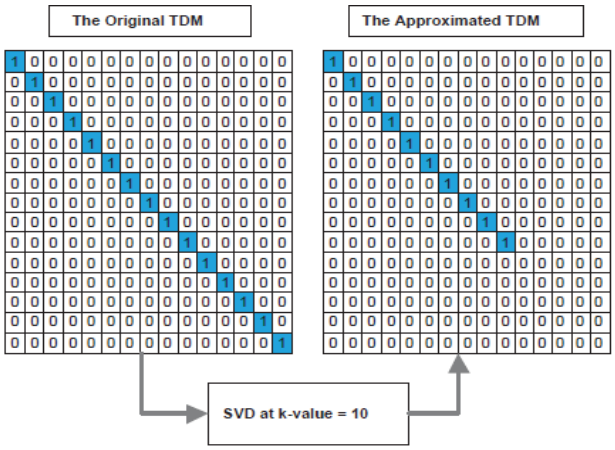

Fig. 9. The original diagonal TDM and the approximated TDM.

This query will always return one document, that being the first document in the TDM, which has a cosine value of one, indicating an identical match with the query, it is also important to note that no other cosine values are returned above zero for the other documents. This occurs due to the query having only one term, and since the matrix is diagonal, the term will only appear in one document, the first document in this case.

• **Example two:** The query vector is amended for this example by adding another term at the bottom of the vector as shown in Fig. 10. In this example, and when the LSI is applied at different *k−values*, only one document is returned which is the first in the TDM.

Term Document Matrix (TDM)															Query Vector
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Fig. 10. Diagonal TDM of 1's and two terms query vector

The query intersects the TDM at the first and last documents, and as mentioned before the TDM is diagonal and therefore the *S* matrix is also diagonal, and so removing any elements from the diagonal values results in removing the same diagonal values in the original TDM. Thus, when the SVD is applied at the range of $k - values$ (1-14), only the first document is returned, because at this range of values the last document, which is document number 15, is ignored. It is important to note that, applying the LSI at *k−value* 15, which means no dimension reduction

occurs, returns the two documents, the first and the last documents in the TDM, which have a cosine value of 0.7071. This indicates a high match with the query, since the query differs from both documents by one term. Suppose the VSM system is applied for this search query, since no dimension reduction is performed. Two documents will be returned and thus this baseline method outperforms the LSI system at this search for such a database.

• **Example three:** Figure 11 shows that, the query is again amended, by adding more terms, and used for the same database. In the case of three terms, one term is added to the query which is term number four. The query meets the documents one, four and fifteen, which have a cosine value of 0.53. Therefore, if no dimension reduction, or in other words the VSM, is applied, the three documents will be returned for this query.

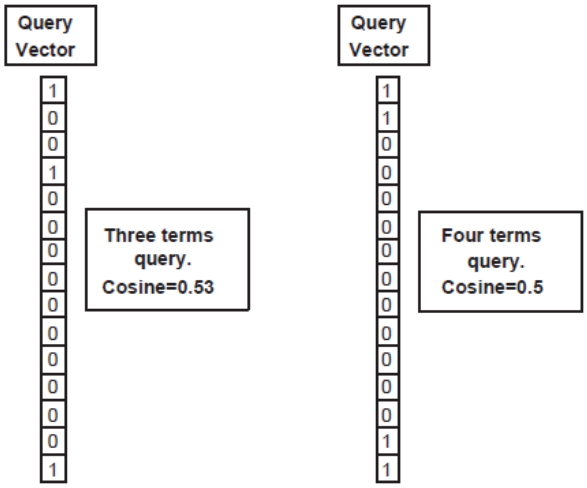

Fig. 11. Three and four terms query vectors.

When the SVD is applied at range of *k−values* (1-14), only the first and the fourth documents are returned, and for *k − values* less than four only the first document is returned. In the second case, four non-zero items in the query intersect with four documents in the TDM, the first and last two documents. Again for this case, the dimension reduction step affects the number of documents returned as the values removed from the *S* diagonal matrix are the same values in the original TDM. The four documents can be returned with VSM as the cosine value they have is 0.5. As shown in the above examples, the similarity cosine value

decreases as the number of terms in the query increases. Hence, for any number of terms more than four in the search query, no results are returned as the cosine value for all the documents is lower than 0.5.

The above examples clearly show that, such a TDM structure does not support LSI searching, since it represents a very poor database, which contains 15 documents each of which consists of only one word. Large sizes of the same structure of TDM have been tested and produce the same results.

5.2.2 The TDM Column

This section tests other structures within the TDM, in order to determine more potential features that would improve efficiency. Let us consider that the original TDM is a matrix of columns of ones, which represents a database with certain documents containing all the keywords in the database. The query vector with different length is investigated as well. Again, it is expected that, this structure of database will not support the LSI search, with no real impact for the decomposition step on the results returned. The following examples show the results of this examination:

• **Example four:** In this example a 15x15 matrix, where the first column elements are ones and elsewhere are zeros, is generated to test in the LSI search.

For Fig. 12, no results are returned in this search at any of the $k-values$. As shown in the figure, the TDM has only one column of ones, which is the first, and zeros elsewhere, and the query for this search contains one term and it is the first one. The difference between the query and the first document in the TDM is large, therefore the cosine value for this document is less than 0.5, which is the threshold cosine value, all documents not exceeding the threshold are not returned.

In the previous examples of TDM structure, the S diagonal matrix was the same as the original TDM, which had diagonal values of ones, therefore the stage of dimension reduction affects the original TDM by eliminating some values.

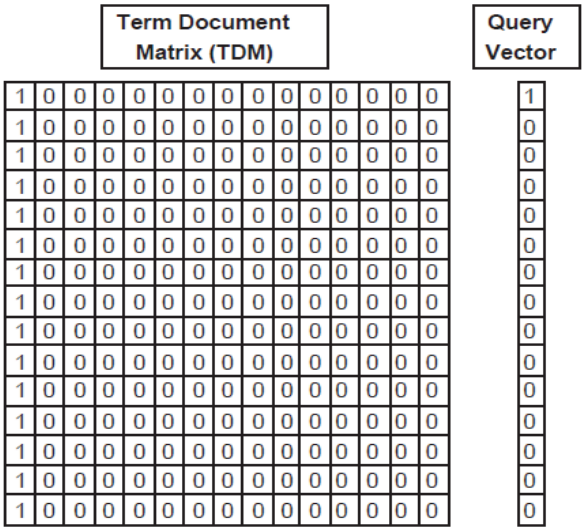

Fig. 12. TDM with one column of 1's and one term query vector.

In this example, and at different *k−values*, the approximated TDM remains the same as the original TDM with no change (no sparseness or noise is removed). Fig. 13 shows the *S* diagonal matrix of the TDM.

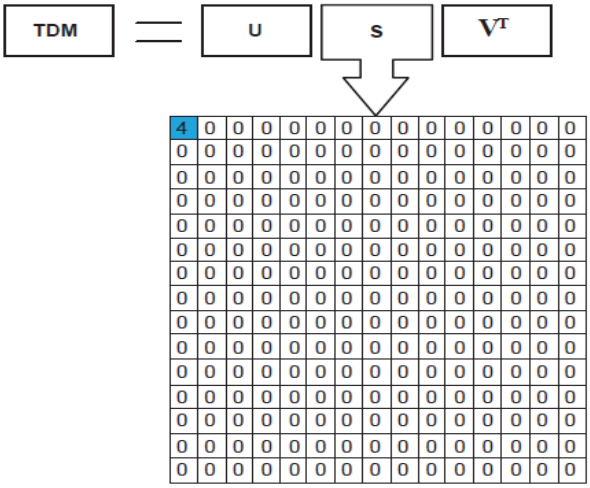

Fig. 13. The S diagonal matrix of the column TDM.

As shown, the *S* diagonal matrix contains only one non-zero value and it is the first value, whereas all other diagonal values are zeros, because the original TDM only contains one column of ones, indicating a bad distribution of the non-zero values in the TDM. As a result and since the *k−value* for the SVD algorithm must be at least one, the dimension

reduction stage at different $k-values$ has no effect on the TDM, and thus, the lexical noise is still present in the matrix.

• **Example five:** In Fig. 14, another column of ones is added to the TDM, and the search is again applied for the same query.

Term Document Matrix (TDM)

1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Query Vector

1
0
0
0
0
0
0
0
0
0
0
0
0
0
0

Fig. 14. TDM with two columns of 1's and one term query vector.

Again, the same scenario occurs as in the previous example, but there is one more extra column of ones, which is the third in the TDM. No results are returned at the different $k - values$ for this query. For this example, and for more than three terms in the query as well, the cosine value of the two documents exceeds 0.5, and documents one and three in the TDM are returned at the different $k - values$ as will be shown in the next example. The S diagonal matrix for this TDM contains, as in the previous example, a high value and it is the first value of the diagonal, the second and the third values are very small negative values, and all the other diagonal values are zeros. The SVD at the small $k - values$ (1, 2), which remove these small values has insignificant impact on the TDM, and again that is due to the bad distribution of the values in the TDM. The next example presents more clarifications of these points.

• **Example six:** The example makes changes to the query by adding some terms which results in improving the similarity by increasing the cosine value between the query and the first document in the TDM.

Consequently, and for more than three terms, the cosine value of this document exceeds 0.5, and the document is returned at the different $k-values$. The number of columns of ones is increased in this example, as well as the number of terms in the query, as illustrated in Fig. 15, in order to achieve sufficient similarity between the documents and the query vector to exceed the threshold value 0.5.

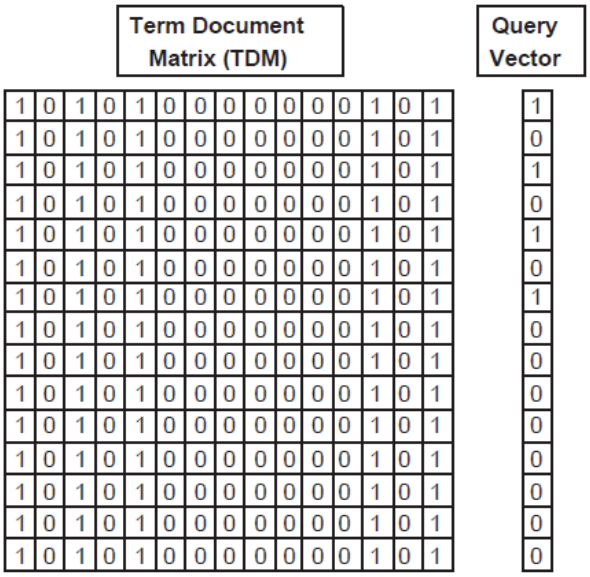

Fig. 15. TDM with five columns of 1's and four terms query vector.

For this search, five documents, which are represented by the five ones columns in the TDM, are returned for any $k-values$ in the range (1-14).

The similarity cosine value for the five documents is 0.516, which cannot be obtained at query with a number of terms less than four. The order of the terms in the query is not important since the non-zeros columns of the TDM are all ones. The different $k-values$ have no effect on the number of documents returned.

This can be explained by observing the diagonal matrix in Fig 16. The *S* diagonal matrix for this TDM contains, as values (shown as zeros in the figure), and all the other diagonal values are zeros. The approximated TDM is roughly the same as the original one at the full range of k − values, because removing these small negative values has a minor impact on the TDM, as the concept of the TDM (non-zero

elements) is preserved in the high positive value of the S diagonal matrix. This can explain the reason for obtaining no results for both the LSI and the standard VSM for this search query. In other words there is no advantage in using the LSI system, in such a situation. Once again, this is due to the bad distribution of the values in the TDM.

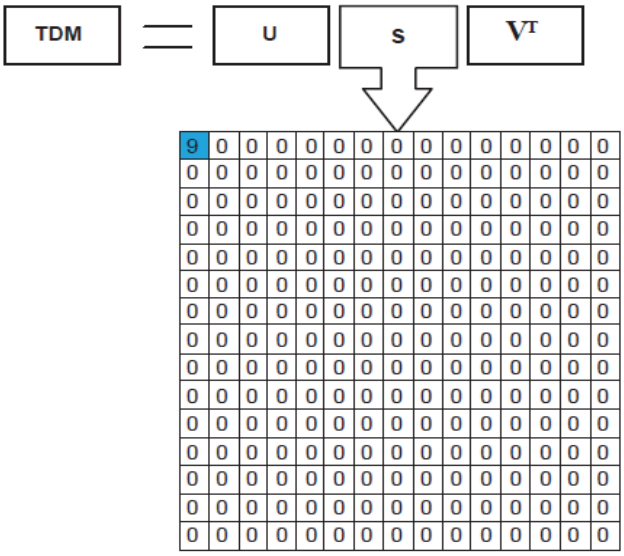

Fig. 16. The S diagonal matrix of the five columns TDM.

For the examples in this section, the TDM contains only a number of columns of ones, these columns representing a document in the database.

In other words, all the keywords in the TDM appear only in these documents while other documents contain no words. Such poor structure in the databases does not reflect the basic trends in the LSI system, e.g. very bad distribution for the non-zero values in the matrix (the appearance for the keywords in the documents of the database), even though the volume of non-zero values in the TDM according to the size of the matrix is good especially in the last example. No results are returned for query vectors of one, two or three terms, as the number needs to exceed three terms in order to retrieve results, which is impractical and regarded as a weakness of the database.

5.2.3 The TDM Row

Row structures for TDM are examined in this section, with the goal of revealing more characteristics that would indicate an effective

database structure. As in the previous structures, a poor performance is expected from the LSI system, since a database with this structure contains certain keywords that appear in all the documents, which does not help in determining the semantic meaning between the documents in the database. As before, the effect of the length of search query and the decomposition step on this structure for the database are presented in the analysis of the results in the following examples:

• **Example seven:** In this example a 15x15 matrix, where the first row items are ones and elsewhere are zeros, is generated and tested in LSI.

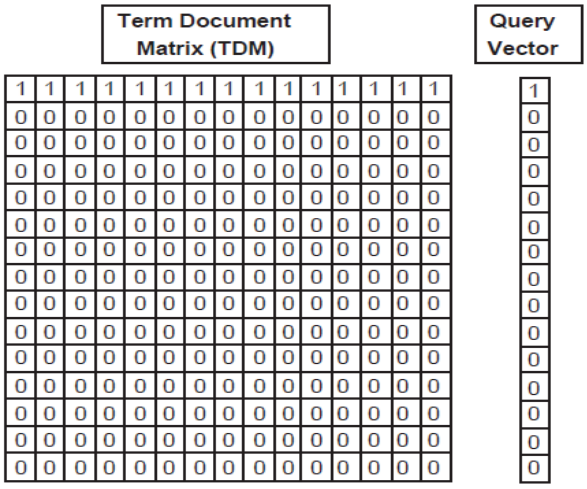

Fig. 17. TDM with one row of 1's and one term query vector.

In Fig. 17, 15 results are returned in this search at the full range of *k − values*. As shown in the figure, the TDM has one row of ones, which is the first row, and zeros elsewhere, and the query for this search contains one term and it is the first one as well. Hence there is an identical match between the query and all the documents in the TDM, therefore the cosine value for these documents is 1. It is clear that, if the query contains terms other than the first one no results will be returned, as the query intersects the TDM at the first row.

In this example, as in the previous structure of TDM, and at the different *k −values*, the approximated TDM remains the same as the original TDM with no change.

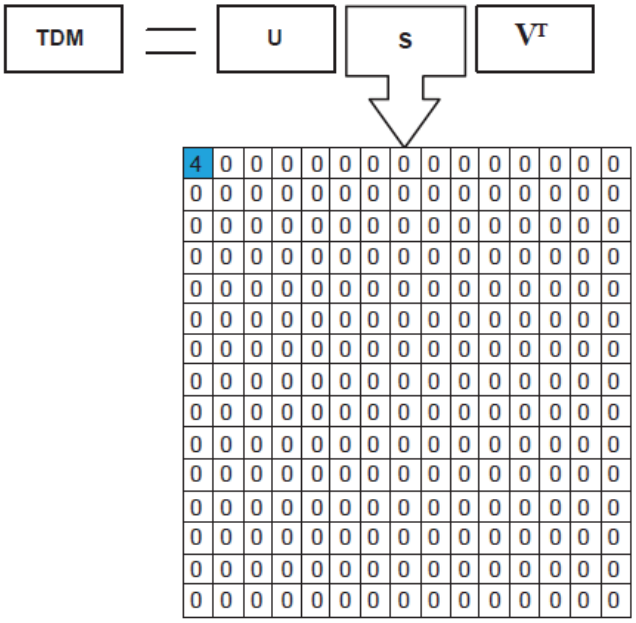

Fig. 18. The S diagonal matrix of the row TDM.

As shown in Fig. 18, the *S* diagonal matrix has one non-zero value which is the first value in the diagonal, whereas all other diagonal values are zeros, because the original TDM contains only one row of ones, which indicates bad distribution for the non-zero values in the TDM. As a result and since the $k - value$ for the SVD algorithm must be at least one, the dimension reduction stage at different $k - values$ has no effect on the TDM.

If the number of keywords is increased in the search query, the similarity cosine value is expected to decrease for the same database in this example. Therefore, for more than four terms the cosine value of this document does not exceed 0.5, and thus no documents are returned as clarified in Fig. 19.

• **Example eight:** For the example shown in Fig. 20, a TDM with two rows of ones, which are the first and third ones, and a query with two terms, which are the first and the fourth ones, are tested in LSI.

In this search no results are obtained at the different $k-values$. The query vector meets all the documents in the database at first value, and differs at the third and the fourth values, which results in similarity

cosine values less than 0.5. The *S* diagonal matrix for this TDM contains, as experienced before, a high value and it is the first value of the diagonal, and all the other diagonal values are zeros. The SVD at the full range of $k - values$ has no impact on the TDM, and again, that is due to the bad distribution of the values in the TDM.

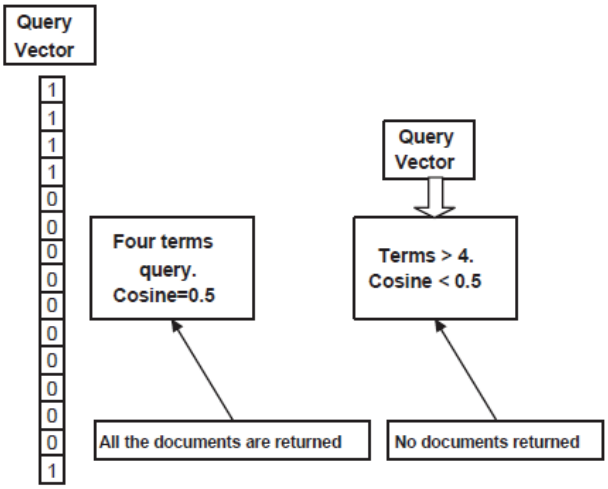

Fig. 19. Search queries for example seven.

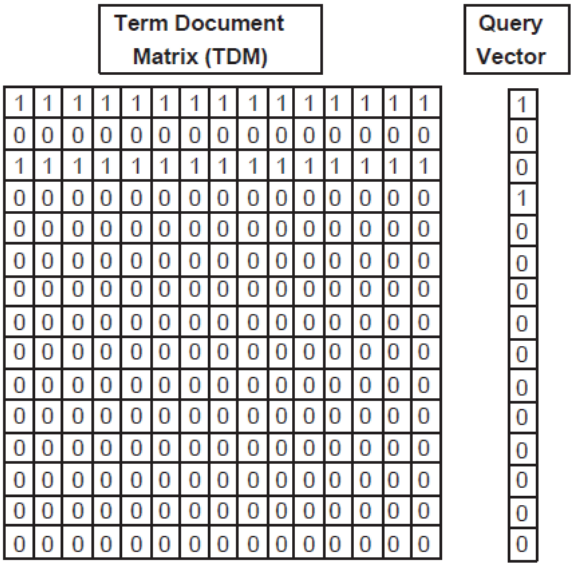

Fig. 20. TDM with two rows of 1's and two terms query vector.

5.3 Analysis

In this paper, an initial investigation of the noise in the TDM has been presented. Random TDMs with different structures and degrees of sparsity were generated and tested in the LSI system using some random query vectors. The goal being to determine the best description of the TDM structure which gives the best search results. An LSI system attempts to compare the conceptual meaning among documents rather than words alone, by returning documents that are similar in meaning, even though the keywords in the search query may not appear in the document's description. For the examples used in this work, the TDM contains only diagonals of ones, a number of columns of ones or a number of rows of ones have been tested. In the diagonal case, each document contains only one term, which means the documents do not share any words with others. The *S* diagonal matrix of the SVD algorithm is the same as the original TDM, which means that the singular diagonal values of the *S* matrix are still ones, and applying the dimension reduction at different *k – values* on the matrix will result in ignoring the same diagonal values in the original TDM resulting in the loss of important information. The power of the LSI in finding the latent semantic meaning in the database cannot be demonstrated for such database structures.

For the column structure, each column represents a document in the database, which means the keywords in the TDM appear only in these documents while other documents contain no words. Therefore, some documents identically match each other, while on the other hand with other documents no words are shared. Such poor structure of databases does not enable the LSI system to perform accurately, *e.g.* very bad distribution of the keywords in the documents of the database, even though the volume of non-zero values in the TDM according to the size of the matrix is good. The non-zero values are usually preserved in the first diagonal value of *S*, while the rest of the values in the diagonal are zeros. Therefore, applying the SVD algorithm at different *k −values* does not remove the lexical noise in the TDM. As a result the LSI and standard VSM produce the same results in such cases.

The same scenario as the column structure is obtained with the row structure. The only difference with this structure is that, non-zero rows represent the appearance of a certain keyword in all the documents, and

indicate at the same time that some keywords do not exist in any documents. This is again a bad structure for the database, indicating a bad distribution. These words that have dominant existence represent stop words and they add no meaning to the documents. In addition the dimension reduction has no affect on the original matrix.

Database structures used in these examples may not exist, as no document is presented in the TDM unless it has a number of keywords, and at the same time no keyword is indexed unless it appears in at least a couple of documents. However, the aim behind the work presented in this paper is to specify the good characteristics of a database that support the LSI searching, as a technique used in finding the similarity between documents. In addition, to present a simple and clear practical illustration for the functionality of the LSI system, in particular the dimension reduction stage.

In order to achieve an ideal effective structure, a good structure for the database is needed as illustrated in the following:

• The sparsity (volume of non-zero values) and a good distribution for the non-zero values in the TDM are very important attributes to obtain a good structure of database that will support the effective use of LSI searching

• Databases with more entries and longer document titles have more keywords with better distribution within the documents. As a result, this will produce better search results and support the use of the LSI system.

• The LSI system works better with large datasets, as such databases assert the important features, listed above, necessary for effective structure of database, which helps the SVD to deal with the lexical noise more efficiently.

• Although a query vector with only one term is not useful in determining relevant results, a search with too many keywords reduces the similarity between documents and gives a low volume of results. A query vector with a couple of keywords, which express the required information, is needed for effective relevant searching.

This work also provides a good understanding of the LSI method, particularly the dimension reduction stage, which is represented by the SVD algorithm.

5.4 Basis for Mathematical Model

The work presented in this paper describes and defines the nature of the noise in text processing. In addition the various types of noise in image processing are reviewed, and the different mathematical modeling techniques for the noise are studied as well. This basic understanding will be used as a base of future work on finding a mathematical model for the lexical noise, in order to facilitate efficient searching within the LSI system.

For example, let consider the Memos example presented in this paper. If *gaussian* and *salt−and−pepper* are added to the TDM of Memos database, and the signal to noise ratio (SNR) tool is applied to measure the noise in the generated TDMs. The LSI search is applied for a certain query. The results of this process are provided in Table 4.

Table 4. Memos Database: Searching "graph theory".

	Documents Returned	SNR value
Original TDM	4	0.9808
TDM with Gaussian	2	1.1137
TDM with Salt-and-Pepper	0	1.0591

The Table shows that, the TDMs with additive *gaussian* and *salt − and − pepper* have a higher SNR values than the original TDM, which indicates that the noise has been reduced. However, the LSI search with *salt − and − pepper* does not produce any result, while with the *gaussian* TDM returns two documents for the search query. This may be due to the fact that the lexical noise in the TDM more closely matches the distribution of *gaussian*. A full investigation on producing a mathematical model for noise in LSI is required, and forms an interesting search direction.

6 Conclusion

The investigation of TDM modeling shows that, those databases with more entries and longer document titles produce better search results and supports the use of the LSI system. This result arises as larger

databases and longer document descriptions increase the keywords appearing in more than a couple of documents, which will improve the semantic meaning in the documents set. In turn, this leads to a good distribution of the non-zero values in the TDM. Consequently, this will reduce the sparseness (lexical noise) in the TDM and allow the SVD algorithm in the LSI process to show better performance. It can be suggested that, there is a real need for seminal work geared to producing a mathematical model for noise in LSI.

References

[1] **K. Bharat** and **A. Broder,** "A technique for measuring the relative size and overlap of public web search engines," *Proceedings of the 7th International Conference on World Wide Web 7, Brisbane, Australia*, pp: 379-388, (1998).

[2] **S. Lawrence** and **C. Giles**, "Searching the world wide web," *Science*, **280**: 98-100, (1998).

[3] **M. Berry, S. Dumais**, and **G. OBrien**, "Using linear algebra for intelligent information retrieval," *SIAM Review*, **37**: 573-595, (1995).

[4] **G. W. Furnas, T. K. Landauer, L.M. Gomez**, and **S.T. Dumais**, "The vocabulary problem in human-system communication," *Communications of the ACM*, **30**: 964-971, (1987).

[5] **S. Deerwester, S. Dumais, T. Landauer, G. Furnas**, and **R. Harshman**, "Indexing by latent semantic analysis," *Journal of the Society for Information Science*, **41**: 391-407, (1990).

[6] **T. Jaber, A. Amira**, and **P. Milligan**, "Tdm modeling and evaluation of different domain transforms for lsi," *Elsevier Neurocomputing*, **72**: 2406-2417, (2009).

[7] **T. Jaber, A. Amira**, and **P. Milligan**, "Latent semantic indexing using multiresolution analysis," *International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS). Algarve, Portugal*, (2011).

[8] **T. Jaber, A. Amira**, and **P. Milligan**, "Textual noise analysis and removal for effective search engines," *European Workshop on Visual Information Processing (EUVIP), Paris, France*, pp: 129-133, (2010).

[9] **C. Fox**, "Lexical analysis and stoplists. in information retrieval – data structures & algorithm," *Prentice-Hall*, pp: 102-130, (1992).

[10] **W. B. Frakes**, "Stemming algorithms in information retrieval – data structures & algorithm," *Prentice-Hall*, pp: 131-160, (1992).

[11] **D. Hull**, "Stemming algorithms - a case study for detailed evaluation," *Journal of the American Society for Information Science*, **47**: 70-84, (1996).

[12] **S. Dumais**, "Improving the retrieval of information from external sources," *Behavior Research Methods, Instruments and Computers*, **23**: 229-236, (1991).

[13] **T. Kolda** and **D. O'Leary**, "A semi-discrete matrix decomposition for latent semantic indexing in information retrieval," *ACM Transactions on Information Systems*, **16**: 322-346, (1998).

[14] **E. Hoenkamp**, "Unitary operators on the document space source," *Journal of the American Society for Information Science and Technology*, **54**: 314-320, (2003).

[15] **Cochrane**, "Url: http://www.cochrane.org", (2005).

[16] **J. Gao** and **J. Zang**, "Clustered svd strategies in latent semantic indexing," *Laboratory for High Performance Scientific Computing andComputer Simulation University of Kentucky*, (2004).

[17] **A. Kontostathis**, "Essential dimensions of latent semantic indexing (lsi)," *Proceedings of the* 40*th Hawaii International Conference on System Sciences-2007*, pp: 73-80, (2007).

[18] **T. K. Landauer, D. Laham**, and **P. Foltz**, "Learning human-like knowledge by singular value decomposition," *Proceedings of the* 10*th Conference on Advances in Neural Information Processing Systems*, pp: 45-51, (1997).

[19] **T. A. Letsche** and **M. W. Berry**, "Large-scale information retrieval with latent semantic indexing," *Information Sciences: International Journal*, **100**: 105-137, (1997).

[20] **R. Zhao** and **W. I. Grosky**, "Narrowing the semantic gap-improved textbased web document retrieval using visual features," *IEEE Transactions On Multimedia*, **4**: 189-200, (2002).

[21] **H. Ito** and **H. Koshimizu**, "Keyword and face image retrieval based on latent semantic indexing," *IEEE lntemational Conference on Systems, Man and Cybernetics*, **1**: 358-363, (2004).

[22] **J. Yu**, "Singular value decomposition with applications to ir and text clustering," *Technical report, School of Electronics Engineering and Computer Science Peking University, Beijing*, (2003).

[23] **E. R. Jessup** and **J. H. Martin**, "Taking a new look at the latent semantic analysis approach to information retrieval," *Computational Information Retrieval*, pp: 121-144, (2001).

[24] **G. Metherall**, "Local segmentation of images," *School of Computer Science and Software Engineering, Monash University. Project URL: http://www.csse.monash.edu.au/hons/ projects/2000/Glenn.Metherall/index.html*, (2000).

[25] **R. Gonzalez** and **R. Woods**, "Digital image processing," *Second Edition, Addison Wesley*, (2002).

[26] "Cviptools," *Southern Illinois University at Edwardsville URL: http://www.ee.siue.edu/ CVIPtools/index.php*, (2008).

[27] **T. Jaber, A. Amira**, and **P. Milligan**, "A novel approach for lexical noise analysis and measurement in intelligent information retrieval," *Proceedings of IEEE International Conference on Pattern Recognition*, *ICPR. Hong Kong*, **3**: 370-373, (2006).

تحليل الضوضاء النصية وإزالتها في محركات البحث الذكية

طارق جبر إبراهيم جبر

كلية الحاسبات وتقنية المعلومات، جامعة الملك عبدالعزيز، جـــدة – المملكة العربية السعودية

المستخلص. في حقل استرجاع المعلومات الذكية (IR)، الفهرسة الدلالي الكامن (LSI) هي تقنية شعبية تستخدم لاسترجاع المعلومات المترابطة في المعنى المعجمي أكثر من المتطابقة في النص. هذه التقنية تتغلب على المشاكل المرتبطة مع الترادف وتعدد المعاني (الأسباب الشائعة لعدم الدقة في خوارزميات المطابقة). ومن العناصر الأساسية في العملية استخدام تجزؤ القيمة المفردة (SVD) الذي يعمل كنموذج رياضي لضوضاء المعجمية في مصفوفة الكلمة–الوثيقة (TDM). تبحث هذه الدراسة الجوانب المختلفة في LSI من وجهة نظر النمذجة وإزالة الضوضاء في معالجة الصور. نقاش وتحقيق عن النمذجة الرياضية للضوضاء المعجمية في TDM مقدم في هذه الورقة البحثية. العمل يتناول تعريفًا للضجيج في معالجة النص ويسعى لتحديد أفضل هيكل لـــ TDM. وبعبارة أخرى، التركيبة لـــ TDM التي من شأنها أن تسهل البحث بكفاءة في LSI.