Research Article

A Heterogeneous System Based on Latent Semantic Analysis Using GPU and Multi-CPU

Gabriel A. León-Paredes,^{1,2} Liliana I. Barbosa-Santillán,² and Juan J. Sánchez-Escobar³

¹Universidad Politécnica Salesiana, Cuenca, Ecuador

²Universidad de Guadalajara, Guadalajara, JAL, Mexico

³Technical and Industrial Teaching Center, Guadalajara, JAL, Mexico

Correspondence should be addressed to Gabriel A. León-Paredes; gleon@ups.edu.ec

Received 15 June 2017; Accepted 26 September 2017; Published 5 November 2017

Academic Editor: José María Álvarez-Rodríguez

Copyright © 2017 Gabriel A. León-Paredes et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Latent Semantic Analysis (LSA) is a method that allows us to automatically index and retrieve information from a set of objects by reducing the term-by-document matrix using the Singular Value Decomposition (SVD) technique. However, LSA has a high computational cost for analyzing large amounts of information. The goals of this work are (i) to improve the execution time of semantic space construction, dimensionality reduction, and information retrieval stages of LSA based on heterogeneous systems and (ii) to evaluate the accuracy and recall of the information retrieval stage. We present a heterogeneous Latent Semantic Analysis (hLSA) system, which has been developed using General-Purpose computing on Graphics Processing Units (GPGPUs) architecture, which can solve large numeric problems faster through the thousands of concurrent threads on multiple CUDA cores of GPUs and multi-CPU architecture, which can solve large text problems faster through a multiprocessing environment. We execute the hLSA system with documents from the PubMed Central (PMC) database. The results of the experiments show that the acceleration reached by the hLSA system for large matrices with one hundred and fifty thousand million values is around eight times faster than the standard LSA version with an accuracy of 88% and a recall of 100%.

1. Introduction

Latent Semantic Analysis (LSA) is a method that allows us to automatically index and retrieve information from a set of objects by reducing a term-by-document matrix using term weighting schemes such as Log Entropy or Term Frequency-Inverse Document Frequency (TF-IDF) and using the Singular Value Decomposition (SVD) technique. LSA improved one of the main problems of information retrieval techniques, that is, handling polysemous words, by assuming there is some underlying latent semantic structure in the data that is partially obscured by the randomness of word choice [1]. LSA uses statistical techniques to estimate this latent structure and get rid of the obscuring “noise.” Also, LSA has been considered as a new general theory of acquisition of similarities and knowledge representation, which is helpful in simulating the learning of vocabulary and other psycholinguistic phenomena [2].

Latent Semantic Analysis, from its beginnings to the present, has been implemented in several research topics, for example, in applications to predict a reader’s interest in a selection of news articles, based on their reported interest in other articles [3]; in the self-diagnosis of diseases through the description of medical imaging [4]; in applications to detect cyberbullying in teens and young adults [5]; in the field of visual computers by improving techniques for tracking moving people [6]; in applications for the classification of less popular websites [7].

LSA has a computational complexity of $O(d^2k^3)$, where d is the smaller value between the number of documents and the number of terms and k is the number of singular values [8]. LSA takes a considerable amount of time to index and to compute the semantic space, when it is applied to large-scale datasets [9–11].

An introduction for a parallel LSA implementation based on a GPU has achieved acceleration of five to seven times with

large matrices divisible by 16- and 2-fold for matrices with another size. The GPU is being used for the tridiagonalization of matrices, and the routines that compute the eigenvalues and eigenvectors of matrices are still being implemented on the CPU. The results present that the accuracy and speed needed further research in order to produce an effective fully implementable LSA algorithm [12].

A technique called *index interpolation* is presented for a rapid computation of the term-by-document matrix for large documents collections; the associated symmetric eigenvector problem is then solved by distributing its computation among any number of computational units without increasing the overall number of multiplications. The experiments took 42.5 hours to compute 300,000 terms on 16 CPUs [13].

We present a fully *heterogeneous system based on Latent Semantic Analysis* (hLSA), which utilizes the resources of both GPU and CPU architectures to accelerate execution time. Our aim is to compute, reduce, and retrieve information faster than standard LSA versions and to evaluate the accuracy and recall of the information retrieval procedure in the hLSA system. The performance of hLSA has been evaluated, and the results show that acceleration as high as eight times could be achieved, with an accuracy of 88% and a recall of 100%. An early version of the hLSA system has been presented as a poster in the GPU Technology Conference [14].

The rest of the paper is organized as follows. Section 2 introduces the related background of LSA. In Section 3, we present our heterogeneous Latent Semantic Analysis system. Section 4 gives a description of the design of experiments. Section 5 presents the results of the experiments. Finally, Section 6 concludes the work.

2. Background

LSA takes a matrix of a term-by-document (M) and constructs a semantic space wherein terms and documents that are closely associated are placed near one another. Normally, the constructed semantic space has as many dimensions as unique terms. Additionally, instead of working with count data, the entries of matrix M are weighted with a representation of the occurrence of a word token within a document. Hence, LSA uses a normalized matrix which can be large and rather sparse. For this research, two types of weightings scheme are used.

(i) The first is a logarithmic local and global entropy weighting, known as a Log Entropy scheme. That is, if $M_{[w,d]}$ denotes the number of times (frequency) that a word w appears in document d and N is the total number of documents in the dataset, then

$$A_{[w,d]} = \log(M_{[w,d]} + 1) \times \left(1 + \frac{\sum_{i=1}^N P_{[w,i]} \times \log(P_{[w,i]})}{\log(N)} \right), \quad (1)$$

where $P_{[w,i]}$ is the fraction of documents containing the i th term; for example,

$$P[w, d] = \frac{M_{[w,d]}}{\sum_{i=1}^N M_{[w,i]}}. \quad (2)$$

This particular term weighting scheme has been very successful for many LSA studies [15], but other functions are possible.

(ii) The second is a term frequency and inverse document frequency weighting, known as a TF-IDF scheme, which assigns to word w a weight in document d , when N is the total number of documents in the dataset and M is the term frequency by documents matrix; for example,

$$A_{[w,d]} = \frac{M_{[w,d]}}{\sum_{i=1}^N M_{[i,d]}} \times \log\left(\frac{N}{\sum_{i=1}^N M_{[w,i]}}\right). \quad (3)$$

This particular term weighting scheme has been very successful for many LSA studies [8].

To reflect the major associative patterns in matrix A and ignore the smaller less important influences, a reduced-rank approximation of matrix A is computed using the truncated Singular Value Decomposition [16]. Note that the SVD of the original weighted matrix can be written as

$$A_{[w,d]} = T_{[w,n]} \times S_{[n,n]} \times D_{[n,d]}^T, \quad (4)$$

where $A_{[w,d]}$ is the words-by-documents matrix; T is a $[w \times n]$ orthogonal matrix whose values represent the left singular vectors of A ; D is a $[d \times n]$ orthogonal matrix whose values represent the right singular vectors of A ; and S is a $[n \times n]$ diagonal matrix which contains the singular values of A in descending order. Note that n is the smaller value between the total number of words w and the total number of documents d .

To obtain the truncated SVD denoted by A' , it is necessary to restrict SVD matrices to their first $k < \min(\text{terms, documents})$ dimensions, as revealed by

$$A'_{[w,k]} = T_{[w,k]} \times S_{[k,k]} \times D_{[k,d]}^T. \quad (5)$$

Choosing the appropriate number of dimensions k is an open research problem. It has been proposed that the optimum value of k is in a range from 50 to 500 dimensions, depending on the size of the dataset. As described in [17], if the number of dimensions is too small, significant semantic content will remain uncaptured, and if the dimension is too large, random noise in word usage will be remodeled. Note that the truncated SVD represents both terms and documents as vectors in k -dimensional space.

Finally, for information retrieval purposes, the k -dimensional semantic space is used. Thus, the terms of a user query are folded into the k -dimensional semantic space to identify a point in the space. This can be accomplished by parsing a query into a vector denoted by q whose nonzero values correspond to the term weights of all unique valid words of the user query. Then, the query folding process denoted by q' can be represented as

$$q'_{[k]} = q_{[w]}^T \times T_{[w,k]} \times S_{[k,k]}^{-1}. \quad (6)$$

Then, this q' vector can be compared with any or all documents/terms vectors of the k -dimensional semantic space. To

FIGURE 1: The hLSA system presents three stages: semantic space, dimensionality reduction, and information retrieval. Also, the hLSA system works with a user query, as well as a knowledge base, and presents the relevant documents.

compare vectors, the dot product or cosine between points is used. For example,

$$\cos_{(v,q')} = \frac{\vec{v} \times \vec{q'}}{|\vec{v}| \times |\vec{q'}|}, \quad (7)$$

where v is a vector representation of the k -dimensional space. LSA proposes the retrieval of information in two ways: (1) by establishing a minimum value of similarity, for example, all the similarities greater than 0.90, and (2) by obtaining the top-ranked values of similarity, for example, the top 10 and the top 5.

3. Heterogeneous Latent Semantic Analysis System

The hLSA system has several technical challenges: (1) how to construct the *semantic space* using the multi-CPU architecture to speed up the text processing; (2) how to *reduce the dimensionality* of the term-by-document matrix using the GPU architecture to accelerate the matrices processing; and (3) how to *retrieve information* from the semantic space using GPU mechanisms to speed up the matrix and text computations. Figure 1 presents the proposed hLSA system for constructing, reducing, and retrieving relevant documents using heterogeneous architectures.

Notably, one of the features of the hLSA system is the use of GPU architecture, which is able to execute SIMD (Single Instruction, Multiple Data) operations, such as matrix and

vector multiplications, very efficiently with high parallelism. In addition, we take advantage of the multi-CPU architecture, which is able to execute SIMD operations, such as *map* and *reduce* functions. On the one hand, the hLSA system utilizes the GPU architecture to solve matrix operations, especially in the stage of *dimensionality reduction* and *information retrieval*. On the other hand, the hLSA system utilizes CPU and multi-CPU architectures to solve text processing, in the *semantic space* stage. Thus, with the use of these mechanisms the hLSA system is able to achieve accelerated performance.

3.1. Semantic Space Stage. The input for constructing the semantic space is a knowledge base; this is represented by a dataset of raw texts, which is generally stored in the disk drive of a personal computer. Therefore, the hLSA system first needs to find the text documents. This execution has no major computational cost and therefore is executed by the CPU in a sequential manner. As a result, a document list is generated which is loaded into the main memory of the CPU.

From the document list obtained, the hLSA system begins to read each document text and starts the preprocessing, for instance, eliminating the blank spaces at the beginning and the end and, also, eliminating special characters such as "?!(),,:;" and ignoring common words known as stopwords, such as "the"; "for"; and "you." As a result, the hLSA system generates a list of preprocessed texts, which is stored in the main CPU memory. This procedure is performed on CPU runtime in a sequential manner.

Then, the hLSA system starts to generate the count matrix of term frequency by documents denoted as A , where rows represent the terms and columns represent the documents.

FIGURE 2: The architecture used in the semantic space stage of the hLSA system. The blocks in light yellow represent the principal procedures executed by the CPU, the blocks in light blue represent the procedures executed by multi-CPU, and the blocks in light green represent the procedures executed by GPU, respectively. The multi-CPU includes shared memory models for multiprocessing programming.

In order to do this, the hLSA system analyzes the list of preprocessed documents.

Therefore, the hLSA system takes each preprocessed text of the list and splits the content when a blank space appears. For example, the text “This is an example” is split at each blank space. As a result, a list of words: [“This”; “is”; “an”; “example”] is generated.

Next, the hLSA system iterates the new list of words and for each time the word appears in a document the hLSA system adds one to the corresponding cell value of matrix A . This procedure has a high computational cost.

Thus, we implement a parallel multiprocessing model using a multi-CPU architecture. Each CPU core is in charge of processing an element of the preprocessed text of the list as shown in Figure 2. As a result of the semantic space stage, the hLSA system generates in parallel and sequential procedures

matrix A , where each cell value corresponds to the frequency that a word appears in document of the knowledge base.

3.2. Dimensionality Reduction Stage. The hLSA system uses the CUDA programming model in order to process the dimensionality reduction of matrix A . As a consequence, the hLSA system uses two dimensions (x , y) of the CUDA architecture; the x dimension is associated with the rows of matrices and the y dimension with the columns of matrices. Each block of the CUDA architecture executes 32 concurrent threads and the total number of blocks executed depends on the total size of the matrix to be processed.

The hLSA system needs to normalize matrix A . Therefore, two term weighting schemes are used to normalize matrix A : the Log Entropy and the Term Frequency-Inverse Document Frequency schemes. In order to compute the term weighting

FIGURE 3: The architecture used in the dimensionality reduction stage of the hLSA system. The blocks in light yellow represent the principal procedures executed by the CPU, the blocks in light blue represent the procedures executed by multi-CPU's, and the blocks in light green represent the procedures executed by GPU, respectively. The GPU executions include CUDA kernels and CUDA functions using floating-point numbers.

schemes, the hLSA system loads matrix A into the GPU global memory and, then, executes the CUDA kernel functions and the resulting matrix is saved into the GPU global memory as A_norm .

The hLSA system uses the SVD algorithm, which decomposes matrix A_norm into three matrices. Matrix T is an orthogonal matrix of left singular vectors. Matrix S is a diagonal matrix and contains the singular values ordered by magnitude. Matrix D^T is an orthogonal matrix of right singular vectors. Therefore, the hLSA system needs to reserve space in the GPU global memory for the resultant matrices T , S , and D^T . Then, it executes the `compute_svd` CUDA function. At the end of the execution the hLSA system frees the GPU memory space of matrix A_norm .

Finally, the hLSA system truncates the resultant matrices of the SVD function into their first k dimensions. Therefore, it executes the `truncate_svd` CUDA function. As a result, the hLSA system generates three matrices denoted by S_y , T_y , and D_y . At the end of the execution the matrices are copied into the main memory of the CPU and then the GPU global memory is freed. The dimensionality reduction stage of the hLSA system is shown in Figure 3.

3.3. Information Retrieval Stage. The hLSA system uses a user query in order to retrieve the relevant information from the semantic space. Therefore, the hLSA system creates a one-dimensional vector that is denoted by q , with a size equal to the total number of terms obtained in the semantic

space. Then, it analyzes the query; namely, if a word of the query corresponds to a word in the semantic space, the hLSA system increments by one the corresponding cell value in the vector q . This execution has no major computational cost and therefore is executed by the CPU in a sequential manner.

Then the hLSA system folds the vector q into the semantic space by using the query folding equation (6). Therefore, the hLSA system executes the *query_folding* CUDA function using the matrices S_y , T_y , and D_y loaded in the GPU global memory. As a result, the hLSA system generates a vector q' , whose values correspond to the terms weights of all the unique words of the query. Then, the hLSA system compares the vector q' with all document vectors inside the semantic space in order to find their similarity.

To compute the similarity, the hLSA systems uses the cosine similarity equation (7). Hence, it executes the *cosine_similarity* CUDA function using the vector q' and the document vectors of matrix D_y . Ultimately, the hLSA system presents the most relevant documents based on the similarity value between the vector q' and each document vector. To present the results, the hLSA system establishes a threshold of similarity, for example, all the similarities greater than 0.90, and/or the hLSA systems obtain the top-ranked values of similarity, for example, the top 20, the top 10, and the top 5. The information retrieval stage of the hLSA system is shown in Figure 4.

4. Experiments

The objective of this section is to evaluate the performance of the proposed hLSA system. The design of the experiments is detailed in (1) Definition of the Knowledge Base, (2) Definition of Dataset Size, (3) Definition of Stopword Document, (4) Definition of k Dimensions, (5) Definition of Use Cases, and (6) Evaluation Methods.

(1) *Definition of Knowledge Base.* The experiments conducted with the hLSA system use documents of the Open Access Subset of PubMed Central (PMC) as the knowledge base of the experiments. PMC is an online digital database of freely available full-text biomedical literature. Extensively used in the Text REtrieval Conference (TREC) (the Text REtrieval Conference (TREC), cosponsored by the National Institute of Standards and Technology (NIST) and the US Department of Defense, can be found at <http://trec.nist.gov>). The text of each article in the open access subset is represented as an NXML file, which is an XML format used by NML Journal Archiving and Interchange Tag Set (the Journal Archiving and Interchange Tag Set defines elements and attributes that describe the content and metadata of journal articles, including research and nonresearch articles, letters, editorials, and book and product reviews).

The documents are named and identified by a unique number (PMCID) defined by an `<article-id>` element. Also, each NXML file has XML elements such as `<article-title>`, `<doi>`, `<abstract>`, `<fig>`, and `<publisher-name>`. For experimental purposes, we only use the abstract of the document. This information is extracted from each NXML and copied

to a new text file (.txt). Each text file is stored in a working directory with a file name equal to the PMCID identifier.

(2) *Definition of Dataset Size.* The experiments conducted with the hLSA system use a total of five thousand documents of the PMC database. In order to observe the variation of the total time of execution of the hLSA system with large-scale datasets, we partition the five-thousand documents into twelve subsets of different sizes ranging in size from five hundred to five thousand documents. In Table 1, we present a summary of each subset. The first column represents the total number of documents. The second column represents the total number of words in a dataset after preprocessing the raw texts. The third column presents the total number of unique words in the dataset. The fourth column represents the total number of elements in the matrix of term-by-document and the last column represents the size in megabytes of the term-by-document matrix.

(3) *Definition of Stopword Document.* The experiments conducted with the hLSA system uses a document file of stopwords to exclude certain single words like determinants, that is, “the,” “a,” “an,” and “another”; coordinating conjunctions, that is, “for,” “but,” “or,” and “so”; prepositions, that is, “in,” “under,” and “towards”; pronouns forms, that is, “me,” “my,” “myself,” and “you”; verb forms, that is, “am,” “is,” “was,” and “do”; among other words. In total, one thousand nine hundred eighty-two words were used.

(4) *Definition of k Dimensions.* The experiments conducted with the hLSA uses several values of k to find the appropriate value with which the hLSA system is more accurate. In total, twenty values of k were used ranging from twenty-five to five hundred dimensions in increments of twenty-five.

(5) *Definition of Use Cases.* The experiments in the hLSA system were conducted with three use cases: (a) bipolar disorders, (b) lupus disease, and (c) topiramate weight-loss.

(a) *Bipolar Disorders.* Bipolar disorders could be presented in patients in their middle age with obese disorders, who struggle with their weight and eat when feeling depressed, excessively anxious, agitated, and irritable and when having suicidal thoughts and/or difficulty sleeping.

(b) *Lupus Disease.* Lupus disease could be presented in patients with alopecia, rash around their nose and cheeks, delicate nonpalpable purpura on their calves, and swelling and tenderness of their wrists and ankles and with normocytic anemia, thrombocytopenia, and being positive for protein and RBC casts.

(c) *Topiramate Weight-Loss.* Some research reports have shown that the anticonvulsant topiramate causes weight-loss in various patient groups. Patients have lost a mean of 7.75 kg in six months and at twelve months patients have lost a mean of 9.61 kg.

(6) *Evaluation Methods.* We have defined the size of the subsets, the number of k dimensions, and the use cases for our

FIGURE 4: The architecture used in the information retrieval stage of the hLSA system. The blocks in light yellow represent the principal procedures executed by the CPU, the blocks in light blue represent the procedures executed by multi-CPU, and the blocks in light green represent the procedures executed by GPU, respectively. The GPU executions include CUDA kernels and CUDA functions using floating-point numbers.

experiments. Hence, to evaluate our hLSA system, we have used two methods: (a) we have built an LSA system using CPU sequential-only architecture, in order to specifically evaluate the acceleration reached by the hLSA system. Thus, we have compared the execution time of the hLSA and LSA (CPU sequential-only) system. (b) We have evaluated the accuracy and recall of the information retrieval stage.

(a) *Time Execution.* We have executed the hLSA system and LSA systems a total of four hundred eighty times to complete all the experiments with the different subsets, k dimensions, weighting schemes, and use cases. Therefore, in order to

compare the execution time of the hLSA system versus the LSA system, we have collected the execution time of each procedure involved in each stage of both systems. Hence, we have presented in Section 4 the mean values of these executions.

(b) *Similarity Accuracy.* We have executed the hLSA system and LSA systems a total of four hundred eighty times to complete all the experiments with the different subsets, k dimensions, weighting schemes, and use cases. Therefore, in order to evaluate the information retrieval process of the hLSA system, we have presented the top 10 ranked

TABLE 1: Subsets used in the experiments for the hLSA system.

Number of documents	Number of words	Number of unique words	Number of elements in matrix	Size of matrix (Mb)
500	13856	6819	3,409,500	27
750	18194	9039	6,779,250	54
1000	22107	11026	11,026,000	88
1250	25264	12587	15,733,750	126
1500	28251	14066	21,099,000	169
1750	31282	15596	27,293,000	218
2000	34024	16990	33,980,000	272
2500	39239	19650	49,125,000	393
3000	44114	22039	66,117,000	529
3500	48622	24284	84,994,000	680
4000	53003	26373	105,492,000	844
5000	61466	30273	151,365,000	1211

documents, and we have compared the similarity values generated by the hLSA system in each experiment with the subset of five thousand documents.

5. Results

In this section, we present the results of the experiments with the hLSA system. First, we show the execution time for the semantic space stage. Second, we present the execution time for the dimensionality reduction stage. Third, we show the execution time for the information retrieval stage. Fourth, we present the execution time for the full hLSA system. Finally, the results to evaluate the similarity accuracy and recall of the information retrieval stage are presented.

The hLSA system has been executed on a Linux OpenSUSE 13.2 (3.16.6-2 kernel) system with Core i7 Intel processors at 2.50 GHz with 16 GB DDR3L of main memory and 4 cores with 8 threads for each core and a NVIDIA GeForce GTX 970M. The GPU has 10 SMs with 128 CUDA cores for each SM, a total of 1280 CUDA cores. The GPU has a total amount of global memory of 3064 Mbytes and a clock speed of 2.5 GHz. Also, the maximum number of threads per multiprocessor is 2048, and the maximum number of threads per block is 1028. The GPU maximum dimension size of thread block is “1024, 1024, and 64” in the (x, y, z) dimension.

(1) *Runtime Results: Semantic Space Stage.* The procedures executed in the semantic space stage are to read the stopwords file, find documents in the hard disk drive, read documents from the hard disk drive, parse documents into the main CPU memory, and build the matrix of terms frequency by documents. The hLSA system uses the benefits of multi-CPU architecture to parse the documents into the main CPU memory. Table 2 present the execution time of procedures executed in CPU sequential-only. In Table 3, we compare the execution time of the *Document Parser* procedure using CPU sequential-only and multi-CPU architecture with four, eight, and sixteen processors. Additionally, in Figure 5, we present the overall execution time using CPU and multi-CPU

FIGURE 5: Results of the overall execution time in seconds of the semantic space stage using CPU sequential-only architecture and multi-CPU architecture with four, eight, and sixteen processors.

architectures. The execution times were obtained as mean values of the four hundred eighty executions and are presented in milliseconds.

The procedures with high execution time in the LSA system are *Read Documents* and *Document Parser*. On the other hand, the procedure with the highest execution time in hLSA system is *Read Documents*. This is due to the fact that the *Read Documents* procedure has to access the drive disk in order to read the text documents, and the time to access the drive disk depends on the hardware. Therefore, there is a bottleneck which is limited by the hardware of disk I/O system, which goes beyond the scope of our work.

Moreover, we have found a substantial improvement in the *Document Parser* procedure in the hLSA system using multi-CPU architecture. Acceleration of 2.50 times is seen with five hundred documents and maximum acceleration of 2.90 times with five-thousand documents using four processors. When using eight processors, acceleration of 3.00 times

TABLE 2: Results of the *CPU sequential-only* procedures in milliseconds of the semantic space stage in the hLSA system.

Number of documents	Read stopwords (ms)	Find documents (ms)	Read documents (ms)	Build matrix (ms)
500	27.016	186.764	4.044	41.646
750	27.016	187.885	5.929	61.891
1000	27.016	195.235	7.845	82.295
1250	27.016	196.450	9.752	105.363
1500	27.016	189.951	12.333	121.197
1750	27.016	191.454	13.816	152.200
2000	27.016	198.723	16.475	164.590
2500	27.016	194.696	19.134	208.550
3000	27.016	192.528	23.034	250.311
3500	27.016	191.431	26.621	297.541
4000	27.016	198.942	34.243	339.805
5000	27.016	189.929	37.598	434.999

TABLE 3: Results of the *Document Parser* procedure in milliseconds of the semantic space stage in the LSA system using CPU sequential-only architecture and in the hLSA system using multi-CPU architecture with four, eight, and sixteen processors.

Number of documents	Document Parser Procedure			
	LSA (ms)	hLSA with 4 processors (ms)	hLSA with 8 processors (ms)	hLSA with 16 processors (ms)
500	1681.982	676.587	504.683	549.639
750	2587.080	1003.574	720.285	780.755
1000	3453.187	1351.907	984.871	1037.802
1250	4241.641	1549.046	1205.027	1128.528
1500	5121.449	1770.975	1303.507	1347.592
1750	6016.346	2236.582	1522.095	1596.852
2000	6806.595	2158.235	1789.595	1766.482
2500	8645.503	2667.631	2212.338	2215.629
3000	10351.710	3799.512	2594.976	2740.206
3500	12066.547	4420.248	3077.626	3065.884
4000	13762.060	4582.366	3495.762	3442.618
5000	17247.301	6029.288	4477.206	4352.604

is seen with five-hundred documents and maximum acceleration of 3.85 times with five thousand documents. Finally, for sixteen processors, acceleration of 3.30 times is seen with five-hundred documents and maximum acceleration of 3.96 times with five thousand documents. Thus, a better overall performance is found with sixteen processors and greater acceleration when the document corpus increases.

(2) *Runtime Results: Dimensionality Reduction Stage.* The weighting schemes Log Entropy and Term Frequency-Inverse Document Frequency have a high computational cost in the LSA system, as shown in Table 4. The hLSA system reaches acceleration of thirty times for the smaller datasets (five hundred to two thousand five hundred) and acceleration of eight hundred fifty times for the larger datasets (three thousand to five thousand) using the TF-IDF scheme. Also,

using the Log Entropy scheme the hLSA system has reached acceleration of sixty-two times for the smaller datasets and acceleration of one thousand fifty-two times for the bigger datasets.

The computational complexity of Log Entropy is $O(d)$ and that of TD-IDF schemes is $O(dw)$, where w is the total number of individual words and d is the total number of documents in the corpus. As shown in the results, the TF-IDF scheme in the hLSA system has reduced the execution time from more than 160 seconds to less than 0.20 seconds, and the Log Entropy scheme in the hLSA system has reduced the execution time from more than 321 seconds to less than 0.20 seconds.

Meanwhile, an exact SVD has a computational complexity of $O(dw^2)$, and this is expensive for large matrices. In the hLSA system, we implement the GPU-SVD function, which

TABLE 4: Results of the execution time in the hLSA system using GPU-accelerated procedures and in the LSA system using sequential-only procedures for the Log Entropy scheme and the TF-IDF scheme.

Number of documents	TF-IDF		Log Entropy	
	hLSA (ms)	LSA (ms)	hLSA (ms)	LSA (ms)
500	117.814	3676.445	118.735	7401.700
750	118.319	7315.378	121.697	14300.521
1000	118.622	11845.949	121.143	23370.072
1250	123.446	17255.605	124.997	33131.246
1500	127.632	22982.827	132.696	45446.027
1750	130.057	29501.134	129.801	57330.141
2000	134.766	36193.939	135.845	71768.188
2500	143.089	54128.704	154.173	103304.719
3000	148.864	70769.040	154.600	139719.203
3500	161.621	92035.205	161.590	179290.609
4000	167.414	114055.179	175.994	233267.328
5000	189.126	160822.907	206.725	320797.375

FIGURE 6: Results of the execution time of the SVD procedure applied in the hLSA system using GPU-accelerated procedures and in the LSA system using CPU sequential-only procedures.

gives us acceleration of up to three times compared to the LSA system. In Figure 6, we present a comparison of the SVD execution time between the hLSA system and the LSA system.

In the smallest dataset, we measure an execution time of 1.60 seconds in the LSA system and, in the largest dataset, we measure an execution time of 122.10 seconds. On the other hand, in the hLSA system, we measure an execution time of 0.50 seconds in the smallest dataset and an execution time of 53.40 seconds in the largest dataset. Hence, we reach acceleration of two to three times in most cases.

Consequently, the SVD procedure with the NVIDIA Visual Profiler was analyzed. We found that SVD generates about 77 percent of all the time-processing and launches around six thousand seven hundred thirty kernel instances of SVD in CUDA architecture. In addition, the profiler showed an optimization problem in memory access for computing the SVD procedure. For shared memory, the SVD utilizes a low bandwidth of 39.193 GB/s for 273,163 total load/store transactions, and, for device memory, the SVD utilizes a low

FIGURE 7: Results of a complete execution of dimensionality reduction stage with k -dimension equal to 300 and TF-IDF weighting scheme applied in the hLSA system using GPU-accelerated procedures and the LSA system using CPU sequential-only procedures.

bandwidth of 42.814 GB/s for 1,193,617 read/write transactions.

Meanwhile, the computational cost for the truncated SVD procedure takes less than one second in the LSA system using the biggest dataset. However, in the hLSA system, we have accelerated up to nine times for the smaller datasets and up to four times for the larger datasets. In Table 5, we present the results of an execution with the dataset of two thousand documents. In this execution, our results present acceleration of four times.

We have presented the results of procedures: Log Entropy, TF-IDF, SVD, and Truncated SVD. As shown in results, the dimensionality reduction stage has the most computational cost in our experiments. Therefore, we now present in Figure 7 the results of a complete execution of dimensionality reduction stage with $k = 300$ and the TF-IDF weighting scheme. We obtain for the smaller datasets acceleration from

TABLE 5: Results of execution time in a truncated SVD procedure applied in the hLSA system using GPU-accelerated procedures and the LSA system using CPU sequential-only procedures for 2000 documents.

Number of k	hLSA				LSA			
	Matrix S (ms)	Matrix D (ms)	Matrix T (ms)	Total (ms)	Matrix S (ms)	Matrix D (ms)	Matrix T (ms)	Total (ms)
25	0.158	7.241	59.635	67.033	0.094	11.151	254.472	265.717
50	0.134	6.540	59.228	65.902	0.113	12.756	251.627	264.496
75	0.144	6.870	59.576	66.590	0.127	10.842	250.042	261.011
100	0.149	4.216	60.443	64.808	0.118	10.594	251.530	262.242
125	0.118	5.064	60.644	65.826	0.115	11.048	262.973	274.136
150	0.176	4.657	61.183	66.016	0.127	10.387	261.292	271.806
175	0.113	6.631	61.259	68.002	0.120	10.626	257.150	267.896
200	0.162	4.195	62.297	66.653	0.135	10.148	265.645	275.928
225	0.174	4.204	63.519	67.897	0.140	10.298	266.845	277.283
250	0.177	4.941	63.100	68.219	0.142	10.090	256.517	266.749
275	0.177	4.599	65.915	70.691	0.140	10.917	257.991	269.048
300	0.174	4.829	64.671	69.675	0.147	10.869	269.890	280.906
325	0.174	6.413	64.655	71.242	0.150	10.774	254.650	265.574
350	0.178	4.269	67.372	71.818	0.149	10.570	253.184	263.903
375	0.157	4.843	68.000	72.999	0.194	10.630	252.340	263.164
400	0.209	5.808	66.520	72.537	0.181	10.776	252.570	263.527
425	0.209	6.443	66.587	73.239	0.174	12.384	265.330	277.888
450	0.243	5.847	69.124	75.213	0.162	10.868	258.973	270.003
475	0.198	4.866	70.410	75.474	0.169	10.796	265.007	275.972
500	0.201	5.765	70.373	76.338	0.172	10.656	256.289	267.117

TABLE 6: Results of the execution time of query parser procedure applied in the hLSA system with the three use cases: *bipolar disorders*, *lupus disease*, and *topiramate weight-loss*.

Number of documents	Bipolar disorders (sec)	Lupus disease (sec)	Topiramate weight-loss (sec)
500	7.81	6.22	27.13
750	10.28	8.20	35.99
1000	12.62	9.96	44.06
1250	14.22	11.39	50.22
1500	15.92	12.88	56.38
1750	17.71	14.11	62.29
2000	19.34	15.38	68.33
2500	22.22	17.81	78.71
3000	25.05	20.08	88.35
3500	27.51	22.09	97.49
4000	30.11	23.93	107.81
5000	34.46	27.50	121.37

six to ten times, and for the larger datasets acceleration from five to eight times.

(3) *Runtime Results: Information Retrieval Stage.* We have implemented the hLSA system not just for indexing documents, but also for information retrieval. Thus, we now present the results of the execution time in information

retrieval procedures, which are as follows: query parser, query folding, and cosine similarity.

We obtain the process time for the query parser procedure, which has been implemented in CPU sequential-only for both systems hLSA and LSA. We present, in Table 6, the results of the query parse procedure on each dataset using the three use cases.

TABLE 7: Results of execution time for the query folding and cosine similarity procedures are applied in the hLSA system using GPU-accelerated procedures and the LSA system using sequential-only procedures for 5000 documents and topiramate weight-loss use case.

Number of k	hLSA		LSA	
	Query folding (ms)	Cosine similarity (ms)	Query folding (ms)	Cosine similarity (ms)
175	1.989	585.023	26.348	245.986
200	3.484	620.742	2.401	246.752
225	3.979	621.718	14.623	247.087
250	3.929	614.037	6.367	248.554
275	4.065	595.509	19.193	256.337
300	4.539	623.648	7.603	255.993
325	4.900	595.137	4.495	259.016
350	4.805	589.494	30.019	262.982

TABLE 8: Results of the total execution time in the hLSA system versus the LSA system. Results include all procedures using 5000 documents, $k = 300$, Log Entropy, TF-IDF, and four, eight, and sixteen multi-CPU.

	Bipolar disorders		Lupus disease		Topiramate weight-loss	
	Log Entropy (sec)	TF-IDF (sec)	Log Entropy (sec)	TF-IDF (sec)	Log Entropy (sec)	TF-IDF (sec)
hLSA	96.29	110.16	87.42	86.40	181.93	180.25
LSA	770.50	610.82	756.33	590.01	848.97	689.12
Acceleration	8.00	5.54	8.65	6.83	4.67	3.82

The process with the highest computational cost in terms of the information retrieval stage is the query parser procedure. For the biggest dataset with the use case of bipolar disorders it took around 34 seconds, with the use case of lupus disease it took around 27 seconds, and with the use case of topiramate weight-loss it took more than 120 seconds. This is due to the number of unique words in the queries. The first use case has an average of 31 unique words, the second use case has an average of 22 unique words, and the last use case has an average of 136 unique words.

Additionally, we now present the result times for the query folding procedure and cosine similarity procedure. We present the best results using the use case of topiramate weight-loss. These procedures do not have a high computational cost; in most cases, the execution time is less than one second for all datasets in the hLSA and LSA system. Table 7 shows several results of query folding and cosine similarity procedures.

(4) *Runtime Results: hLSA versus LSA.* We present in Table 8 the overall execution time for the hLSA system versus the LSA system with five thousand documents, k -dimension equal to three hundred, three use cases, and two weighting schemes. Also, for use case of bipolar disorders, we present results with four processors; for use case of lupus disease, we present results with eight processors; and, for use case of topiramate weight-loss, we present results with sixteen processors. We reach overall acceleration of five to eight times.

(5) *Information Retrieval Results.* To evaluate the accuracy, we compare the documents retrieved by the hLSA system based on a text query related to each use case versus the relevant documents defined by the experts. The experts define that the most relevant documents for use case of bipolar disorders are the articles with identifiers 1087494, 1434505, and 2031887; for use case of lupus disease the most relevant documents are the articles with identifiers 1065341, 1459118, and 1526641; and for use case of topiramate weight-loss the most relevant document is the article with identifier 1087494. We present the results of the similarities for each relevant document covering all the documents of the knowledge base with two weighting schemes and twenty values of k , for use case of bipolar disorders in Figure 8, for use case of lupus disease in Figure 9, and for use case of topiramate weight-loss in Figure 10.

As shown in the results, the best similarities found in the experiments with the hLSA system for use case of bipolar disorders are found with values of $k = 50$ and accuracy = 0.88, for use case of lupus disease are found with values of $k = 25$ and accuracy = 0.56, and for use case of topiramate weight-loss are found with values of $k = 25$ and accuracy = 0.98. Anyhow, the similarities values are normalized in the range from one hundred fifty to three hundred.

Moreover, we present the results of the top ten documents with use case of bipolar disorders in Table 9; with use case of lupus disease, we show results in Table 10, and, for use case of topiramate weight-loss, we present the results in Table 11.

TABLE 9: Results of the information retrieval process in the hLSA system with use case of bipolar disorders, using Log Entropy and TF-IDF weighting scheme with 500 documents and $k = 75, 100, 125$, and 150 ; 2000 documents and $k = 175, 200, 225$, and 250 ; 4000 documents and $k = 275, 300, 325$, and 350 .

Tf-Idf		Log Entropy		Tf-Idf		Log Entropy			
File	Similarity	File	Similarity	File	Similarity	File	Similarity		
500 documents									
$k = 75$				$k = 100$					
1	1434505.txt	0.62	3662141.txt	0.53	1	1434505.txt	0.59	1087494.txt	0.53
2	3714066.txt	0.51	3416914.txt	0.47	2	3714066.txt	0.48	3425832.txt	0.44
3	3181786.txt	0.39	1087494.txt	0.47	3	3821851.txt	0.37	3650299.txt	0.42
4	3103169.txt	0.38	3395358.txt	0.42	4	3181786.txt	0.35	3662141.txt	0.41
5	3821851.txt	0.35	3574453.txt	0.42	5	3662141.txt	0.34	2776598.txt	0.39
6	3425832.txt	0.34	2776598.txt	0.41	6	3585785.txt	0.31	3078090.txt	0.37
7	3662141.txt	0.32	3078747.txt	0.40	7	2933487.txt	0.31	2031887.txt	0.36
8	2933487.txt	0.32	3425832.txt	0.39	8	3425832.txt	0.30	3483824.txt	0.33
9	3585785.txt	0.30	3483824.txt	0.39	9	1087494.txt	0.28	3443410.txt	0.33
10	3574453.txt	0.29	3650299.txt	0.38	10	3785904.txt	0.28	3395358.txt	0.32
$k = 125$				$k = 150$					
1	1434505.txt	0.55	1087494.txt	0.53	1	1434505.txt	0.49	1087494.txt	0.57
2	3714066.txt	0.46	3425832.txt	0.45	2	3714066.txt	0.40	3650299.txt	0.45
3	2933487.txt	0.33	3650299.txt	0.41	3	2933487.txt	0.34	3425832.txt	0.43
4	3821851.txt	0.31	3662141.txt	0.38	4	3483824.txt	0.33	3483824.txt	0.37
5	1087494.txt	0.28	3483824.txt	0.34	5	3416914.txt	0.33	3443410.txt	0.33
6	1779284.txt	0.28	3078090.txt	0.34	6	3662141.txt	0.33	3078680.txt	0.33
7	3181786.txt	0.27	3443410.txt	0.32	7	3078747.txt	0.32	3416914.txt	0.32
8	3662141.txt	0.26	2031887.txt	0.30	8	1779284.txt	0.31	1434505.txt	0.31
9	3585785.txt	0.26	3078680.txt	0.29	9	1087494.txt	0.31	3662141.txt	0.29
10	3785904.txt	0.26	3416914.txt	0.28	10	2031887.txt	0.30	3180447.txt	0.27
2000 documents									
$k = 175$				$k = 200$					
1	1434505.txt	0.53	2928181.txt	0.47	1	1434505.txt	0.52	3662141.txt	0.42
2	2928181.txt	0.50	2758213.txt	0.45	2	3714066.txt	0.44	2758213.txt	0.40
3	2758213.txt	0.47	3662141.txt	0.43	3	2758213.txt	0.41	2928181.txt	0.38
4	3714066.txt	0.46	3650299.txt	0.42	4	2928181.txt	0.41	3574453.txt	0.37
5	2031887.txt	0.39	3574453.txt	0.41	5	2031887.txt	0.39	2031887.txt	0.36
6	3833508.txt	0.37	1087494.txt	0.40	6	3615012.txt	0.35	1087494.txt	0.35
7	3615012.txt	0.36	1434505.txt	0.39	7	3833508.txt	0.33	3650299.txt	0.34
8	3759464.txt	0.33	2031887.txt	0.38	8	3759464.txt	0.31	3488815.txt	0.32
9	3662141.txt	0.31	3514449.txt	0.34	9	3662141.txt	0.31	3514449.txt	0.32
10	2426811.txt	0.31	3615012.txt	0.34	10	3641510.txt	0.30	3615012.txt	0.32
$k = 225$				$k = 250$					
1	1434505.txt	0.50	3662141.txt	0.38	1	1434505.txt	0.48	2758213.txt	0.37
2	3714066.txt	0.43	2758213.txt	0.37	2	3714066.txt	0.41	3662141.txt	0.33
3	2031887.txt	0.38	3650299.txt	0.34	3	2031887.txt	0.37	3650299.txt	0.33
4	2928181.txt	0.36	2928181.txt	0.34	4	2928181.txt	0.34	1087494.txt	0.32
5	2758213.txt	0.36	3641510.txt	0.32	5	3615012.txt	0.32	2031887.txt	0.32
6	3615012.txt	0.33	2031887.txt	0.32	6	2758213.txt	0.30	3641510.txt	0.31
7	3833508.txt	0.32	1087494.txt	0.32	7	3759464.txt	0.29	2928181.txt	0.30
8	3759464.txt	0.31	3615012.txt	0.30	8	3641510.txt	0.28	3724308.txt	0.29
9	3641510.txt	0.29	3574453.txt	0.29	9	3833508.txt	0.27	3615012.txt	0.28
10	2426811.txt	0.27	1434505.txt	0.28	10	3662141.txt	0.25	1434505.txt	0.28
4000 Documents									
$k = 275$				$k = 300$					
1	1434505.txt	0.42	3662141.txt	0.42	1	2031887.txt	0.41	3662141.txt	0.42
2	2031887.txt	0.41	1087494.txt	0.35	2	1434505.txt	0.40	1434505.txt	0.34

TABLE 9: Continued.

Tf-Idf		Log Entropy		Tf-Idf		Log Entropy	
File	Similarity	File	Similarity	File	Similarity	File	Similarity
3 3615012.txt	0.40	2758213.txt	0.32	3 3615012.txt	0.39	1087494.txt	0.34
4 2928181.txt	0.39	1434505.txt	0.32	4 3714066.txt	0.36	2758213.txt	0.33
5 3714066.txt	0.37	2928181.txt	0.32	5 3290984.txt	0.36	2928181.txt	0.31
6 3290984.txt	0.35	3574453.txt	0.31	6 2928181.txt	0.36	2990827.txt	0.31
7 2758213.txt	0.35	3488815.txt	0.30	7 3426905.txt	0.33	3135225.txt	0.30
8 3442741.txt	0.33	3135225.txt	0.30	8 3641510.txt	0.33	3650299.txt	0.29
9 3426905.txt	0.33	2031887.txt	0.30	9 2758213.txt	0.32	3488815.txt	0.29
10 3641510.txt	0.32	2990827.txt	0.30	10 3442741.txt	0.32	2031887.txt	0.29
$k = 325$				$k = 350$			
1 2031887.txt	0.39	3662141.txt	0.39	1 2031887.txt	0.38	3662141.txt	0.37
2 1434505.txt	0.39	1087494.txt	0.34	2 1434505.txt	0.38	1087494.txt	0.34
3 3615012.txt	0.36	1434505.txt	0.34	3 3615012.txt	0.35	1434505.txt	0.33
4 3714066.txt	0.34	2928181.txt	0.31	4 3290984.txt	0.34	2990827.txt	0.31
5 3290984.txt	0.34	2990827.txt	0.29	5 3426905.txt	0.33	2928181.txt	0.30
6 3426905.txt	0.33	2758213.txt	0.29	6 3714066.txt	0.33	3135225.txt	0.28
7 2758213.txt	0.32	3514449.txt	0.29	7 2758213.txt	0.30	2031887.txt	0.28
8 2928181.txt	0.31	2031887.txt	0.28	8 3641510.txt	0.30	2758213.txt	0.27
9 3641510.txt	0.31	3833508.txt	0.28	9 2928181.txt	0.29	3650299.txt	0.27
10 2579333.txt	0.29	3615012.txt	0.28	10 2579333.txt	0.28	3615012.txt	0.27

FIGURE 8: Results of the similarity for relevant document 1: 1087494, relevant document 2: 1434505, and relevant document 3: 2031887 in the use case of bipolar disorder using the subset of 5000 documents, twenty values of k , and two weighting schemes (TF-IDF and Log Entropy).

FIGURE 9: Results of the similarity for relevant document 1: 1065341, relevant document 2: 1459118, and relevant document 3: 1526641 in the use case of lupus disease using the subset of 5000 documents, twenty values of k , and two weighting schemes (TF-IDF and Log Entropy).

As shown in the results, in several of the experiments the identifiers for the relevant documents defined by the experts are retrieved in the top ten.

6. Conclusions

The paper has introduced a heterogeneous system based on the Latent Semantic Analysis method. The proposed system

TABLE 10: Results of the information retrieval process in the hLSA system with use case of lupus disease, using Log Entropy and TF-IDF weighting scheme with 1000 documents and $k = 50, 75, 100$, and 125; 3000 documents and $k = 250, 275, 300$, and 325; 5000 documents and $k = 350, 375, 400$, and 425.

Tf-Idf		Log Entropy		Tf-Idf		Log Entropy	
File	Similarity	File	Similarity	File	Similarity	File	Similarity
1000 documents							
$k = 50$				$k = 75$			
1	3654263.txt	0.64	3754886.txt	0.55	1	1065341.txt	0.85
2	2974973.txt	0.63	2106780.txt	0.47	2	1526641.txt	0.73
3	3271631.txt	0.61	2892131.txt	0.47	3	2816238.txt	0.72
4	3283793.txt	0.60	2836250.txt	0.46	4	3423037.txt	0.68
5	2933487.txt	0.59	3339798.txt	0.45	5	2141996.txt	0.59
6	2229596.txt	0.59	3517290.txt	0.45	6	1459118.txt	0.55
7	3501694.txt	0.58	3420478.txt	0.43	7	2138500.txt	0.48
8	2816238.txt	0.58	3412204.txt	0.43	8	2832080.txt	0.47
9	2684704.txt	0.58	3538549.txt	0.42	9	2133386.txt	0.44
10	2108140.txt	0.58	3754383.txt	0.42	10	2974973.txt	0.44
$k = 100$				$k = 125$			
1	1065341.txt	0.77	2106780.txt	0.36	1	1065341.txt	0.72
2	1526641.txt	0.61	3340108.txt	0.35	2	2816238.txt	0.57
3	2816238.txt	0.60	3654263.txt	0.33	3	2141996.txt	0.51
4	2141996.txt	0.56	3420478.txt	0.33	4	3423037.txt	0.49
5	3423037.txt	0.53	3517290.txt	0.32	5	1526641.txt	0.48
6	1459118.txt	0.48	3002958.txt	0.32	6	2974973.txt	0.44
7	2974973.txt	0.45	2816647.txt	0.29	7	1459118.txt	0.43
8	2138500.txt	0.43	3754886.txt	0.29	8	3514429.txt	0.40
9	3514429.txt	0.40	2836250.txt	0.28	9	2138500.txt	0.36
10	2119723.txt	0.40	3334796.txt	0.27	10	2119723.txt	0.33
3000 documents							
$k = 250$				$k = 275$			
1	1065341.txt	0.49	3507913.txt	0.39	1	1065341.txt	0.50
2	2247119.txt	0.47	2247119.txt	0.36	2	2247119.txt	0.46
3	3842271.txt	0.38	2975002.txt	0.33	3	3842271.txt	0.37
4	3260316.txt	0.38	3272557.txt	0.32	4	3260316.txt	0.36
5	3512896.txt	0.37	270621.txt	0.30	5	3512896.txt	0.36
6	270621.txt	0.36	2724402.txt	0.30	6	270621.txt	0.34
7	2141996.txt	0.34	3514429.txt	0.29	7	2141996.txt	0.34
8	2873406.txt	0.33	3368184.txt	0.29	8	2974973.txt	0.31
9	2974973.txt	0.33	3372565.txt	0.29	9	2133386.txt	0.30
10	3526860.txt	0.32	2820672.txt	0.28	10	3526860.txt	0.30
$k = 300$				$k = 325$			
1	1065341.txt	0.50	3507913.txt	0.35	1	1065341.txt	0.48
2	2247119.txt	0.46	3272557.txt	0.31	2	2247119.txt	0.46
3	3842271.txt	0.34	2975002.txt	0.31	3	3842271.txt	0.33
4	270621.txt	0.33	2820672.txt	0.31	4	3260316.txt	0.31
5	3512896.txt	0.32	2247119.txt	0.30	5	270621.txt	0.31
6	3260316.txt	0.31	2995072.txt	0.29	6	3489096.txt	0.31
7	2974973.txt	0.31	270621.txt	0.29	7	2873406.txt	0.31
8	1459118.txt	0.30	3368184.txt	0.29	8	1459118.txt	0.29
9	2873406.txt	0.30	3512896.txt	0.27	9	2133386.txt	0.29
10	3526860.txt	0.30	3586474.txt	0.26	10	3512896.txt	0.29
5000 documents							
$k = 350$				$k = 375$			
1	2247119.txt	0.47	3507913.txt	0.35	1	2247119.txt	0.47
2	3842271.txt	0.36	2247119.txt	0.33	2	3514209.txt	0.34

TABLE 10: Continued.

Tf-Idf		Log Entropy		Tf-Idf		Log Entropy	
File	Similarity	File	Similarity	File	Similarity	File	Similarity
3 3160874.txt	0.35	3272557.txt	0.31	3 2801992.txt	0.33	3603166.txt	0.31
4 270621.txt	0.34	3603166.txt	0.31	4 3842271.txt	0.33	3586474.txt	0.30
5 1065341.txt	0.33	270621.txt	0.30	5 1065341.txt	0.32	3272557.txt	0.30
6 3514209.txt	0.33	2975002.txt	0.30	6 3160874.txt	0.32	2975002.txt	0.30
7 2873406.txt	0.32	2820672.txt	0.29	7 2873406.txt	0.32	2820672.txt	0.29
8 3510383.txt	0.30	3284243.txt	0.28	8 3284243.txt	0.31	3507913.txt	0.29
9 3284243.txt	0.30	3435747.txt	0.27	9 270621.txt	0.31	3284243.txt	0.29
10 2801992.txt	0.29	3586474.txt	0.27	10 3489096.txt	0.30	3435747.txt	0.28
$k = 400$				$k = 425$			
1 2247119.txt	0.47	2247119.txt	0.33	1 2247119.txt	0.47	3272557.txt	0.31
2 3842271.txt	0.33	270621.txt	0.31	2 1065341.txt	0.34	270621.txt	0.31
3 3514209.txt	0.32	3272557.txt	0.30	3 3514209.txt	0.33	2247119.txt	0.30
4 2873406.txt	0.32	3507913.txt	0.29	4 3842271.txt	0.32	3507913.txt	0.29
5 2801992.txt	0.32	2975002.txt	0.28	5 2873406.txt	0.32	2975002.txt	0.28
6 1065341.txt	0.32	3435747.txt	0.28	6 2801992.txt	0.30	3435747.txt	0.28
7 3160874.txt	0.31	3586474.txt	0.28	7 3160874.txt	0.30	3603166.txt	0.27
8 270621.txt	0.30	2820672.txt	0.28	8 3501694.txt	0.30	2820672.txt	0.27
9 3489096.txt	0.30	3284243.txt	0.28	9 270621.txt	0.29	3368184.txt	0.26
10 3284243.txt	0.29	3603166.txt	0.27	10 2778347.txt	0.29	3586474.txt	0.26

FIGURE 10: Results of the similarity for relevant document 1: 1087494 in the use case of topiramate weight-loss using the subset of 5000 documents, twenty values of k , and two weighting schemes (TF-IDF and Log Entropy).

allows the indexing of text documents from a knowledge base faster than the standard LSA versions. In addition, the hLSA system allows the retrieval of relevant documents based on a query. The results we have found are feasible and promising.

The hLSA system is divided into three main stages: semantic space, dimensionality reduction, and information

retrieval. Acceleration of four times is shown in the semantic space stage, acceleration of eight times is shown in a dimensionality reduction stage, and in the information retrieval stage we do not find acceleration. However, the hLSA system is limited by the main and global memory of GPU and CPU. Therefore, in future work, we propose adding a multi-GPU functionality to increase the size of matrices to be processed.

We have succeeded with our main motivation of improving the execution time of Latent Semantic Analysis method through the use of heterogeneous architectures such as GPUs and multi-CPU. As shown in experimental tests, we have achieved overall acceleration of around eight times faster than standard LSA systems for five thousand documents.

We have retrieved documents using twelve subsets of different sizes from five hundred to five thousand documents of the Open Access Subset of PubMed Central by using two weighting schemes: the Log Entropy and the Term Frequency-Inverse Document Frequency and twenty different values of k ranging from 25 to 250 to prove the appropriate value with which the hLSA system is more accurate. Hence, the hLSA has achieved an accuracy of 88% with the use case of bipolar disorders, an accuracy of 56% with the use case of lupus disease, and an accuracy of 98% with the use case of topiramate weight-loss. Moreover, we can infer that in our experiments the Log Entropy scheme has a higher value of similarity in two out of three use cases over the TF-IDF scheme.

TABLE 11: Results of the information retrieval process in the hLSA system with use case of topiramate weight-loss, using Log Entropy and TF-IDF weighting scheme with 1500 documents and $k = 25, 50, 75$, and 100; 3500 documents and $k = 150, 175, 200$, and 225; 4000 documents and $k = 425, 450, 475$, and 500.

Tf-Idf		Log Entropy		Tf-Idf		Log Entropy			
File	Similarity	File	Similarity	File	Similarity	File	Similarity		
1500 documents									
k = 25				k = 50					
1	1087494.txt	0.97	1087494.txt	0.92	1	1087494.txt	0.96	1087494.txt	0.89
2	2838810.txt	0.94	3552983.txt	0.82	2	2362538.txt	0.86	3128260.txt	0.74
3	3720211.txt	0.94	3055095.txt	0.82	3	2063236.txt	0.85	3771300.txt	0.73
4	2246781.txt	0.94	3880815.txt	0.82	4	3845698.txt	0.85	3055095.txt	0.71
5	3337849.txt	0.93	3771300.txt	0.82	5	2719193.txt	0.85	3148993.txt	0.69
6	2918360.txt	0.93	3284015.txt	0.79	6	2246781.txt	0.84	2275478.txt	0.68
7	3519703.txt	0.93	3128260.txt	0.78	7	3800929.txt	0.84	2941806.txt	0.64
8	2719193.txt	0.93	3800929.txt	0.75	8	2360336.txt	0.83	3284015.txt	0.64
9	3191588.txt	0.93	1896169.txt	0.74	9	545962.txt	0.82	3540966.txt	0.62
10	3845698.txt	0.93	3617973.txt	0.74	10	3095993.txt	0.82	3005214.txt	0.62
k = 75				k = 100					
1	1087494.txt	0.94	1087494.txt	0.90	1	1087494.txt	0.93	1087494.txt	0.90
2	2362538.txt	0.83	3128260.txt	0.65	2	2246781.txt	0.77	3128260.txt	0.61
3	2063236.txt	0.81	3055095.txt	0.59	3	2063236.txt	0.77	3055095.txt	0.60
4	2719193.txt	0.80	3148993.txt	0.56	4	3128260.txt	0.75	3650299.txt	0.54
5	2246781.txt	0.80	3098785.txt	0.56	5	2719193.txt	0.75	3771300.txt	0.52
6	3128260.txt	0.80	3650299.txt	0.55	6	2362538.txt	0.75	3443410.txt	0.49
7	3800929.txt	0.78	3617973.txt	0.55	7	2275478.txt	0.74	2275478.txt	0.48
8	545962.txt	0.78	3540966.txt	0.54	8	2360336.txt	0.73	3540966.txt	0.48
9	3720211.txt	0.77	2275478.txt	0.54	9	3720211.txt	0.72	1434505.txt	0.47
10	3112026.txt	0.77	3771300.txt	0.54	10	3800929.txt	0.72	3337849.txt	0.47
3500 documents									
k = 150				k = 175					
1	1087494.txt	0.90	1087494.txt	0.84	1	1087494.txt	0.90	1087494.txt	0.84
2	2001661.txt	0.72	3128260.txt	0.48	2	2001661.txt	0.69	3443410.txt	0.47
3	3857881.txt	0.70	3443410.txt	0.48	3	3857881.txt	0.65	3128260.txt	0.45
4	3855068.txt	0.68	3650299.txt	0.45	4	2275478.txt	0.64	3650299.txt	0.41
5	2275478.txt	0.67	3055095.txt	0.42	5	3855068.txt	0.64	3420720.txt	0.39
6	3720211.txt	0.67	2988027.txt	0.39	6	3720211.txt	0.64	3771300.txt	0.38
7	2246781.txt	0.66	1434505.txt	0.39	7	3065878.txt	0.64	3055095.txt	0.38
8	3065878.txt	0.66	3420720.txt	0.39	8	3196922.txt	0.63	3286846.txt	0.38
9	2063236.txt	0.65	3771300.txt	0.39	9	2246781.txt	0.61	3022769.txt	0.38
10	2887190.txt	0.65	3751340.txt	0.37	10	2063236.txt	0.61	1434505.txt	0.38
k = 200				k = 225					
1	1087494.txt	0.89	1087494.txt	0.85	1	1087494.txt	0.89	1087494.txt	0.86
2	2001661.txt	0.64	3443410.txt	0.48	2	2001661.txt	0.59	3443410.txt	0.48
3	3065878.txt	0.60	3650299.txt	0.41	3	3128260.txt	0.56	1434505.txt	0.41
4	2275478.txt	0.60	3128260.txt	0.40	4	3720211.txt	0.56	3650299.txt	0.40
5	3128260.txt	0.59	2988027.txt	0.39	5	2275478.txt	0.55	3286846.txt	0.39
6	3855068.txt	0.58	3286846.txt	0.39	6	3855068.txt	0.55	3128260.txt	0.39
7	3720211.txt	0.57	3540966.txt	0.37	7	3065878.txt	0.54	3540966.txt	0.38
8	3196922.txt	0.57	1434505.txt	0.36	8	2887190.txt	0.53	2988027.txt	0.36
9	2887190.txt	0.57	3420720.txt	0.36	9	2063236.txt	0.53	3592507.txt	0.34
10	2063236.txt	0.57	3065878.txt	0.35	10	3448379.txt	0.51	3065878.txt	0.33
4000 documents									
k = 425				k = 450					
1	1087494.txt	0.89	1087494.txt	0.86	1	1087494.txt	0.89	1087494.txt	0.86
2	2350122.txt	0.47	3443410.txt	0.36	2	2350122.txt	0.44	3650299.txt	0.36

TABLE 11: Continued.

Tf-Idf		Log Entropy		Tf-Idf		Log Entropy	
File	Similarity	File	Similarity	File	Similarity	File	Similarity
3 2362551.txt	0.44	1434505.txt	0.36	3 2362551.txt	0.42	3443410.txt	0.35
4 2001661.txt	0.42	3650299.txt	0.35	4 3065878.txt	0.40	1434505.txt	0.35
5 3065878.txt	0.41	3286846.txt	0.31	5 3530724.txt	0.40	3128260.txt	0.31
6 1832214.txt	0.40	3128260.txt	0.30	6 3650299.txt	0.40	2350122.txt	0.30
7 3530724.txt	0.39	2350122.txt	0.29	7 1832214.txt	0.39	3286846.txt	0.30
8 3592507.txt	0.39	3540966.txt	0.27	8 2001661.txt	0.39	2750184.txt	0.25
9 3857881.txt	0.39	2750184.txt	0.26	9 3592507.txt	0.39	3540966.txt	0.24
10 1831765.txt	0.39	293436.txt	0.24	10 1831765.txt	0.37	293436.txt	0.23
$k = 475$				$k = 500$			
1 1087494.txt	0.89	1087494.txt	0.87	1 1087494.txt	0.89	1087494.txt	0.87
2 2350122.txt	0.44	3650299.txt	0.37	2 2350122.txt	0.45	3650299.txt	0.37
3 3650299.txt	0.40	3443410.txt	0.34	3 3650299.txt	0.42	3443410.txt	0.35
4 2362551.txt	0.40	1434505.txt	0.33	4 3530724.txt	0.40	1434505.txt	0.34
5 3530724.txt	0.39	3128260.txt	0.31	5 2362551.txt	0.38	3128260.txt	0.30
6 3065878.txt	0.38	3286846.txt	0.30	6 3065878.txt	0.37	3286846.txt	0.29
7 1832214.txt	0.38	2350122.txt	0.30	7 2001661.txt	0.36	2350122.txt	0.29
8 2001661.txt	0.37	2750184.txt	0.25	8 3592507.txt	0.36	2750184.txt	0.24
9 3592507.txt	0.36	293436.txt	0.23	9 1832214.txt	0.36	3181688.txt	0.23
10 1831765.txt	0.36	2988027.txt	0.23	10 3286846.txt	0.35	293436.txt	0.23

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Universidad Politécnica Salesiana (UPS) through the research group of Cloud Computing, Smart Cities & High Performance Computing (GIHP4C) and the Sciences Research Council (CONACyT) through the Research Project no. 262756.

References

- [1] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the Association for Information Science and Technology*, vol. 41, no. 6, pp. 391–407, 1990.
- [2] T. K. Landauer and S. T. Dumais, "A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge," *Psychological Review*, vol. 104, no. 2, pp. 211–240, 1997.
- [3] T. J. Connolly, V. D. Veksler, and W. D. Gray, "Predicting Interest: Another Use for Latent Semantic Analysis," in *Proceedings of the Eighth International Conference on Cognitive Modeling*, 2009.
- [4] B. Li and K. Wang, "Computer aided diagnosis semantic model for the report of medical image via LDA and LSA," in *Proceedings of the 2011 IEEE International Symposium on IT in Medicine and Education, ITME 2011*, pp. 699–703, chn, December 2011.
- [5] A. Kontostathis, K. Reynolds, A. Garron, and L. Edwards, "Detecting cyberbullying: Query terms and techniques," in *Proceedings of the 3rd Annual ACM Web Science Conference, WebSci 2013*, pp. 195–204, fra, May 2013.
- [6] P. Zhang, Y. Zhang, T. Thomas, and S. Emmanuel, "Moving people tracking with detection by latent semantic analysis for visual surveillance applications," *Multimedia Tools and Applications*, vol. 68, no. 3, pp. 991–1021, 2014.
- [7] J. Wang, J. Peng, and O. Liu, "A classification approach for less popular webpages based on latent semantic analysis and rough set model," *Expert Systems with Applications*, vol. 42, no. 1, pp. 642–648, 2015.
- [8] W. Zhang, T. Yoshida, and X. Tang, "TFIDF, LSI and multi-word in information retrieval and text categorization," in *Proceedings of the 2008 IEEE International Conference on Systems, Man and Cybernetics, SMC 2008*, pp. 108–113, sgp, October 2008.
- [9] S. T. Dumais, "Latent Semantic Analysis," *Annual Review of Information Science and Technology*, vol. 38, pp. 189–230, 2004.
- [10] J. R. Herrera-Morales and L. I. Barbosa-Santillán, "Analysis of Medical Publications with Latent Semantic Analysis Method," in *Proceedings of the The Third International Conference on Advances in Information Mining and Managament*, pp. 1–6, 2013.
- [11] B. Rosario, "Latent semantic indexing: An overview," *Techn rep INFOSYS*, vol. 240, pp. 1–16, 2000.
- [12] J. M. Cavanagh, T. E. Potok, and X. Cui, "Parallel latent semantic analysis using a graphics processing unit," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pp. 2505–2510, ACM Press, NY, USA, 2009.
- [13] S. Vigna, "Distributed, Large-Scale Latent Semantic Analysis by Index Interpolation," in *Proceedings of the 3rd International ICST Conference on Scalable Information Systems*, Vico Equense, Italy, June 2008.
- [14] G. A. León-Paredes and L. I. Barbosa-Santillán, "A Heterogeneous System Based on Latent Semantic Analysis (hLSA)," in *Proceedings of GPU Technology Conference, 2017*, http://on-demand.gputechconf.com/gtc/2017/posters/images/1920x1607/GTC.2017_Algorithms.AL.11.P7175_WEB.png.

- [15] B. Pincombe, “Comparison of human and latent semantic analysis (lsa) judgements of pairwise document similarities for a news corpus,” Tech. rep., Defence Science and Technology Organisation Salisbury (Australia) Info Sciences Lab, 2004.
- [16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, Md, USA, 4th edition, 2013.
- [17] S. C. Deerwester, S. T. Dumais, G. W. Furnas et al., “Computer information retrieval using latent semantic structure,” *US Patent 4,839,853*, 1989.

Hindawi

Submit your manuscripts at
<https://www.hindawi.com>