Chapter 7

Cross - Language Information Retrieval Using Two Methods: LSI via SDD and LSI via SVD

Daniel Enrique Tamara Lopez,
Estefania Julieth Guevara Blanquicett and
Carlos Daniel Acosta Medina

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74171

Abstract

This chapter presents a method for the recovery of bilingual information based on semidiscrete matrix decomposition (SDD); that is, the problem of retrieving information in two languages, Spanish and English, is studied when the queries are made only in Spanish. In it, four case studies that exhibit the performance of the use of the latent semantic index (LSI) via SDD method for cross-language information retrieval (CLIR) are displayed. Concurrently, these results are compared with those obtained by applying LSI via singular value decomposition (SVD). All experiments were performed from a bilingual database, built from the gospels of the Bible, which combines documents in Spanish and English. For this, a fusion strategy was used that increases the size of the database by 10%. It was found that in terms of errors, the methods are comparable, since equal results were obtained in 58.3% of the queries made. In addition, the methods presented a success rate of at least 65% in the task of retrieving relevant information in the two languages considered.

Keywords: information retrieval, latent semantic indexing, semidiscrete decomposition, singular value decomposition, cross-language

1. Introduction

The retrieval of information (IR) is focused on the problem of finding information that is relevant for a specific query. It is common that in many fields of research such as medicine, theology, international law, mathematics, among others, there is a need to retrieve relevant information from databases that have documents in multiple languages, which makes reference to cross-language information retrieval (CLIR), whose objective is to identify useful

information in the same and other languages than the language of the queries. For example, a user could ask a question in language X (source language) to find documents in languages X, Y, Z (target languages).

Many methods have been used in IR, among them the vector model, which interprets queries and documents as vectors and information retrieval, is based on operations among them. Latent semantic indexing (LSI) is an IR method based on the vector model, which replaces a term document matrix with a sum of matrices of a particular structure. In this sense, the singular value decomposition (SVD), QR and ULV factorizations, and the semidiscrete decomposition (SDD) have been used in LSI to IR. The SDD has shown benefits in saving storage of large databases, but has not been tested in CLIR. This chapter examines and evaluates a method for bilingual retrieving information (Spanish-English) based on semidiscrete decomposition (SDD), which retrieves relevant information in both languages when the query is made in Spanish. In addition, are presented four case studies that show the performance of the LSI via SDD method for CLIR and the results are compared with those obtained by applying the LSI via SVD method. To do this, a database was built combining documents (Bible Gospels) in Spanish and English.

2. Vector model

One of the most common methods in text mining for automatic indexing is the vector model. In it, every document and any need for information or query is encoded as a vector whose components reflect the importance of a particular term in its meaning or semantics.

2.1. Matrix term document

A database containing n documents described by m terms is represented as a matrix $A_{m\times n}$ called *matrix term document*, where the element a_{ij} denotes the *weight* of term i in document j. A natural choice for the components of vector document is a function of the frequency with which each term occurs in it, that is to say, $a_{ij} = f_{ij}$, where f_{ij} is the number of times the term i appears in the document j. There are more sophisticated schemes such as those given in [1, 2] that may lead to better results, and in general, as is done in [2], the procedure is to define the inputs from A to

$$a_{ij} = l_{ij} g_i d_j, \tag{1}$$

where l_{ij} is the *local weight* of term i in document j, g_i is the *overall weight* of term i in the collection of documents, and d_j is the *component standardization*, which specifies whether the columns of A (that is, the documents) are normalized or not. Local and global weights are applied to increase or decrease the importance of terms within or between documents. In [1], they explain the weight scheme for terms that are recommended to use depending on the characteristics of the document collection. In the language of the vector model, the columns

and rows of A are called document vectors and term vectors, respectively. Because each vector document contains only a small part of the totality of terms that describe the entire collection of documents, normally, the term document matrix is *sparse*; i.e., most of its entries are zero.

2.2. Latent semantic indexing

Latent semantic indexing (LSI) [3–5], also called latent semantic analysis (LSA) is an automatic indexing method based on the semantics of documents, which attempts to overcome the two main problems that have the traditional indexing schemes of lexical coincidences: *polysemy* and *synonymy.* The first has to do with a word that can have multiple meanings, and therefore, the words of a query may not coincide in meaning with those of the documents; the second means that several terms can have the same meaning and hence the words used in queries can match nonrelevant documents.

LSI is based on the assumption that there is some latent semantic structure underlying data that is corrupted by the variety of words used [4], but this semantic structure can be discovered and enhanced by approximating the matrix term document by a summation of matrices of particular structure, for example, by a low rank approximation obtained by some matrix decomposition.

2.3. Queries and measures of performance

In the vector model, the queries are also seen as vectors and then match a query q means finding in the column space of A (the subspace generated by the vectors documents) the documents a_j that are most similar to her in meaning. In [2], they explain that it is possible to associate a weight scheme with a query and have

$$q_k = l_k g_k, \tag{2}$$

where q_k is the kth input of q, and l_k and g_k are the components of local and global weight, respectively. The documents considered as relevant are those that are geometrically closer to the query according to some measure, and often the cosine of the angle between the query vector and each of the document vectors is used as a measure of similarity, so that the largest values correspond to the most relevant documents. Then, a_j is recovered if

$$\cos\left(\theta\left(q, a_j\right)\right) = \frac{q^t a_j}{\|q\|_2 \|a\|_j} > tol, \tag{3}$$

where *tol* is a predefined tolerance. Another commonly used measure of similarity is the *dot product* between query vector and each document vector that is computed as

$$p = q^t A, \tag{4}$$

where the ith entry of p represents the score of the document i. Thus, the documents can be organized from major to minor, by relevance to the consultation, according to their score.

In [6], they describe that a good process of matching queries is when the intersection between the set of documents retrieved and the set of relevant documents is as large as possible and the number of irrelevant documents recovered is small. In this way, to measure the performance of an information retrieval system, one must evaluate the ability of the system to retrieve relevant information (*recall*) and to reduce irrelevant information (*precision*).

Other measures frequently used to evaluate the quality of an IR system are the *pseudoprecision, average pseudoprecision,* and *mean average pseudoprecision (MAP).* Let r_i be the number of relevant documents retrieved up to position i in the sorted list of documents:

- The recall for the ith document from the list, R_i, is the ratio of relevant documents seen so far, that is, $R_i = r_i/r_n$, where r_n is the amount of relevant documents retrieved.
- The precision for the i-th document, P_i, is the proportion of documents up to position i that are relevant to a given query, that is, $P_i = r_i/i$.
- The *pseudoprecision* for a level of recall x, $\tilde{P}(x)$, is defined by

$$\tilde{P}(x) = \max P_i, \quad where\ x \leq \frac{r_i}{r_n}, i = 1, 2, \cdots, n. \tag{5}$$

- The *average pseudoprecision* for a single query is defined by

$$P_{av} = \frac{1}{n}\sum_{i=0}^{n-1} \tilde{P}\left(\frac{i}{n-1}\right). \tag{6}$$

- The *mean average pseudoprecision (MAP)*, used to evaluate yield in a set of queries, is defined by

$$MAP = \frac{1}{M}\sum_{j=1}^{M}\left[\frac{1}{n}\sum_{i=0}^{n-1} \tilde{P}\left(\frac{i}{n-1}\right)\right], \tag{7}$$

where M is the number of queries.

3. Latent semantic indexing via singular value decomposition

In this section, we initially present the singular value decomposition (SVD) and two theorems that show how the SVD gives useful information about the structure of a matrix. Also, it is explained why these theorems are important for IR and in particular for LSI. Subsequently, the LSI method based on the SVD is exposed.

3.1. Singular value decomposition

The SVD of a matrix $A_{m\times n}$, with $m \geq n$, is a factorization of the form

http://dx.doi.org/10.5772/intechopen.74171

$$A = U\begin{pmatrix}\Sigma\\0\end{pmatrix}V^t, \tag{8}$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices whose columns are called, respectively, singular vectors to the left and to the right of A, and $\Sigma \in \mathbb{R}^{n\times n}$ is a diagonal matrix that contains the singular values $\sigma_1 \geq \sigma_2 \geq \sigma_n \geq, \cdots, \geq 0$ of A in decreasing order within its diagonal. This factorization exists for any matrix A, and numerical linear algebra texts commonly include it in their content [7, 8]. Methods to calculate the SVD of *dens*e and *sparse* matrices are well documented [1, 6, 7].

The following two theorems show how the SVD reveals important information about the structure of a matrix.

Theorem 1. Let $A_{m\times n}$, where without loss of generality $m \geq n$, $A = U\Sigma V^t$, the *SVD* of A and $\sigma_1 \geq \sigma_2 \geq, \cdots, \geq \sigma_r > \sigma_{r+1} =, \cdots, = 0$. If *R (A)* and *N (A)* denote the column space and null space of *A,* respectively, and if $U = [u_1\ u_2, \cdots, u_m]$ and $V = [v_1\ v_2, \cdots, v_n]$, then,

- *rank (A) = r*
- $R(A) = span\{u_1,\ u_2, \ldots, u_r\}$
- $N(A) = span\{v_{r+1},\ v_{r+2}, \ldots, v_n\}$
- $R\left(A^t\right) = span\{v_1,\ v_2, \ldots, v_r\}$
- $N\left(A^t\right) = span\{u_{r+1},\ u_{r+2}, \ldots, u_n\}$

$$A = \sum_{i=1}^{r} u_i\sigma_i v_i^t$$

Proof: See [6–8].

The theorem reveals that the SVD gives orthogonal bases for the four fundamental subspaces associated with a matrix and, in particular, in the context of term document matrices, the second part indicates that generating the semantic content of a database does not require to use all document vectors but a subset of the singular vectors to the left corresponding to the range of the matrix. The sum of the last part of the theorem is usually called *expansion in singular values* of *A*.

Theorem 2 (Eckart and Young). Suppose $A \in \mathbb{R}^{m\times n}$ has rank $r > k$. Then,

$$\min_{\text{rank}(B)} ||A - B||_f^2 = ||A - A_k||_f^2, \tag{9}$$

where

$$A_k = \sum_{i=1}^{k} u_i\sigma_i v_i^t := U_k\Sigma_k V_k^t. \tag{10}$$

Proof: See [6–8].

In this case, the theorem states that A_k is the matrix of rank k closest to A. The U_k columns live in the semantic space and are used to approximate the documents. As is known, truncated SVD is useful for "*eliminating noise*" present in an array and therefore, in the case of matrices representing a database, to remove term-document associations that are obscuring the real meaning of it.

3.2. LSI via SVD

As mentioned earlier, LSI is an IR method based on the vector model that approximates a document term matrix by a sum of the matrices of particular structure. In this regard, according to Theorem 2, LSI via SVD uses the singular value decomposition to obtain a k-rank approximation of the original document term matrix $A_{m\times n}$ in order to eliminate the noise present in it and project the m terms, n documents, and queries in a k-dimensional space, where $k \ll \min(m, n)$. It is important to keep in mind that document term matrices are commonly *well conditional*, that is, their singular values have no gaps and do not decay rapidly to zero, so a suitable k to truncate the SVD cannot be estimated [6]; experimentally, it has been concluded that for very large databases, k is taken between [100; 300] [3].

As in the vector model, in LSI, it is possible to match a query q through operations between vectors, namely, the cosine of the angle or the product point between the query vector and the document vectors. In this case, we calculate

$$p = \tilde{q}^t \tilde{A}, \tag{11}$$

where $\tilde{q} = U_k^t q$, $\tilde{A} = \Sigma_k V_k^t$ and $A_k = \mathrm{U_k}\Sigma_k V_k^t$ is the approximation of rank k of A obtained from the truncated SVD. Therefore, the recovered documents will be those corresponding to the largest components of p.

4. Latent semantic indexing via semidiscrete matrix decomposition

In this section, we present the semidiscrete decomposition (SDD) of a matrix and the method LSI via SDD. For more details, see [2, 9].

4.1. Semidiscrete decomposition

A semidiscrete decomposition (SDD) expresses a matrix as a weighted sum of outer products formed by vectors whose inputs are taken from the set $\mathrm{S} = \{-1, 0, 1\}$ that is given as

$$A_k = \underbrace{[x_1\, x_2 \cdots x_k]}_{X_k} \underbrace{\begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_k \end{bmatrix}}_{D_k} \underbrace{\begin{bmatrix} y_1^t \\ y_2^t \\ \vdots \\ y_k^t \end{bmatrix}}_{Y_k^t} = \sum_{i=1}^{k} d_i x_i y_i^t, \tag{12}$$

http://dx.doi.org/10.5772/intechopen.74171

where the $x_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}^n$ are formed by elements of the set $S = \{-1, 0, 1\}$, and d_i is a positive scalar, called the *i*-th SDD value. The matrix A_k is called semidiscrete decomposition of rank *k* (or SDD *k*-term). The algorithm that allows to calculate the SDD of a matrix and some of its properties, for example, its convergence, is described in [9].

4.2. LSI via SDD

The truncated SVD produces the best approximation of range *k* for a matrix; however, generally, even for a very low range approximation, more storage is required than the original matrix if it is *sparse*, that is, if the majority of its components are zero. As document term matrices that correspond to real databases are commonly large and sparse, using truncated SVD can be extremely expensive in terms of storage. It is for the above, that to save space (and consultation time) in [2], they propose the SDD as an alternative of SVD in LSI.

In this sense, LSI via SDD consists of replacing the term document matrix by an approximation that allows, as they sign in [10], to identify the clusters that form the documents present in databases and at the same time save a considerable amount of storage with respect to other factorizations. In [2], they show that for equal values of *k*, the SVD requires about 32 times more storage than the SDD. Specifically, LSI via SDD consists of approximating the document term matrix by a sum of exterior products of rank 1 such as in the SVD, but whose vectors consist only of elements of the set $S = \{-1, 0, 1\}$. For more details on LSI via SDD, see [2, 11].

To match the queries with the documents using LSI via SDD, we proceed in the same way as in LSI via SVD, that is, by calculating the product

$$p = \tilde{q}^t \tilde{A}, \tag{13}$$

where $\tilde{q} = X_k^t q$, and $\tilde{A} = D_k Y_k^t$. Relevant documents are those that correspond to the largest components of *p*.

5. The CLIR problem

Different documents can contain information that is conceptually the same without having to use similar words. People when they make a query in an IR system, for example, a search engine such as Google, do so by concept, and the words they use in it generally do not match those of the relevant documents. In this way, the main objective of CLIR, which is the retrieval of relevant information in the same and other languages to the queries, is highly affected because they must be compared terms of different languages.

To address this situation, databases have been created between languages, which are collections of documents that combine low percentages of languages and for its construction, it is necessary to take into account two concepts of close relationship with CLIR: *parallel aligned corpus* and *fusion strategies*.

5.1. Parallel aligned corpus

A *parallel text* is a text accompanied by its translations in other languages. Large collections of parallel text are called *parallel corpus*. In order to use a parallel corpus correctly, it is necessary to align the original text with its (your) translation (translations), that is, you must identify the phrases or words in the original text with their corresponding translations in the other languages. This is known as the *parallel aligned corpus*.

As stated by Kolda et al. in [12], perhaps the biggest decision to make when implementing LSI multilanguage is which parallel aligned corpus to use. In this work, we have adopted the Bible as ours and reasons for this are: (i) it is probably the most translated book in the whole world, which allows us to have many translations of the same documents, (ii) given its presentation by chapters and verses, its parallel alignment is facilitated, (iii) if we refer to the Gospels (Matthew, Mark, Luke, John), it is easy to identify facts related to the life of Jesus and thus recognize relevant documents for queries made in this context.

5.2. Fusion strategies

The central purpose in CLIR is to develop tools that allow the terms of query to coincide with those of documents that describe the same or similar meaning, even if they are in different languages [13]. The goal is the construction of parallel aligned corpus using the languages of the documents, which can be done, for the case of two languages, for example, taking portions of documents in a certain language and adding them to the corresponding documents of the other language. This is called a *fusion strategy*. Related works are [14, 15].

This work seeks to recover relevant documents in Spanish and English when queries are made in Spanish using fusion strategies, which combine approximately 10% of documents. The central idea behind each fusion used is to take a specific amount of verses in a certain language and add them to the corresponding verses of the other language.

6. Study cases

Four case studies are developed with the intention of evaluating the performance of two methods of LSI, LSI via SVD and LSI via SDD, applied in CLIR. The first one identifies the LSI model that allows obtaining the best results in terms of the mean average pseudoprecision (MAP). For this case, two English translations of the Gospels were used: *The King James* and *New Living Bible*. In the second case, we start with the model previously chosen to develop two experiments that involve two fusion strategies that combine small portions of the Gospels in the English-Spanish languages, using the *King James* and *Reina Valera 1966* versions. In Cases 3 and 4, computational comparisons are made between the LSI methods and their performance are analyzed when the collection of documents increases, respectively.

In all cases, the documents used to consist of a group of verses that form a story, verses which were taken from the *New International* version of the Bible, which organizes the verses by

stories and gives each one a title. The queries used are those given in **Table 1**, which describe parables and miracles in the life of Jesus. **Table 2** shows the biblical quotation where each of these queries is located, that is, the documents that are relevant.

6.1. Case 1: identification of the LSI model

An *LSI model* is the set of parameters that are considered in the application of the latent semantic index method, that is, the local and global weight schemes, the number of factors (k), the fusion strategies, etc., that are chosen for performing recovery experiments. A four-letter string has been used to differentiate LSI models. The first three indicate the local weight, the global weight and the use of standardization in the matrix term document, respectively, and the last corresponds to the query matrix and refers to the local weight of the terms. In this way, the nomenclature *fex.l*, for example, means that in the matrix term document, the frequency (f) for the local weight and an entropy value (e) (see [1]) for the global weight of the terms were used; besides, the columns of the matrix document term were not normalized (x) and a logarithmic value (l) was used for the local weight of the query terms.

Query		Query		Query	
1	El bautizo de Jesús	5	Niño epiléptico curado	9	Vino nuevo viejo odres
2	Impuesto al Cesar	6	La alimentación a cinco mil	10	El sembrador y la tierra
3	Limpieza al templo	7	La higuera maldita	11	Grano de mostaza
4	Entrada a Jerusalén	8	Tela nueva vestido Viejo	12	Higuera

Table 1. Queries used for the case studies.

Query	Matthew	Mark	Luke	John
1	Mt 3:13-17	Mc 1:9-11	Lc 3:21-23	Jn 1:29-39
2	Mt 22:15-22	Mc 12:13-17	Lc 20:20-26	
3	Mt 21:12-13	Mc 11:12-14		Jn 2:14-22
4	Mt 21:1-11	Mc 11:1-10	Lc 19:29-44	Jn 12:12-19
5	Mt 17:14-18	Mc 9:17-29	Lc 9:38-43	
6	Mt 14:15-21	Mc 6:35-44	Lc 9:12-17	Jn 6:5-13
7	Mt 21:18-22	Mc 11:12-14,20-25		
8	Mt 9:16	Mc 2:21	Lc 5:36	
9	Mt 9:17	Mc 2:22	Lc 5:37-38	
10	Mt 13:3-8,18-23	Mc 4:3-8,14-20	Lc 8:5-8,11-15	
11	Mt 13:31-32	Mc 4:30-32	Lc 13:18-19	
12	Mt 24:32-35	Mc 13:28-29	Lc 21:29-31	

Table 2. Location of queries in the gospels.

In this case, different LSI models are tested and the best one is determined from the MAP for $k = 100$. **Table 3** reports the results.

It is observed that with both methods, that is, LSI via SVD and LSI via SDD, the highest values of the MAP, marked in bold, were achieved with the models *len.f*, *len.b*, and *len.l*. This means that the *log-entropi* scheme is the one with the best performance and that the local weight of the terms in the query matrix does not affect the quality of a recovery. For this reason, in all subsequent experiments, only the *len.f* model will be used in both methods.

6.2. Case 2: fusion strategies

Two experiments are developed that involve merging the documents in English with their corresponding versions in Spanish. In each one, a different fusion strategy is used and 20 documents are retrieved by query. For each query in each experiment, an analysis of the selection of the k is made in order to establish a margin for the choice of the same. The errors obtained are illustrated in terms of the average of pseudoprecision and tables that give details of what was recovered in each query are shown. The errors were calculated with the formula

$$Error = 1 - \frac{1}{n}\sum_{i=0}^{n-1} \tilde{P}\left(\frac{i}{n-1}\right). \tag{14}$$

The database has 670 documents, of which, considering the 12 queries, 72 are relevant, that is, only 10.74% of the collection is relevant. The amount of storage required is 0.375 MB.

6.2.1. Experiment 1

The fusion strategy increases the size of the database by approximately 10% and consisted of *taking a single verse from the beginning of each document in Spanish and adding it to the end of the corresponding first verse in English*. **Table 4** illustrates the structure of the database and one of its documents.

In **Figure 1**, we show the graphs that relate the k to the error levels for each of the queries. It is observed that the error curves give clues for the selection of a k in almost all the queries. In the

LSI model	SVD	SDD	LSI model	SVD	SDD
cxn.f	0.6948	0.6440	**len.l**	**0.7591**	**0.7035**
fxn.f	0.6762	0.5920	lin.f	0.7375	0.6811
lxn.f	0.7444	0.6276	lin.b	0.7375	0.6811
lxn.b	0.7444	0.6276	lin.l	0.7375	0.6811
lxn.l	0.7444	0.6276	lpn.f	0.7058	0.6957
len.f	**0.7591**	**0.7035**	lpn.b	0.7058	0.6957
len.b	**0.7591**	**0.7035**	lpn.l	0.7058	0.6957

Table 3. MAP for different LSI models.

http://dx.doi.org/10.5772/intechopen.74171

Gospels	Doc. English	Doc. Spanish	Example of a database document
Matthew	Eng + Spa	Spanish	And seeing the multitudes, he went up into a mountain: and when he was set, his disciples came unto him: **Viendo la multitud, subio al monte; y sentandose, vinieron a el sus discipulos.** And he opened his mouth, and taught them, saying,
Mark	Eng + Spa	Spanish	
Luke	Eng + Spa	Spanish	
John	Eng + Spa	Spanish	

Table 4. Fusion scheme of the database (left) and example of a document of it (right).

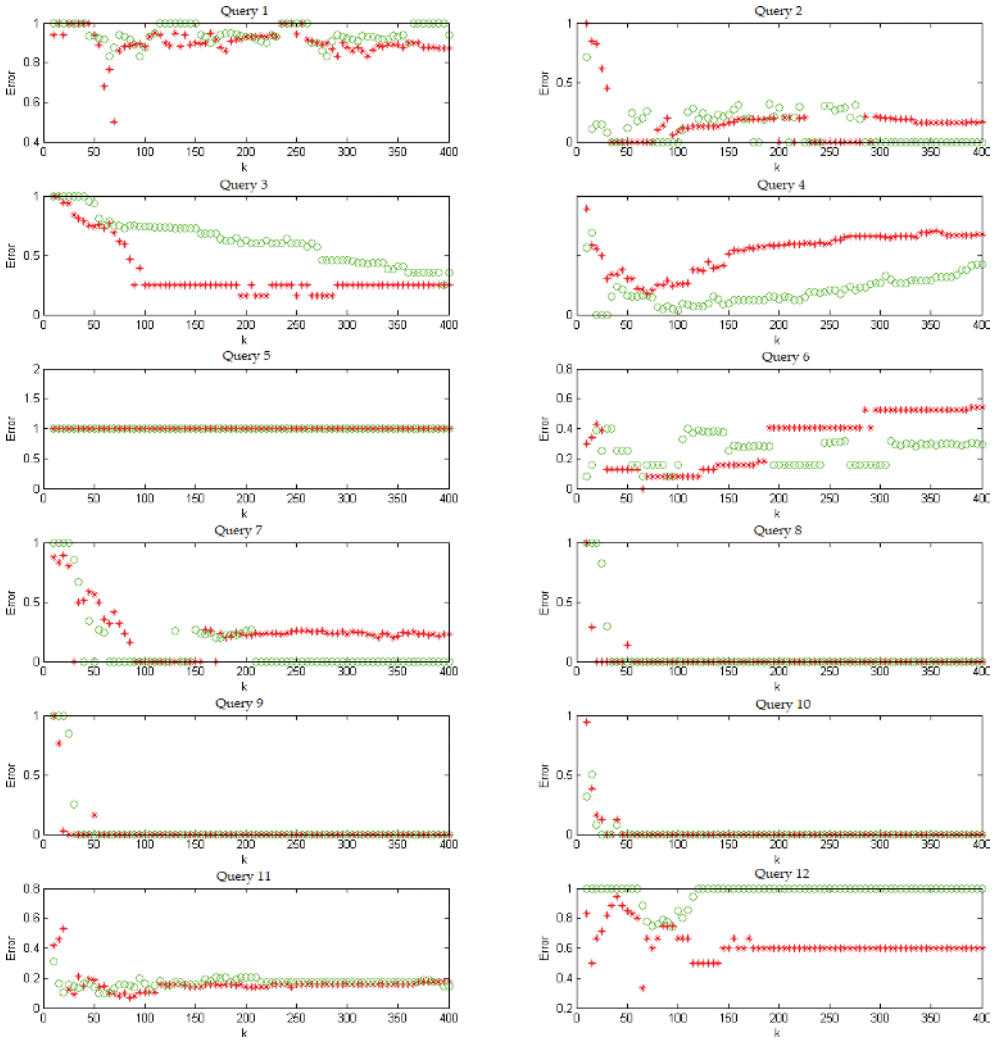

Figure 1. Errors versus k for each query. The asterisks and circles indicate the methods LSI via SVD and LSI via SDD, respectively.

first one, for example, it is observed that $k = 70$ would be the *optimum k* for LSI via SVD. In Query 5, the two methods completely failed with a 100% error; in Queries 8, 9, and 10, the errors are approximately zero in almost all values of k. There is good behavior of the methods in Queries 2, 4, 6, 7, 8, 9, 10, and 11 in many values of k, in particular, for some less than 110. In Queries 1 and 12, the best results were also in these values. It is also observed that there are usually many local minima, which makes it difficult to automate the choice of k through some parameter selection algorithm.

In **Table 5**, we show for each query two values of k and the corresponding errors. The first, called *optimal k*, indicates the smallest k for which the smallest error was obtained. The other represents the same but considers $k < 110$. It is noted that in all queries, except q3, the *optimum k* matches the *selected k*, which leads us to think about the possibility of reducing the domain of choice of k when considering the *k selected* in an interval of lower amplitude. **Table 6** shows analogous results to those in **Table 6** considering the values of the *k selected* in the interval [70, 100].

	SVD		SDD		SVD		SDD		SVD		SDD
q1	^{k}*opt Err*	70	65	**q5**	^{k}*opt Err*	10	10	**q9**	^{k}*opt Err*	25	35
	^{k}*sel Err*	0.5	0.833		^{k}*sel Err*	1	1		^{k}*sel Err*	0	0
		70	65			10	10			25	35
		0.5	0.833			1	1			0	0
q2	^{k}*opt Err*	35	35	**q6**	^{k}*opt Err*	65	10	**q10**	^{k}*opt Err*	30	25
	^{k}*sel Err*	0	0		^{k}*sel Err*	0	0.078		^{k}*sel Err*	0	0
		35	35			65	10			30	25
		0	0			0	0.078			0	0
q3	^{k}*opt Err*	195	395	**q7**	^{k}*opt Err*	30	40	**q11**	^{k}*opt Err*	85	55
	^{k}*sel Err*	0.157	0.25		^{k}*sel Err*	0	0		^{k}*sel Err*	0.065	0.097
		90	80			30	40			85	55
		0.25	0.727			0	0			0.065	0.097
q4	^{k}*opt Err*	70	20	**q8**	^{k}*opt Err*	20	35	**q12**	^{k}*opt Err*	65	95
	^{k}*sel Err*	0.173	0		^{k}*sel Err*	0	0		^{k}*sel Err*	0.33	0.74
		70	20			20	35			65	95
		0.173	0			0	0			0.33	0.74

Table 5. Fusion 1. Query errors for the optimal k and the selected k.

	SVD		SDD		SVD		SDD		SVD	SDD
q1	^{k}*sel Error*	70	95	**q5**	^{k}*sel Error*	70	70	**q9**	^{k}*sel Error*	70
		0.5	0.83			1	1			0.5
q2	^{k}*sel Error*	70	75	**q6**	^{k}*sel Error*	70	90	**q10**	^{k}*sel Error*	70
		0	0			0.07	0.07			0
q3	^{k}*sel Error*	90	80	**q7**	^{k}*sel Error*	90	70	**q11**	^{k}*sel Error*	90
		0.25	0.72			0	0			0.25
q4	^{k}*sel Error*	70	100	**q8**	^{k}*sel Error*	70	70	**q12**	^{k}*sel Error*	70
		0.17	0.03			0	0			0.17

Table 6. Fusion 1. Errors per query for the selected k in [70, 100].

Here, it is appreciated that in all queries, except for q3 and q12, the *selected k* increased; however, the errors in q1, q2, q5, q6, q7, q8, q9, and q10 remained the same. In q4 and q11, the errors increased from 0 to 3% and from 9 to 12%, respectively.

From the above, it is concluded that the performance of the LSI methods subtly deteriorated when considering k in such interval, since in 10 of the 12 queries, the errors were maintained and in only two they increased in small percentages. The main contribution of these tables is to have identified a small range for the choice of the parameter k.

6.2.2. Experiment 2

The fusion strategy used in this case also combines a single verse, that is, 10% of the documents, but unlike the previous experiment, it takes verses in English and adds them to the corresponding verses in Spanish and vice versa. The structure of the database is illustrated in **Table 7**.

Figure 2 illustrates the error curves for each query by increasing the parameter k.

Again it is observed that at q5, an error of 100% was obtained. In q2, q6, q7, q8, q9, and q10, the methods reached errors close to zero in some values of k. In q1 and q11, only LSI via SDD obtained errors close to that value. Again, the existence of many local minimums in the error levels of each query is highlighted. Information on the *optimal k*, the *selected k*, the *k selected in the interval* [70, 100] (denoted by k_{sel1} and k_{sel2}, respectively) and the corresponding errors is given in **Table 8**.

We find that in q2, q5, q6, q7, q8, q9, q10, q11, and q12, the errors for k_{opt}, k_{sel1}, and k_{sel2} did not change when using LSI via SVD, that is, for this group of nine queries, the *optimal k* lies in the interval [70, 100]. With LSI via SDD, for this same group of queries, except for q2 and q11, the k_{opt} is also obtained in that interval; in q3, the errors increased when the selection interval of

		Chapters	
Matthew	Eng + Spa	1–14	Spanish
	English	15–28	Spa + Eng
Mark	Eng + Spa	1–8	Spanish
	English	9–16	Spa + Eng
Luke	Eng + Spa	1–12	Spanish
	English	13–24	Spa + Eng
John	Eng + Spa	1–10	Spanish
	English	11–21	Spa + Eng

Table 7. Scheme of the database for the structure of the merger 2.

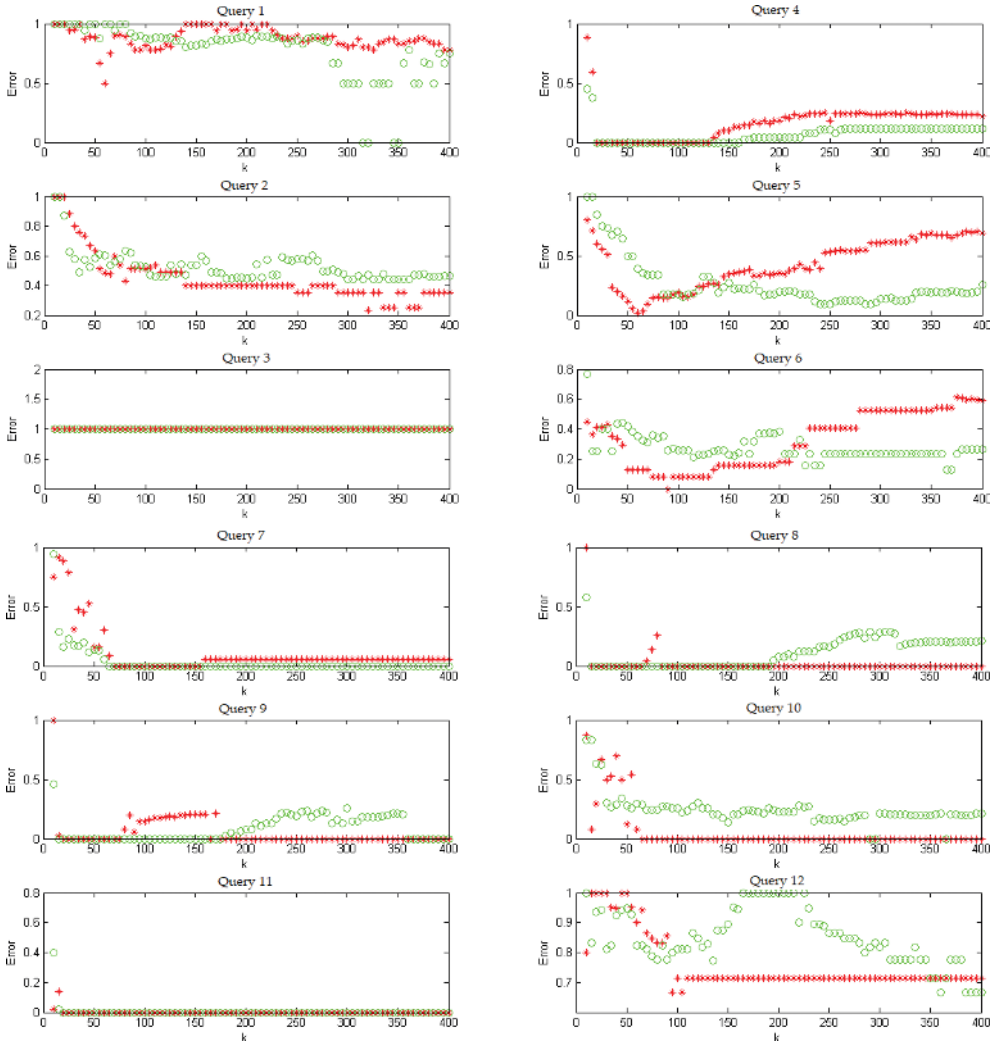

Figure 2. Errors versus k for each query. The asterisks and circles indicate the methods LSI via SVD and LSI via SDD, respectively.

the k was reduced; in q4, for the k_{sel1} and the k_{sel2}, the errors were equal but superior to the corresponding ones of the k_{opt}.

In this way, considering the two experiments, it is concluded that the results in terms of the Eq. (14) to calculate the errors in the recoveries favor the *Fusion 1* since when considering k in the interval [70, 100], LSI methods obtained minor errors compared to those found with *Fusion 2*. Therefore, in the following cases, only *Fusion 1* will be used in order to continue with the comparison of LSI methods.

	SVD		SDD		SVD		SDD		SVD		SDD
	^{k}opt	55	355		^{k}opt	10	10		^{k}opt	25	15
	Err	0.75	0		*Err*	1	1		*Err*	0	0
q1	$^{k}sel1$ *Err*1	55 0.75	65 0.87	**q5**	$^{k}sel1$ *Err*1	10 1	10 1	**q9**	$^{k}sel1$ *Err*1	25 0	15 0
	$^{k}sel2$	70	70		$^{k}sel2$	70	70		$^{k}sel2$	70	70
	*Err*2	0.83	0.89		*Err*2	1	1		*Err*2	0	0
	^{k}opt	35	15		^{k}opt	60	85		^{k}opt	25	30
	Err	0	0		*Err*	0	0		*Err*	0	0
q2	$^{k}sel1$ *Err*1	35 0	15 0	**q6**	$^{k}sel1$ *Err*1	60 0	85 0	**q10**	$^{k}sel1$ *Err*1	25 0	30 0
	$^{k}sel2$	70	80		$^{k}sel2$	70	85		$^{k}sel2$	70	70
	*Err*2	0	0.16		*Err*2	0	0		*Err*2	0	0
	^{k}opt	130	150		^{k}opt	90	40		^{k}opt	75	20
	Err	0.33	0.33		*Err*	0	0		*Err*	0.06	0
q3	$^{k}sel1$ *Err*1	110 0.49	105 0.55	**q7**	$^{k}sel1$ *Err*1	90 0	40 0	**q11**	$^{k}sel1$ *Err*1	75 0.06	20 0
	$^{k}sel2$	100	75		$^{k}sel2$	90	70		$^{k}sel2$	75	85
	*Err*2	0.56	0.61		*Err*2	0	0		*Err*2	0.06	0.25
	^{k}opt	115	150		^{k}opt	25	15		^{k}opt	25	90
	Err	0.3	0.16		*Err*	0	0		*Err*	0.33	0.5
q4	$^{k}sel1$ *Err*1	80 0.34	70 0.21	**q8**	$^{k}sel1$ *Err*1	25 0	15 0	**q12**	$^{k}sel1$ *Err*1	25 0.33	90 0.5
	$^{k}sel2$	80	70		$^{k}sel2$	70	70		$^{k}sel2$	75	90
	*Err*2	0.34	0.21		*Err*2	0	0		*Err*2	0.33	0.5

Table 8. Fusion 2. Errors per query for the optimal *k*, the selected *k*, and the selected *k* in [70, 100].

6.3. Case 3: computational comparison of LSI models

The results shown in the experiments of the second case study do not consider the efficiency of the IR systems, that is, the time of the LSI methods, the amount of storage required by each of them, the ability to quickly obtain relevant documents, and the relationship between these aspects. For this reason, in this case, computational results are presented that allow the LSI methods to be compared in such aspects. All tests were performed on a computer with *Intel (R) Core (TM) I5–3230 CPU @ 2.60 Hz* and with 6 GB of *RAM*. In **Figures 3** and **4**, the results obtained by SVD and SDD have been marked with an asterisk (*) and a circle (°), respectively.

Figure 3 illustrates the size in *megabytes* (MB) of the SVD and SDD decompositions for various values of *k*. Clearly, it is observed that in all the *k*, the SDD consumes much less space than the SVD. For $k = 400$, for example, the SVD occupies a space of 22.325 MB, while the SDD 0.875 MB, that is, there is a saving of 21.45 MB. In addition, in the lower part, the time used,

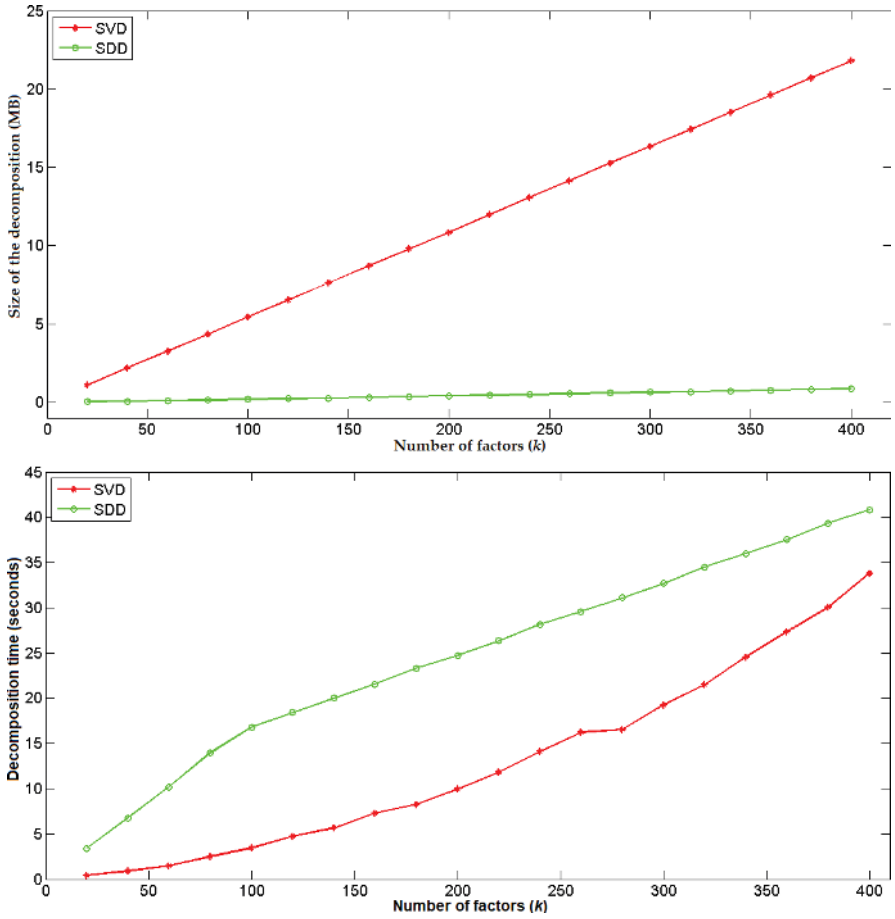

Figure 3. Size in megabytes (above) and execution time in seconds of the decompositions (below), as a function of the number of factors (*k*).

in seconds, to obtain each decomposition is shown. For the presented *k*, it is evident that in the SVD, less time is used. However, the amount of seconds used by each algorithm to build the matrices of the SVD and SDD factorizations is small, because even for high values of *k*, the recorded time is approximately 40 seconds.

Finally, in **Figure 4**, the MAP is presented as a function of the time of the LSI method, calculated using the formula *Time LSI methods = Decomposition Time + Query time*, and the amount of storage required by each one. In the graph, on the left, there are 20 asterisks corresponding to the values $k = 20, 40, \ldots, 400$ and 20 circles related to the same values of *k*. The second asterisk (corresponding to $k = 40$), for example, means that LSI via SVD required approximately 1.06 seconds to reach a quality of 0.5850, while the second circle shows that LSI via SDD at 9.21 seconds reached a MAP of 0.6062.

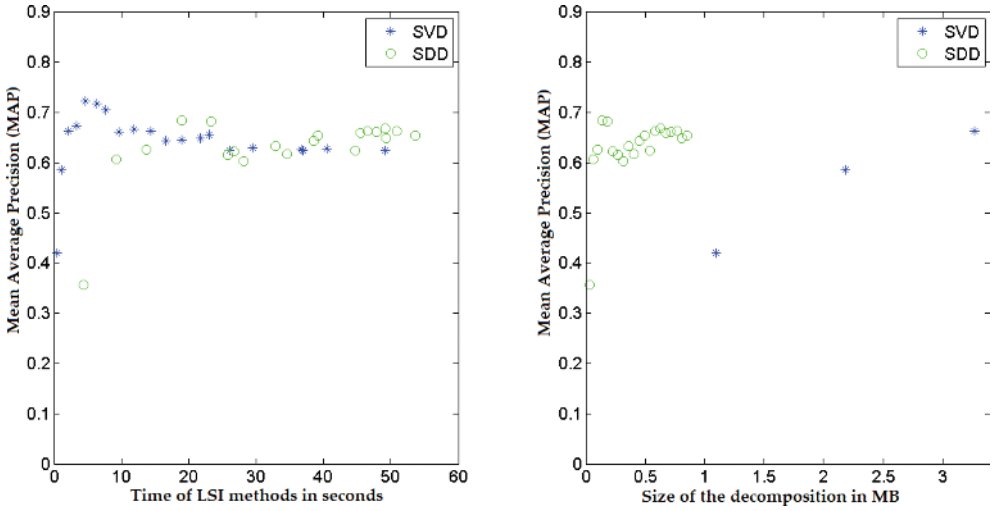

Figure 4. MAP versus time of LSI methods (left) and size of decompositions (right).

It is observed that the SVD-based method reaches its highest score, 0.7227, in $k = 100$ (at 4.7 seconds), and that the other one does it in $k = 80$ (at 18.9 seconds) with a value of 0.6838. It should be noted that the qualities of the methods, from $k = 40$ for LSI via SDD, were very close. On the right side, the size of the decompositions is crossed with the MAP. The last circle (for $k = 400$) means that a quality of 0.6539 was reached with just 0.85 MB; in turn, the first asterisk illustrates that with 1.09 MB, there was a MAP of only 0.4196. It is also observed that there are only three asterisks for SVD, and it is because the rest surpasses the scope of the graph. Likewise, it is highlighted that the highest score for the SDD-based method required only 0.14 MB of storage (approximately one-third of the weight of the matrix term document), while LSI via SVD required 5.6 MB of storage (approximately 15 times the weight of the matrix term document) to achieve its best performance. This time LSI via SDD widely outperformed the other method.

6.4. Case 4: adding documents to the data base

So far, we have only studied the LSI methods when you have a fixed document collection. In practice, it often happens that these collections are dynamic, that is, that new documents are added or that some existing ones are deleted. In this case study, the performance of the LSI methods is analyzed when different amounts of documents are added to the database. For this, the average pseudoprecision (see Eq. (9)) is used as a measure of quality to make an analysis by query and the MAP to generalize the study to all of them.

Specifically, 20 and 88 documents were added to the initial collection of 670 documents in order to obtain two new databases with 690 and 758 documents, which have 75 and 78 relevant documents, respectively. For these three collections, the results of **Figure 5** and **Table 9** are presented.

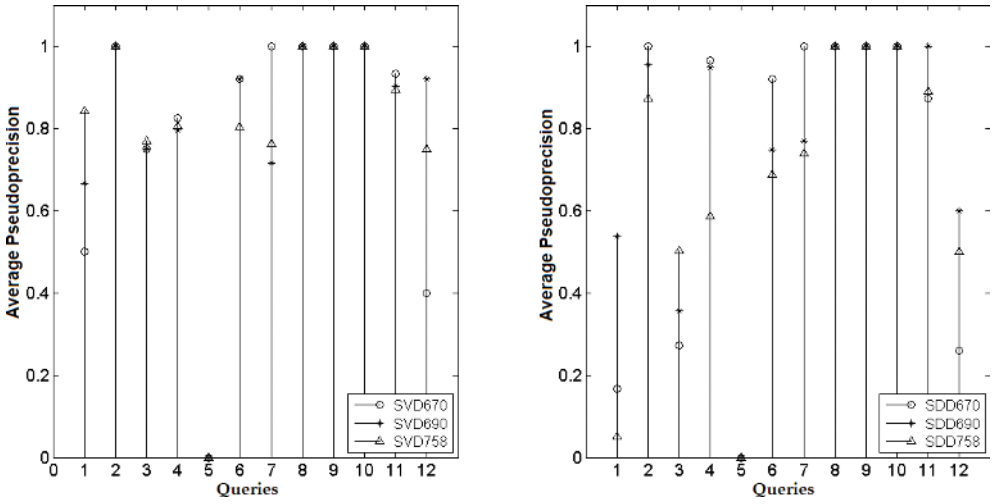

Figure 5. Comparison of the LSI methods with respect to the average pseudoprecision when adding documents to the database.

Documents			**MAP**	
Existing	Additions	% de Doc. relevant	SVD	SDD
670	—	10.74	0.7776	0.7046
670	20	10.86	0.8059	0.7428
670	88	10.29	0.8022	0.6523

Table 9. Percentage of relevant documents and MAP for the different databases.

In this figure, the average pseudoprecision obtained with the *k* in **Table 7** is shown for each of the 12 queries and for each database. The results of the LSI via SVD method are shown on the left and on the right are those corresponding to LSI via SDD. Methods for collections with 670, 690, and 758 documents have been labeled with a circle, an asterisk, and a triangle, respectively. It is observed that in the two methods, the average of pseudoprecision for Query 5 is 0 and for Queries 8, 9, and 10, it is 1. In Query 2, LSI via SVD also had a score of 1, while LSI via SDD did so only for 670 documents. In the rest of the queries, there are averages that go up or down as documents are added to the database. In Query 12, for example, it is noted that the highest score is for 690 documents, decreases when the collection is increased to 758 and decreases again when there are barely 670 documents. From this, it is concluded that there is no direct or inverse relationship between the average pseudoprecision and the number of documents in the database.

On the other hand, in order to evaluate the performance of the methods considering all the queries, in **Table 9**, the MAP obtained in each database is presented when again using the *k* of the **Table 6**.

As a first observation, it is highlighted that MAP levels are higher, in all databases, when LSI is used via SVD. However, the six scores shown give evidence of the good performance of the two methods considering the low percentages of relevant documents in each collection, since all the success rate exceeds 65%. On the other hand, it is emphasized that there seems to be a direct relationship between the percentage of relevant documents and MAP values with the SDD-based method, that is, the higher the percentage of relevant documents, the greater the MAP.

7. Conclusions

The LSI method originally used the singular values decomposition (SVD) for the benefits that it has in terms of data representation in spaces of reduced dimension and other properties with respect to data filtering. This makes the SVD a powerful tool in IR and in CLIR. The semidiscrete decomposition (SDD), of which few investigations have been developed, has been successfully used in IR, and this research has shown that it is also useful in CLIR and that it is also comparable with the standard approach used by the SVD. Evidence of this is that

- In Case 2 for Fusion 1, the errors for the k selected in the interval [70, 100] were the same in 7 of the 12 queries and in the rest, they differ at most by 47%.
- When in Case 3 efficiencies were evaluated, in aspects, particularly one method surpassed the other. Specifically, when the MAP measure was related to the time of the methods, LSI via SVD was imposed because it requires fewer seconds to reach its highest performance; in contrast, when analyzing the MAP and the amount of storage, LSI via SDD showed a significantly higher performance. In this aspect, it is emphasized that with SDD, only one-third of the weight of the original matrix was needed to reach its highest performance; with SVD, on the other hand, it required almost 15 times the weight of the matrix term document to achieve such value. This is the true impact of the SDD, the ability to obtain good results at a very low cost in terms of storage.
- The MAP quality measure evaluates the performance of an IR method considering a set of queries, so that the higher this value, the better the method's performance will have been. In the fourth case study, when the performance of the methods was considered by increasing the number of documents in the database, higher performance was obtained when using the SVD since higher MAP values were found. However, with both methods, satisfactory results were obtained, because when conducting a search in Spanish, you can retrieve relevant documents in this language and in English with a success rate of at least 65%.

Therefore, it is concluded that although the LSI via SVD method has been widely used and is a powerful tool in CLIR, the LSI via SDD method results in an important and innovative alternative in information recovery tasks, since, in addition to achieving results comparable to those of the other method in the task of retrieving relevant information in multiple languages after consulting only one, and also has the benefit of saving large amounts of space when huge databases are stored.

Author details

Daniel Enrique Tamara Lopez*, Estefania Julieth Guevara Blanquicett and Carlos Daniel Acosta Medina

*Address all correspondence to: detamaral@unal.edu.co

Universidad Nacional de Colombia, Manizales, Colombia

References

[1] Berry M, Browne M. Understanding Search Engines: Mathematical Modeling and Text Retrieval. 2nd ed. Knoxville, Tennessee: SIAM; 2005. p. 135. DOI: 10.1137/1.9780898718164

[2] Kolda T, O' Leary D. A semidiscrete matrix decomposition for latent semantic indexing in information retrieval. ACM Transactions on Information Systems (TOIS). 1998;(4):322-346. DOI: 10.1145/291128.291131

[3] Berry M, Drmac Z, Jessup E. Matrices, vector spaces, and information retrieval. SIAM Review. 1999;**41**(2):335-362. DOI: 10.1137/S0036144598347035

[4] Berry M, Dumais S, O'brien G. Using linear algebra for intelligent information retrieval. SIAM Review. 1995;**37**(4):573-595. DOI: 10.1137/1037127

[5] Deerwester S, Dumais S, Landauer T, Furnas G, Harshman R. Indexing by latent semantic analysis. Journal of the Association for Information Science and Technology. 1990;**41**(6): 391-407. DOI: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

[6] Lars E. Matrix methods in data mining and pattern recognition. Linköping, Sweden: SIAM; 2007. p. 234. DOI: 10.1137/1.9780898718867

[7] Golub G, Van Loan C. Matrix Computations. 4th ed. Baltimore: Johns Hopkins University Press; 2012. 780 p. DOI: 10.1137/1028073

[8] Ipsen I. Numerical matrix analysis: Linear systems and least squares. Raleigh, North Carolina: SIAM; 2009. p. 140. DOI: 10.1137/1.9780898717686

[9] Kolda T, O' Leary D. Algorithm 805:Computation and uses of the semidiscrete matrix decomposition. ACM Transactions on Mathematical Software (TOMS). 2000;**26**(3):415-435. DOI: 10.1.1.41.7987

[10] McConnell S, Skillicorn D. Outlier detection using semi-discrete decomposition. Department of Computing and Information Science. Kingston, Ontario, Canada: Queen's University; 2001. DOI:10.1.1.8.7634

[11] Kolda T, O'Leary D. Latent semantic indexing via a semi-discrete matrix decomposition. In: Cybenco G, O'Leary D, Rissanen J, editors. The Mathematics of Information Coding,

Extraction and Distribution. College Park, USA: Springer, University of Maryland; 1999. p. 73-80. DOI:10.1007/978-1-4612-1524-0_5

[12] Kolda T, Chew P, Abdelali A, Bader B. Crosslanguage information retrieval using PARAFAC2. In: Knowledge Discovery And Data Mining. Albuquerque, NM: ACM; 2007. p. 143-152

[13] Jian-Yun N. Cross-language information retrieval. Morgan & Claypool: Quebec, Canadá; 2010. p. 125. DOI: 10.2200/S00266ED1V01Y201005HLT008

[14] Berry M, Young P. Using latent semantic indexing for multilanguage information retrieval. Computers and the Humanities. 1995;**29**(6):413-429. DOI: 10.1007/BF01829874

[15] Dumais S, Letsche T, Littman M, Landauer T. Automatic cross-language information retrieval using latent semantic indexing. AAAI spring symposium on cross-language text and speech retrieval. 1997;**15**:21. DOI:10.1.1.533.9602