University of the Philippines Manila

College of Arts and Sciences

Department of Physical Sciences and Mathematics

A Text Retrieval System Using Latent Semantic Analysis

A Special Problem

In Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science in Computer Science

Submitted by:

Editha Mojar

The special problem entitled “A Text Retrieval System Using Latent Semantic Analysis”, prepared and submitted by Editha Mojar in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science, has been examined and is recommended for acceptance.

Prof. Gregorio B. Baes
Adviser

Examiners	**Approved**	**Disapproved**
1. Prof. Gregorio B. Baes		
2. Prof. Ma. Sheila A. Magboo		
3. Dr. Vincent Peter C. Magboo		
4. Ms. Celina D.G. Umagtang		
5. Mr. Philip S.D. Zamora		

Date

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Vincent Peter C. Magboo, M.D., M.S.
Unit Head
Mathematical and Computing Science Unit
Department of Physical Sciences and Mathematics
University of the Philippines Manila

Harry Engle, M.S.
Chair
Department of Physical Sciences and Mathematics
University of the Philippines Manila

Marilou G. Nicolas, Ph.D.
Dean
College of Arts and Sciences

TABLE OF CONTENTS

to

GOD

“who was

and is

and is to come”

ABSTRACT

A text retrieval system that uses the Latent Semantic Analysis for indexing is developed. A collection of 106 documents are represented as vectors in a 377-dimensional term space. The number of dimensions corresponds to the number of extracted content words found in all the document titles in the database. The 377 by 106 matrix representing the entire data set is decomposed using singular value decomposition and the resulting matrices are truncated to 10 orthogonal factors. The recombination of the truncated matrices forms the basis for the computation of the distances of each document from a query vector obtained by treating a query as a pseudo-document. Results indicate that indexing using LSA is promising tool for improving retrieval.

I. INTRODUCTION

A. Background of the Study

Current information retrieval systems make use of keyword matching which heavily depends upon exact matches between query words and words in document titles and/or document abstracts [1]. Typically, a result set is created by checking to see whether a document contains certain keywords or not [2]. This method, however, fails to retrieve relevant materials that do not contain words in users' queries [1]. Furthermore, search engines usually return documents containing one or more query words even if these documents are not relevant to the user. Basing search results on keyword matching alone results in poor relevancy so that the desired documents may not appear near the top of the list [3].

In order to better document retrieval, there is a need to come up with a system that analyzes the latent semantic structure in the relationships between terms and documents. This process is known as the Latent Semantic Analysis (LSA) with the truncated Singular Value Decomposition (SVD) as model. This is a purely mathematical approach in finding the correlation among terms and documents in a given collection [2, 4] where documents are represented as vectors in a multidimensional term space [1, 5-7].

B. Statement of the Problem

The failure of the keyword matching method is much due to polysemy[1] and synonymy[2] [6]. Polysemy is illustrated by the fact that a single word may have more that one distinct meaning. This factor is accountable for poor precision[3] [7]. On average, about 50% of retrieved relevant information will be relevant [8, 9]. Synonymy is explained by Dumais' observation that the same word is used by two people to describe an object only 10 to 20% of the time [6, 7]. Deerwester et al. says this factor brings about low recall[4]. On average, only about 20% of the available relevant information is retrieved [8, 9].

[1] The many ways of defining a concept. *See* Section III-E.
[2] The many ways of referring to a concept. *See* Section III-E.
[3] The fraction of returned documents that is relevant. *See* Section III-E.
[4] The fraction of relevant documents that is retrieved. *See* Section III-E.

A third, more technical, factor that makes keyword matching in texts fail is that they use models such as Boolean, standard vector and probabilistic that treat words as if they are independent [7] when, in fact, they are not [1, 7].

C. **Objectives of Dealing with the Proposed Problem**

The main objective of the study is to create a document retrieval system that is capable of both high precision and high recall through a model that estimates term-to-document similarity with the use of the Latent Semantic Analysis.

Specifically, it aims to create a retrieval system that will:

a. enable users to add, modify and delete document titles in database;
b. generate an index following the Singular Value Decomposition with term-by-document matrix as initial input;
c. allow queries on document titles and return search results based on Latent Semantic Analysis.

D. **Significance of the Study**

Since retrieval is based on context as “learned” by the system through the discovery of implicit yet indirect correlation (or “latent semantics”), it is anticipated that information retrieval will improve.

Indexing using LSA has been known to be attractive in systems where high recall is needed such as data mining, trouble reports, law or medicine research; when document texts are short like figure captions, multimedia information or advertisements, or matching new problems against a database of existing trouble reports; when user queries or texts are noisy (e.g. occasional wrong spellings); or when there is a need to retrieve information in multiple languages [8].

E. Scope and Limitations

A database system where administrators are distinguished from ordinary users is not intended. All operations including `AddDocument`, `ModifyDocument` and `DeleteDocument` and all menus are accessible to any user. This application does not require user passwords and user information.

The system will provide only information that comprises a document record: 1) document filename which is automatically generated during addition of document title and 2) document title which also serves as document content. Access to actual documents is beyond the scope of the project.

The application does not try to deal with speed of retrieval. The SVD involves computations on matrices which may cause retrieval speed to be compromised. The size of the matrices greatly depends upon the total number of content words[5] in the entire collection of documents. Therefore, a set of 100 titles of technical reports will comprise the entire collection set. The size of the database may be increased depending on the availability of storage.

The filtering process, which is responsible for picking the content words in a document title or query, is language-specific. If a word in the title of the document is not included in the list of stop words[6] it will be considered a content word. Since the list of stop words includes English words only, the Spanish conjunction *con*, for instance, will pass through the filtering process as a content word.

II. REVIEW OF RELATED LITERATURE

A study was conducted by Deerwester et al. involving a collection of 1033 medical abstracts as documents. Their automatic indexing resulted in 5823 indexing terms. They used a 100-factor Singular Value

[5] *See* Section III-E.
[6] *See* Section III-E.

Decomposition of the 5823 by 1033 matrix. Retrieval effectiveness was evaluated against the 30 queries available with the dataset. Refinements such as stemming, phrases, term-weighting and Boolean combinations have been avoided to better evaluate the performance of LSI. The results of queries using LSI were compared against three other methods: 1) ordinary keyword matching which uses a table of word occurrence in documents within the collection; 2) SMART system that indexes using a stop list of common words, full stemming, and raw term frequencies; and 3) Voorhees systems that uses vector retrieval system with extended Boolean queries. Evaluation was done by measuring precision of retrieval at different levels of recall. The results showed that the average difference in precision between LSI and the term matching method was .06 representing 13% improvement over raw term matching. They have also found out that LSI performed more impressively at higher recall levels where precision is normally low. [7]

In another experiment by Foltz, users judged articles on how interesting they were. Based on these judgments LSI predicted whether new articles would be judged interesting. LSI improved prediction performance over keyword matching an average of 13% and showed a 26% improvement in precision over presenting articles in the order received. The results indicated that user preferences for articles tend to cluster based on the semantic similarities between articles. [6]

LSI was used by Telcordia Information Superstore, a site where Telcordia products such as documents, training and services can be purchased. LSI was used over the abstracts associated with the products in the store. When a customer adds an item to his/hers shopping basket, a certain number (based on a threshold) of the nearest neighbors in the space to the abstract of the new item in the basket item are suggested to the customer as items he or she might find useful. An empirical study of its effectiveness indicated that 8% of the items added to baskets in orders were LSI recommendations which resulted in 7% revenue increase. [8]

Dunn et al. aimed to compare traditional methods of scoring the Logical Memory Test of the Wechsler Memory Scale-III with a new method using LSA. A sample of 72 elderly individuals of which 14 were classified as cognitively impaired were administered the Logical Memory test. The results indicated that LSA was at least as valid and sensitive as traditional measures. LSA may serve as an improved measure of prose recall as partial correlations between prose recall measures and measures of cognitive function showed that LSA explained all the relationship between Logical Memory and general cognitive function. [5]

Other applications of Latent Semantic Analysis are automatic cross-language information retrieval and information filtering [9]. This mechanism has also been used to successfully simulate child vocabulary learning [4].

While Latent Semantic Analysis using singular value decomposition is strictly a mathematical approach [2] and uses no prior linguistic or perceptual knowledge about a certain vocabulary [4], some of the preparatory work that might be considered before indexing the documents are very language-specific like stemming. Stemming is a process that removes suffixes or common endings in words. This is because morphology of a language may cause inconsistencies to retrieval [2].

Another process that may be used is weighting. Words in a given document might be considered more meaningful than the others based on a number of criteria which might include frequency of a word in a document and/or frequency of a word in the whole document collection. Unlike stemming, it is not language-specific and does not require any linguistic knowledge to calculate the weights. The process is not of critical importance to LSI but is a potentially sound mechanism to improve accuracy of retrieval. [2]

III. CONCEPTUAL THEORETICAL FRAMEWORK

A. Keyword Matching

Regular keyword matching approach is based on the idea that a document contains a given word or it does not. The results are achieved by matching certain keywords with words contained in documents. Documents that do not contain these keywords are not regarded while the rest are ranked and returned to the user. This inexistence of interdependence among the documents results from the difference between the words searchers use and the words by which the information they search is indexed [2, 3].

To illustrate this problem a fictional matrix of documents by terms is shown below:

	biometric	computer	digita l	information	recognition	security	speaker	system	REL	MATCH
Doc 1	x	x*				x*		x*	R	M
Doc 2		x*	x	x				x*		M
Doc 3	x				x		x		R	

Query: audio-based computer security system

Table 1. A document-by-term matrix.

Below the table is a fictional query that might have been passed against the database. An "R" in the column labeled REL (relevant) indicates that the user would have judged the document relevant to the query. Terms occurring in both the query and a document are indicated by an asterisk in the appropriate cell; an "M" in the MATCH column indicates that the document matches the query based on their keywords and would have been returned to the user. Documents 2 and 3 illustrate common classes of problems which the LSA deals with. Document 2 is a non-relevant document but it would be returned because it contains the words "computer" and "system" which are in the query. Document 3 is a relevant document but it would not be returned by merely matching the keywords because it does not contain any of the words in the query. In this example, the meaning conditioning term "audio-based" in the query is not found in the index. Therefore term overlap retrieval would be unlikely to omit document 2. [3]

B. Latent Semantic Analysis

LSA can be described as a process that infers the structure of relationships between articles and words [6]. This method is based on the idea that the observed term-document association data is unreliable and must be treated as a statistical problem. The assumption is that there exists some latent semantic structure in the data that is partially obscured by the diversity of word use with respect to retrieval [7]. Statistical techniques are used to estimate this latent semantic structure [4, 7].

LSA represents texts as vectors in a high-dimensional semantic space [1, 5-7] which reflects the major associative patterns in the data while ignoring some of the smaller variations that may be caused by diversities in word usage [6, 7]. Similar texts tend to cluster in this multidimensional term space [1, 6].

There are several models that have been previously explored for addressing retrieval problems. Deerwester et al., in this search, restricted consideration to proximity models such as hierarchical, partition and overlapping clusterings; ultrametric and additive trees; and factor-analytic and multidimensional models. The idea behind document clustering is that two documents are more similar if they share more terms. The set of data is analyzed to find a hierarchical structure. The model, however, is believed to failing to grasp the semantic richness of document sets. It does not allow cross-classifications nor does it make use of sufficient number of parameters. Earlier factor analytic ventures, on the other hand failed to represent both terms and documents in the same space. Such analyses handle one type of entity at a time. In his paper, Deerwester et al. proposed a two-mode factor analytic method based on the following set of criteria: 1) Representational power can be adjusted by choosing the number of dimensions; 2) Terms and documents are explicitly represented; and, 3) Fitting technique is efficient. [7]

Two-mode factor analysis was the only model that satisfied all these. This can be achieved by applying singular value decomposition where each term and document is represented by its vector of factor values [7]. SVD maintains as much information as possible about the relative distances between the document vectors, while compressing them down into a less number of dimensions. In this process, content words coincide with one another [2]. Noise from the original term-to-document matrix is minimized, revealing similarities that were latent in the document collection [1, 2]. Similar documents cluster, while dissimilar documents remain distant. This reductive mapping is what makes LSI seemingly intelligent to be able to correlate semantically related terms [2]. This is an exploitation of a property of natural language, namely, that related words tend to occur in many of the same contexts [2, 8].

C. Fundamental Concepts in Linear Algebra

1. Matrix

Definition 1. A **matrix** is a rectangular array of numbers. A matrix of size ***m*** × ***n*** is a matrix with *m* **rows** and *n* **columns**. (The number of rows always comes first.) An **element** of a matrix, denoted by a_{ij} is the entry in the *i*th row and *j*th column. [10]

$$\begin{bmatrix} a_{11} & a_{12} & .. & a_{1j} & .. & a_{1n} \\ a_{21} & a_{22} & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ a_{i1} & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ a_{m1} & .. & .. & .. & .. & a_{mn} \end{bmatrix}$$

Example: Let $A = \begin{bmatrix} 1 & -2 & 1 \\ 0 & 2 & -8 \end{bmatrix}$

A has *2* rows and *3* columns. Then *A* is a matrix of size *2* × *3*.

2. Sum of Matrices

Definition 2. If *A* and *B* are *m* × *n* matrices, the **sum** *A* + *B* is the *m* × *n* matrix whose elements are the sums of the corresponding elements in *A* and *B*. [10]

Example: Let $A = \begin{bmatrix} 4 & 0 & 5 \\ -1 & 3 & 2 \end{bmatrix}$ and $B = \begin{bmatrix} 1 & 1 & 1 \\ 3 & 5 & 7 \end{bmatrix}$, then

$$A + B = \begin{bmatrix} 4 & 0 & 5 \\ -1 & 3 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 \\ 3 & 5 & 7 \end{bmatrix} = \begin{bmatrix} 5 & 1 & 6 \\ 2 & 8 & 9 \end{bmatrix}$$

3. Scalar Multiple

Definition 3. If r is a scalar and A is an $m \times n$ matrix, then the **scalar multiple** rA is the $m \times n$ matrix whose elements are r times the corresponding elements in A. [10]

Example: Let $A = \begin{bmatrix} 4 & 0 & 5 \\ -1 & 3 & 2 \end{bmatrix}$ and $r = 2$, then $2A = 2\begin{bmatrix} 4 & 0 & 5 \\ -1 & 3 & 2 \end{bmatrix} = \begin{bmatrix} 8 & 0 & 10 \\ -2 & 6 & 4 \end{bmatrix}$.

4. Product of Matrices

Definition 4. If A is an $m \times n$ matrix, and if B is an $n \times p$ matrix, then the **product** AB is the $m \times p$ matrix whose entry in row i and column j is the sum of the products of corresponding entries from row i of A and column j of B. If $(AB)_{ij}$ denotes the (i, j)-entry in AB, then

$$(AB)_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{in}b_{nj} \quad [10]$$

Example: Let $A = \begin{bmatrix} 2 & 3 \\ 1 & -5 \end{bmatrix}$ and $B = \begin{bmatrix} 4 & 3 & 6 \\ 1 & -2 & 3 \end{bmatrix}$, then $AB = \begin{bmatrix} 11 & 0 & 21 \\ -1 & 13 & -9 \end{bmatrix}$.

5. Diagonal Matrix

Definition 5. Let A be an $n \times n$ matrix. The diagonal entries of in $A = [a_{ij}]$ are those with equal row and column indices, a_{11}, a_{22}, a_{33},...a_{mn}. Then if the nondiagonal entries of A are all zero, A is a **diagonal matrix**. [10]

Example: Let $A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 4 \end{bmatrix}$

A is an $n \times n$ matrix and the diagonal entries are all zero. Therefore, A is a diagonal matrix.

6. The Identity Matrix

Definition 6. An **identity matrix**, denoted by I, is a *diagonal matrix* whose diagonal entries are all 1's. [10]

Example: Let $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

A is a diagonal matrix whose diagonal entries are all 1's. Therefore, A is an identity matrix.

7. Transpose of a Matrix

Definition 7. Let A be an $m \times n$ matrix. The **transpose** of A is the $n \times m$ matrix, denoted by A^T, whose columns are formed from the corresponding rows of A. [10]

Example: Let $A = \begin{bmatrix} -5 & 2 \\ 1 & -3 \\ 0 & 4 \end{bmatrix}$

The rows of A form the corresponding columns of its transpose. Therefore, $A^T = \begin{bmatrix} -5 & 1 & 0 \\ 2 & -3 & 4 \end{bmatrix}$.

8. Properties of the Matrix Transpose

Theorem 1. Let A and B be matrices such that AB is defined. Then, [10]

1. $(A^T)^T = A$
2. $(AB)^T = B^T A^T$

9. Symmetric Matrices

Definition 8. Let A be an $n \times n$ matrix. If $A^T = A$, then A is **symmetric**. [10]

Example: Let $A = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 5 & 8 \\ 0 & 8 & 7 \end{bmatrix}$, then $A^T = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 5 & 8 \\ 0 & 8 & 7 \end{bmatrix}$.

A is symmetric because $A^T = A$.

10. **The Symmetry of A^TA**

A^TA is symmetric because $A^TA = (A^TA)^T$. The following shows proof for this: [10]

$(A^TA)^T = A^T(A^T)^T$ by Theorem 1.2

$= A^TA$ by Theorem 1.1

11. Inverse of a Matrix

Definition 9. Let A be an $n \times n$ matrix. Then if there is another $n \times n$ matrix C such that

$$AC = I \quad \text{and} \quad CA = I$$

Then A is **invertible**; C is the **inverse** of A and is denoted by A^{-1}. [10]

Example: Let $A = \begin{bmatrix} 2 & 5 \\ -3 & -7 \end{bmatrix}$ and $C = \begin{bmatrix} -7 & -5 \\ 3 & 2 \end{bmatrix}$, then

$$AC = \begin{bmatrix} 2 & 5 \\ -3 & -7 \end{bmatrix}\begin{bmatrix} -7 & -5 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{and } CA = \begin{bmatrix} -7 & -5 \\ 3 & 2 \end{bmatrix}\begin{bmatrix} 2 & 5 \\ -3 & -7 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Therefore, A is invertible and $A^{-1} = C$.

12. Orthogonal Matrices

Definition 10. Let A be an $n \times n$ matrix. If A is *invertible* such that $A^{-1} = A^{T}$, then it is **orthogonal**. [10]

Example: Let $A = \begin{bmatrix} 3/\sqrt{11} & -1/\sqrt{6} & -1/\sqrt{66} \\ 1/\sqrt{11} & 2/\sqrt{6} & -4/\sqrt{66} \\ 1/\sqrt{11} & 1/\sqrt{6} & 7/\sqrt{66} \end{bmatrix}$, then

$A^{-1} = \begin{bmatrix} 3/\sqrt{11} & 1/\sqrt{11} & 1/\sqrt{11} \\ -1/\sqrt{6} & 1/\sqrt{11} & 1/\sqrt{6} \\ -1/\sqrt{66} & -1/\sqrt{66} & 7/\sqrt{66} \end{bmatrix}$ and $A^{T} = \begin{bmatrix} 3/\sqrt{11} & 1/\sqrt{11} & 1/\sqrt{11} \\ -1/\sqrt{6} & 1/\sqrt{11} & 1/\sqrt{6} \\ -1/\sqrt{66} & -1/\sqrt{66} & 7/\sqrt{66} \end{bmatrix}$.

Since $A^{-1} = A^{T}$, then A is orthogonal.

13. Orthogonal Diagonalizability

Definition 11. A matrix A is said to be **orthogonally diagonalizable** if there is an orthogonal matrix P (with $P^{-1} = P^{T}$) and a diagonal matrix D such that

$$A = PDP^{T} = PDP^{-1} \qquad [10]$$

Example: Let $A = \begin{bmatrix} 3 & -2 & 4 \\ -2 & 6 & 2 \\ 4 & 2 & 3 \end{bmatrix}$, $P = \begin{bmatrix} -1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 0 & 2/\sqrt{6} & 1/\sqrt{3} \end{bmatrix}$ and $D = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 3 \end{bmatrix}$,

then $P^{T} = \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{6} & -1/\sqrt{6} & 2/\sqrt{6} \\ 6/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \end{bmatrix}$ and $P^{-1} = \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{6} & -1/\sqrt{6} & 2/\sqrt{6} \\ 6/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \end{bmatrix}$.

Because $P^{-1} = P^{T}$, P is orthogonal. Also,

$$A = PDP^{T} = PDP^{-1} = \begin{bmatrix} -1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 1/\sqrt{2} & -1/\sqrt{6} & 1/\sqrt{3} \\ 0 & 2/\sqrt{6} & 1/\sqrt{3} \end{bmatrix} \begin{bmatrix} 8 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{6} & -1/\sqrt{6} & 2/\sqrt{6} \\ 6/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \end{bmatrix}$$

$$= \begin{bmatrix} 3 & -2 & 4 \\ -2 & 6 & 2 \\ 4 & 2 & 3 \end{bmatrix}$$

Therefore, A is orthogonally diagonalizable.

Note: Every symmetric matrix is orthogonally diagonalizable (by Spectral Theorem for Symmetric Matrices; *See* Theorem 3).

14. Vector

Definition 12. A matrix with only one column is a **vector**. An **element** in a vector, denoted by v_i, is the ith entry in the vector. [10]

$$\begin{bmatrix} v_1 \\ v_2 \\ .. \\ v_i \\ .. \\ v_m \end{bmatrix}$$

Example: Let $\mathbf{v} = \begin{bmatrix} 4 \\ 11 \\ 1 \end{bmatrix}$, then $\mathbf{v}$ is a vector with elements $v_1 = 4$, $v_2 = 11$ and $v_3 = 1$.

15. The Zero Vector

Definition 13. A vector whose entries are all zero is called a **zero vector** and is denoted by **0**. [10]

16. Inner Product

Definition 14. If **u** and **v** are $n \times 1$ vectors, then the **inner product** of **u** and **v**, denoted by **u • v**, is the

scalar $\mathbf{u} \bullet \mathbf{v} = \mathbf{u}^T\mathbf{v} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \mathbf{u}_1\mathbf{v}_1 + \mathbf{u}_2\mathbf{v}_2 + \dots + \mathbf{u}_n\mathbf{v}_n$. [10]

Example: Let $\mathbf{u} = \begin{bmatrix} 5 \\ 4 \\ 9 \\ 2 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 3 \end{bmatrix}$. Then their inner product $\mathbf{u} \bullet \mathbf{v} = \mathbf{u}^T\mathbf{v} = \begin{bmatrix} 5 & 4 & 9 & 2 \end{bmatrix}$

$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 3 \end{bmatrix} = (5 \times 0) + (4 \times 1) + (9 \times 1) + (2 \times 3) = 19.$

17. Properties of Dot Product

Theorem 2. Let **u** and **v** be $n \times 1$ vectors, and let c be a scalar. Then [10]

1 1. **u • v = u • v**

2 2. (c**u**) • **v** = c (**u** • **v**) = **u** • (c**v**)

18. Length of a Vector

Definition 15. The **length** of a vector **v** is the nonnegative scalar ||**v**|| defined by ||**v**|| = √(**v** •**v**) =

$\sqrt{v_1^{\ 2} + v_2^{\ 2} + \dots + v_n^{\ 2}}$ where $v_1, \dots, v_n$ are the entries of **v**. [10]

Example: Let $\mathbf{v} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 0 \end{bmatrix}$, then $||\mathbf{v}|| = \sqrt{1^2 + 2^2 + 2^2 + 0^2} = \sqrt{1+4+4+0} = \sqrt{9} = 3$.

19. Unit Vector

Definition 16. Vector whose length is 1 is called **unit vector**. [10]

Example: Let $\mathbf{v} = \begin{bmatrix} 1/3 \\ -2/3 \\ 2/3 \\ 0 \end{bmatrix}$, then

$$||\mathbf{v}|| = \sqrt{\left(\frac{1}{3}\right)^2 + \left(-\frac{2}{3}\right)^2 + \left(\frac{2}{3}\right)^2 + (0)^2} = \sqrt{\frac{1}{9} + \frac{4}{9} + \frac{4}{9} + 0} = \sqrt{\frac{9}{9}} = \sqrt{1} = 1$$

Since the length of **v** is 1, it is a unit vector.

20. Distance between Two Vectors

Definition 17. Let **u** and **v** be vectors in R^n, the **distance**, denoted by dist (**u**, **v**), between **u** and **v** is the *length* of the vector **u** – **v**. That is,

$$\text{dist}(\mathbf{u}, \mathbf{v}) = ||\mathbf{u} - \mathbf{v}|| \qquad [10]$$

Example: Let $\mathbf{u} = \begin{bmatrix} 7 \\ 1 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$, and $\mathbf{u} - \mathbf{v} = \begin{bmatrix} 7 \\ 1 \end{bmatrix} - \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \end{bmatrix}$, then the distance between **u** and **v** is $||\mathbf{u} - \mathbf{v}|| = \sqrt{4^2 + (-1)^2} = \sqrt{17}$.

21. Eigenvalues and Eigenvectors of a Matrix

Definition 18. Let A be an $n \times n$ matrix. The real number λ is called an eigenvalue of A if there exists a nonzero vector $\boldsymbol{X}$ in R^n such that

$$A\boldsymbol{X} = \lambda\boldsymbol{X}.$$

Every nonzero vector $\boldsymbol{X}$ satisfying above equation is called an eigenvector of A associated with the eigenvalue λ. [11]

Example: Let $A = \begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} 6 \\ -5 \end{bmatrix}$, then $A\mathbf{x} = \begin{bmatrix} 1 & 6 \\ 5 & 2 \end{bmatrix}\begin{bmatrix} 6 \\ -5 \end{bmatrix} = \begin{bmatrix} -24 \\ 20 \end{bmatrix} = -4\begin{bmatrix} 6 \\ -5 \end{bmatrix} =$ $-4\mathbf{x}$.

Thus $\mathbf{x}$ is an eigenvector of A corresponding to an eigenvalue -4.

Note: An $n \times n$ symmetric matrix A has n real eigenvalues (by Spectral Theorem for Symmetric Matrices; *See* Theorem 3.)

22. The Spectral Theorem for Symmetric Matrices

Theorem 3. An $n \times n$ symmetric matrix A has the following properties: [10]

1. A has n real eigenvalues.
2. The eigenvectors corresponding to the different eigenvalues are orthogonal.
3. A is orthogonally diagonalizable.

23. **The Eigenvectors of A^TA**

Let A be an $m \times n$ matrix and let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of A^TA, and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ be their respective associated unit eigenvectors. For $1 \le i \le n$,

$\|A\mathbf{v}_i\|^2 = (A\mathbf{v}_i) \bullet (A\mathbf{v}_i)$ Definition 15

$= (A\mathbf{v}_i)^{\mathrm{T}}A\mathbf{v}_i$	Definition 14
$= \mathbf{v}_i^T A^T A\mathbf{v}_i$	Theorem 1.2
$= \mathbf{v}_i^T(\lambda_i\mathbf{v}_i)$	Since $\mathbf{v}_i$ is an eigenvector of A^TA
$= \lambda_i\,\mathbf{v}_i^T\mathbf{v}_i$	Theorem 2.2
$= \lambda_i\,(\mathbf{v}_i \bullet \mathbf{v}_i)$	Definition 14
$= \lambda_i\,\lVert\mathbf{v}_i\rVert^2$	Definition 15
$= \lambda_i$	Since $\mathbf{v}_i$ is a unit vector

Since $\|A\mathbf{v}_i\|^2 \geq 0$ $\forall i$, the eigenvalues of A^TA are all nonnegative. [10]

24. **Singular Values and Singular Vectors**

Definition 19. Let A be an $m \times n$ matrix and let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of A^TA, and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ be their respective associated unit eigenvectors. Then the **singular values** of A are the positive square roots of the eigenvalues of A^TA which are denoted by $\sigma_1, \ldots, \sigma_n$. [10]

Note: The eigenvalues of A^TA are all nonnegative (*See* Section 23). By renumbering, if necessary, we may assume that the eigenvalues are arranged so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$.

25. **Computation of Eigenvectors**

Assume λ is an eigenvalue of the $n \times n$ matrix A. The eigenvector X associated to λ is given by the matrix equation $AX = \lambda X$.

If $\boldsymbol{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$, the above matrix equation reduces to the algebraic system

$$a_{11}x_1 + a_{12}\, x_2 + \ldots + a_{1n}x_n = \lambda x_1$$

$$a_{21}x_1 + a_{22}\, x_2 + \ldots + a_{2n}x_n = \lambda x_2$$

$$\vdots$$

$$a_{n1}x_1 + a_{n2}\, x_2 + \ldots + a_{1n}x_n = \lambda x_n$$

which is equivalent to the system

$$\begin{aligned} &(a_{11}-\lambda)\, x_1 + a_{12}\, x_2 + \ldots + a_{1n}x_n = \lambda x_1 \\ &a_{21}x_1 + (a_{22}-\lambda)\, x_2 + \ldots + a_{2n}x_n = \lambda x_2 \qquad \text{or} \qquad (A - \lambda \boldsymbol{I})\boldsymbol{X} = \mathbf{0} \\ &\vdots \\ &a_{n1}x_1 + a_{n2}\, x_2 + \ldots + (a_{1n}-\lambda)\, x_n = \lambda x_n \end{aligned}$$

The nontrivial (i.e. nonzero) solutions of this homogenous system are the eigenvectors of A corresponding to λ. [12]

Example: Consider the matrix $\begin{bmatrix} 2 & -1 \\ 0 & 3 \end{bmatrix}$. Find the eigenvectors associated to the eigenvalue λ = 3.

Let $\boldsymbol{X}$ be an eigenvector associated to the eigenvalue λ = 3. Then we must have

$$(2\text{-}3)x_0 + \quad -x_1 \ = 0$$

$$0 \quad + (3\text{-}3)x_1 = 0$$

which reduces to the equation $-x_0 - x_1 = 0$, which yields $x_1 = -x_0$.

Therefore, we have $\boldsymbol{X} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} x_0 \\ -x_1 \end{bmatrix} = x_0 \begin{bmatrix} 1 \\ -1 \end{bmatrix}$. We now have all of the eigenvectors associated to the eigenvalue λ = 3. [12]

Also, if $\boldsymbol{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ is one such solution, then the unit eigenvector **u** corresponding to λ is given by

$$\mathbf{u} = (1/\|\boldsymbol{X}\|)\boldsymbol{X}.$$

If $\boldsymbol{X}$ is an eigenvector of A associated with λ (that is, $A\boldsymbol{X} = \lambda\boldsymbol{X}$) and k is any nonzero real number, then $A(k\boldsymbol{X}) = k(A\boldsymbol{X}) = k(\lambda\boldsymbol{X}) = \lambda(k\boldsymbol{X})$. [11]

Thus, $k\boldsymbol{X}$ is also an eigenvector of A associated with λ. [11, 12]

D. Singular Value Decomposition

The core of Latent Semantic Analysis is the singular value decomposition stated as follows:

Theorem 4. Any $m \times n$ rectangular matrix A can be factorized in the form $A = U \sum V^T$ [3, 10-11], where

U = an $m \times m$ matrix whose columns are called the left singular unit vectors;

$\sum$ = an $m \times n$ matrix whose diagonal entries are the first r singular values of A (where r is the smaller of m and n) and non-diagonal entries are zeros.

V = an $n \times n$ matrix whose columns are called the right singular unit vectors [10-11]

Let matrix1 be an $m \times n$ matrix representing the entire document title collection. Then the elements of matrix1 are defined by:

$$\text{matrix1}_{ij} = \begin{cases} 1 \text{ if word in } i\text{th row of content table is in } j\text{th document title} \\ 0 \text{ if word in } i\text{th row of content table is not in } j\text{th document title} \end{cases}$$

$\forall i\, 1 \le i \le m,$

$\forall j\, 1 \le j \le n,$

where m = the total number of content words in the collection;

n = the total number of documents in the collection.

Let evector1 be an $m \times 1$ vector representing the query. Then the elements of evector1 are defined by:

$$\text{evector}_i = \begin{cases} 1 \text{ if term in } i\text{th row of content table is in query} \\ 0 \text{ if term in } i\text{th row of content table is not in query} \end{cases}$$

$\forall i\, 1 \le i \le m,$

where m = the total number of content words in the collection.

The following is an example of calculating for the singular value decomposition of a matrix:

1. Let $A = \begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix}$

2. Compute for A^TA:

$$A^TA = \begin{bmatrix} 4 & 8 \\ 11 & 7 \\ 14 & -2 \end{bmatrix} \begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix} = \begin{bmatrix} 80 & 100 & 40 \\ 100 & 170 & 140 \\ 40 & 140 & 200 \end{bmatrix}$$

3. Get the eigenvalues of A^TA which are $\lambda_1 = 360$, and $\lambda_2 = 90$, and $\lambda_2 = 0$. The singular values are $6\sqrt{10}$,$-3\sqrt{10}$ and 0. The first r singular values ($r = 2$ since $r = \min\{2, 3\}$) form become the diagonal entries of $\sum$. Entries elsewhere are 0's:

$$\Sigma = \begin{bmatrix} 6/\sqrt{10} & 0 & 0 \\ 0 & 3/\sqrt{10} & 0 \end{bmatrix}$$

5. Corresponding unit eigenvectors are, respectively,

$$\mathbf{v}_1 = \begin{bmatrix} 1/3 \\ 2/3 \\ 2/3 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} -2/3 \\ -1/3 \\ 2/3 \end{bmatrix} \text{ and } \mathbf{v}_3 = \begin{bmatrix} 2/3 \\ -2/3 \\ 1/3 \end{bmatrix}$$

$\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$, are the right singular vectors of A.

The right singular vectors form the columns of V:

$$V = \begin{bmatrix} 1/3 & -2/3 & 2/3 \\ 2/3 & -1/3 & -2/3 \\ 2/3 & 2/3 & 1/3 \end{bmatrix}$$

6. Derive the left singular vectors.

$$\mathbf{u}_1 = \left(\frac{1}{\sigma_1}\right) A\mathbf{v}_1 = \frac{1}{6\sqrt{10}} \begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix} \begin{bmatrix} 1/3 \\ 2/3 \\ 2/3 \end{bmatrix} = \begin{bmatrix} 3/\sqrt{10} \\ 1/\sqrt{10} \end{bmatrix}$$

$$\mathbf{u}_2 = \left(\frac{1}{\sigma_2}\right)A\mathbf{v}_2 = \frac{1}{3\sqrt{10}}\begin{bmatrix} 4 & 11 & 14 \\ 8 & 7 & -2 \end{bmatrix}\begin{bmatrix} -2/3 \\ -1/3 \\ 2/3 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{10} \\ -3/\sqrt{10} \end{bmatrix}$$

The left singular vectors form the columns of *U*:

$$U = \begin{bmatrix} 3/\sqrt{10} & 1/\sqrt{10} \\ 1/\sqrt{10} & -3/\sqrt{10} \end{bmatrix}$$

7. Then,

$$A = U\sum V^T$$

$$= \begin{bmatrix} 3/\sqrt{10} & 1/\sqrt{10} \\ 1/\sqrt{10} & -3/\sqrt{10} \end{bmatrix}\begin{bmatrix} 3/\sqrt{10} & 0 & 0 \\ 0 & -3/\sqrt{10} & 0 \end{bmatrix}\begin{bmatrix} 1/3 & 2/3 & 2/3 \\ -2/3 & -1/3 & 2/3 \\ 2/3 & -2/3 & 1/3 \end{bmatrix}$$

E. Definition of Terms

- **Collection** – the set of all documents for indexing.
- **Content word** – any word that does not qualify for a stop word. *See* also **stop word**.
- **Index** – searchable catalogue of documents created by search engine software.
- **Polysemy** – the many ways of defining a concept. The word "push", for example may mean "a force" in Physics but may also be used to mean "place on the stack" in the context of data structures.
- **Precision** – the degree in which a search engine lists documents matching a query. It is the fraction of the number of returned relevant documents to the total number of returned documents. The more matching documents are listed, the higher the precision. For example, if only 4 are relevant out of 10 search results then precision is 40%.
- **Recall** – the degree in which a search engine returns all the matching documents in a collection. It is the fraction of the number of returned relevant documents to the total number of relevant documents. For example, if there are 20 relevant documents in the collection but only 10 of them are returned then recall is 50%.

- **Relevancy** – refers to how well a document provides the information a user is looking for as measured by the user.
- **Search engine** – software that searches an index and returns matches.
- **Stemming** – the process of removing affixes to extract the “stem” of words. For example, stemming allows a user to enter “swimming” and get back results for the stem word “swim”.
- **Stop word** – a conjunction, preposition, article or other character (e.g. ?, *, $) that appear often in documents yet alone may contain little meaning.
- **Synonymy** – the many ways of referring to a concept. In Biology, for example, the relationship where both parties benefit is referred to as “symbiosis” but others would simply call it “mutualism”.
- **Token** – a set of alphanumeric characters with no white spaces.

IV. DESIGN AND IMPLEMENTATION

The context-free diagram of the system is shown in Figure 1. The main player of the system is the user interfacing with the system and its database.

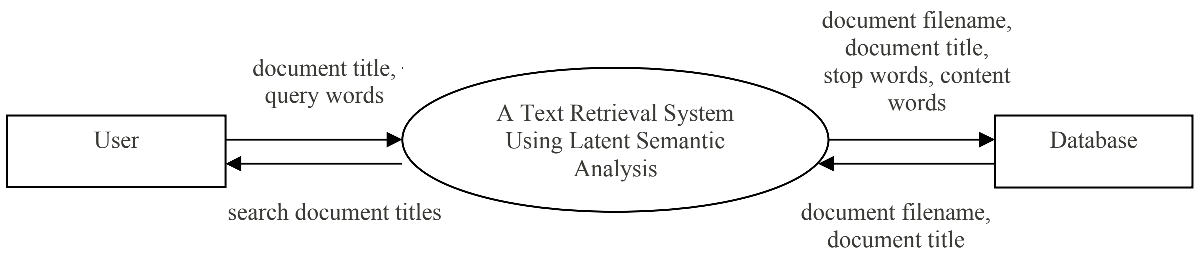

document title,
query words
User
A Text Retrieval System Using Latent Semantic Analysis
search document titles
document filename, document title, stop words, content words
Database
document filename, document title

Figure 1. Context-free Diagram, Text Retrieval System Using Latent Semantic Analysis

The database has the following three tables. The document table contains all the document records added in the database. It has two fields: the filename which is automatically generated by the system and the title which is supplied by the user. Table 2 shows the fields of the document table:

Field	**Type**
filename	Integer
Title	Text

Table 2. The document table, A Text Retrieval System Using Latent Semantic Analysis

The stop table is a list containing all the words that will be filtered out by the system for the latent semantic analysis. Table 3 shows the contents of the stop table:

Field	**Type**
stopword	Text

Table 3. The stop table, A Text Retrieval System Using Latent Semantic Analysis

The content table is a list containing all the words in the document titles in the document table that are not in the stop table. Table 4 shows the contents of the content table:

Field	**Type**
contentword	Text

Table 4. The content table, A Text Retrieval System Using Latent Semantic Analysis

These three tables have no common fields and are therefore independent from each other.

The top-level DFD of the system is shown in Figure 2. It has 4 main processes, namely: 1) Update Document Title, 2) Update Matrix, and 3) Submit Query.

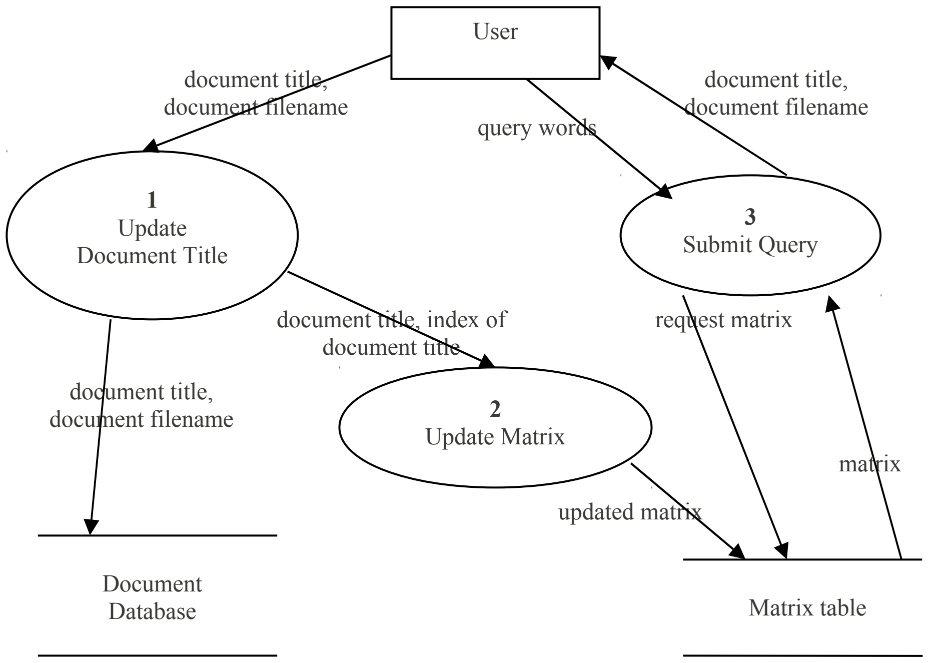
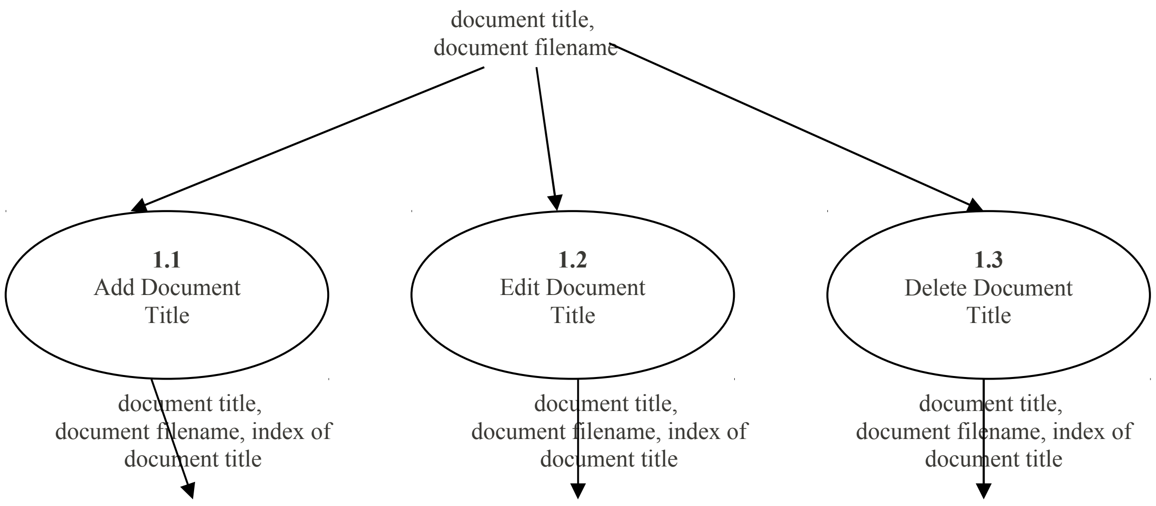

User
document title, document filename
query words
document title, document filename
1 Update Document Title
3 Submit Query
document title, index of document title
request matrix
document title, document filename
2 Update Matrix
matrix
updated matrix
Document Database
Matrix table

Figure 2. Top-Level Data Flow Diagram, A Text Retrieval System Using Latent Semantic Analysis

The Update Document Title process has three sub-processes: 1) Add Document Title, 2) Edit Document Title, and 3) Delete Document Title. Figure 3 shows the sub-explosion of the Update Document Title process.

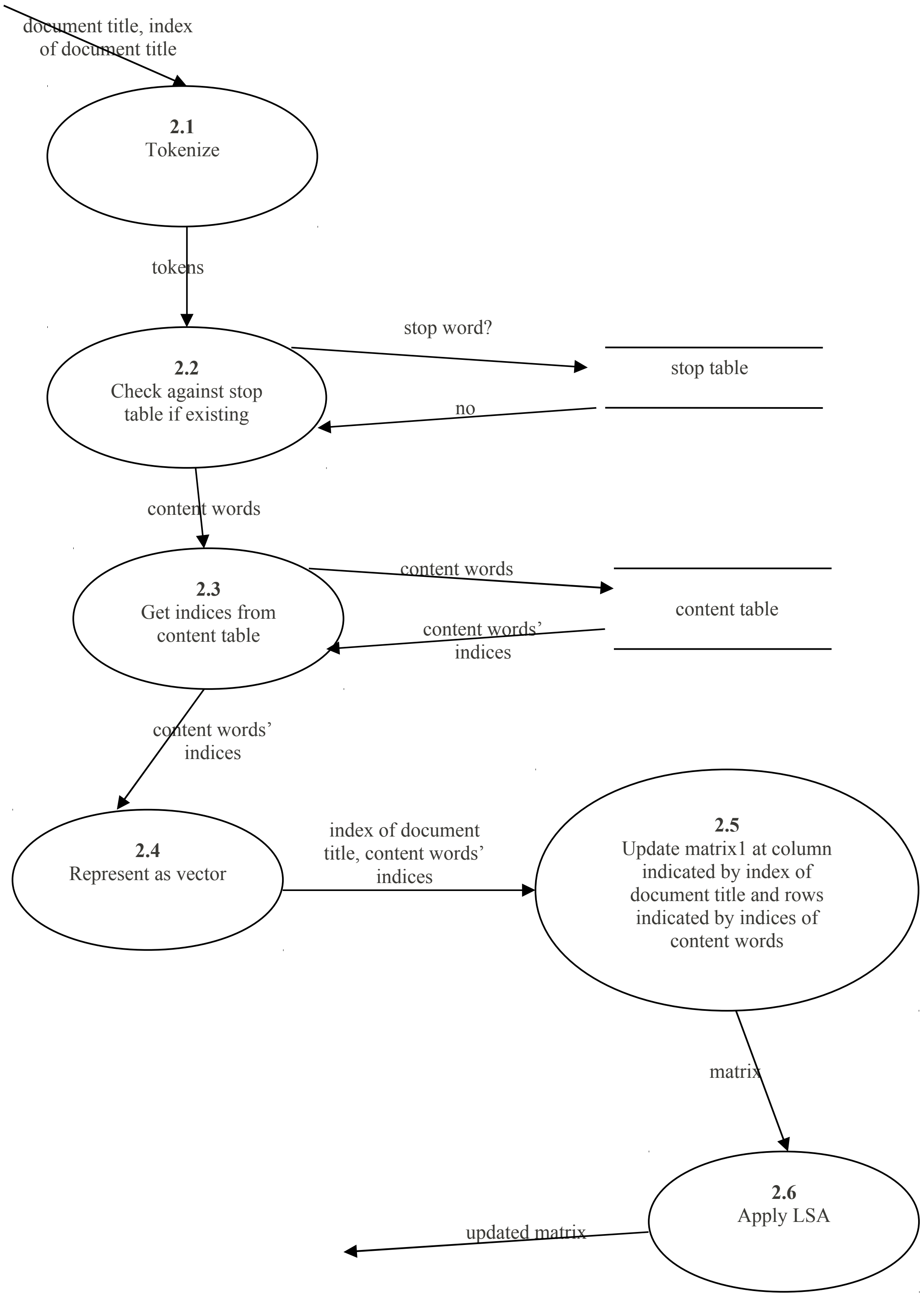


document title,
document filename
1.1
Add Document Title
1.2
Edit Document Title
1.3
Delete Document Title
document title, document filename, index of document title
document title, document filename, index of document title
document title, document filename, index of document title

Figure 3. Sub-explosion of Process Update Document Title,
A Text Retrieval System Using Latent Semantic Analysis

During the Add Document Title, the document title is added in the document table. For the Edit Document Title, the document title is replaced with the new document title supplied by the user. For the Delete Document Title, the document title selected by the user is deleted from the document table.

After the Update Document Title process, the document title is then processed to return a vector representation, which is used to update the matrix. This is done by first tokenizing the document title. Each token is looked-up in the stop table to determine whether it is a top word or not. If the token is not a stop word it is inserted in the content table. The corresponding indices of these content words in the content table are returned. These indices are used to represent the document as a vector by indicating which elements of the vector representation must be given a value of 1. The sub-explosion of the Update Matrix process is shown in figure 4.



document title, index of document title
2.1 Tokenize
tokens
2.2 Check against stop table if existing
stop word?
stop table
no
content words
2.3 Get indices from content table
content words
content table
content words' indices
content words' indices
2.4 Represent as vector
index of document title, content words' indices
2.5 Update matrix1 at column indicated by index of document title and rows indicated by indices of content words
matrix
2.6 Apply LSA
updated matrix

Figure 4. Sub-explosion of Process Update Matrix, A Text Retrieval System Using Latent Semantic Analysis

After the Update Matrix Process, it can be verified that the following holds true for each element in the matrix:

$$M_{ij} = \begin{cases} 1 \text{ if term in } i\text{th row is in } j\text{th document; or} \\ 0 \text{ otherwise} \end{cases}$$

$$\forall i \; 1 \le i \le m,$$
$$\forall i \; 1 \le i \le m,$$

where m = total number of content words in content table;

n = total number of documents in document table.

The resulting matrix in the Update Matrix process is applied the Latent Semantic Analysis. The size of the matrix is determined. Then the singular value decomposition is computed. The resulting matrices are truncated taking only the first k columns of U and V; and the first k columns and first k rows of Σ using k = round (.10 × n) + 1. They are recombined and the resulting matrix is stored. This matrix is of the same size as the original matrix. Figure 5 shows sub-explosion of Process Apply LSA.

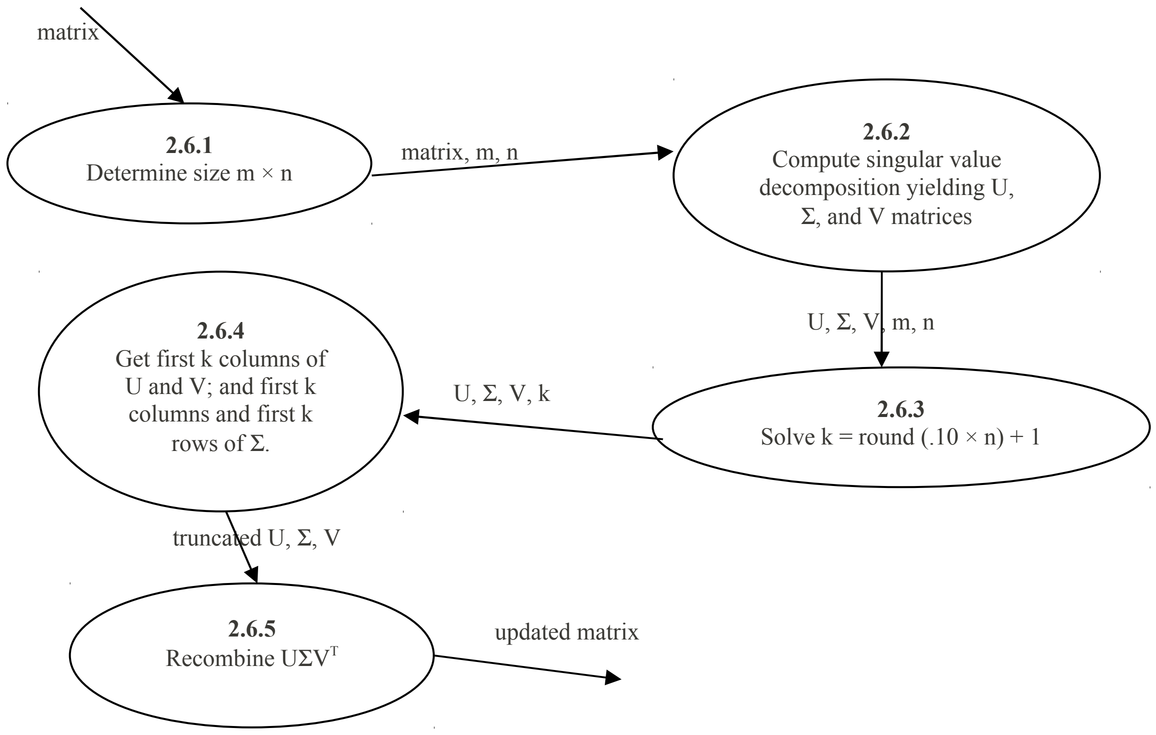

matrix
2.6.1
Determine size m × n
matrix, m, n
2.6.2
Compute singular value decomposition yielding U, Σ, and V matrices
U, Σ, V, m, n
2.6.3
Solve k = round (.10 × n) + 1
U, Σ, V, k
2.6.4
Get first k columns of U and V; and first k columns and first k rows of Σ.
truncated U, Σ, V
2.6.5
Recombine UΣV^T
updated matrix

Figure 5. Sub-explosion of Process Apply LSA, A Text Retrieval System Using Latent Semantic Analysis

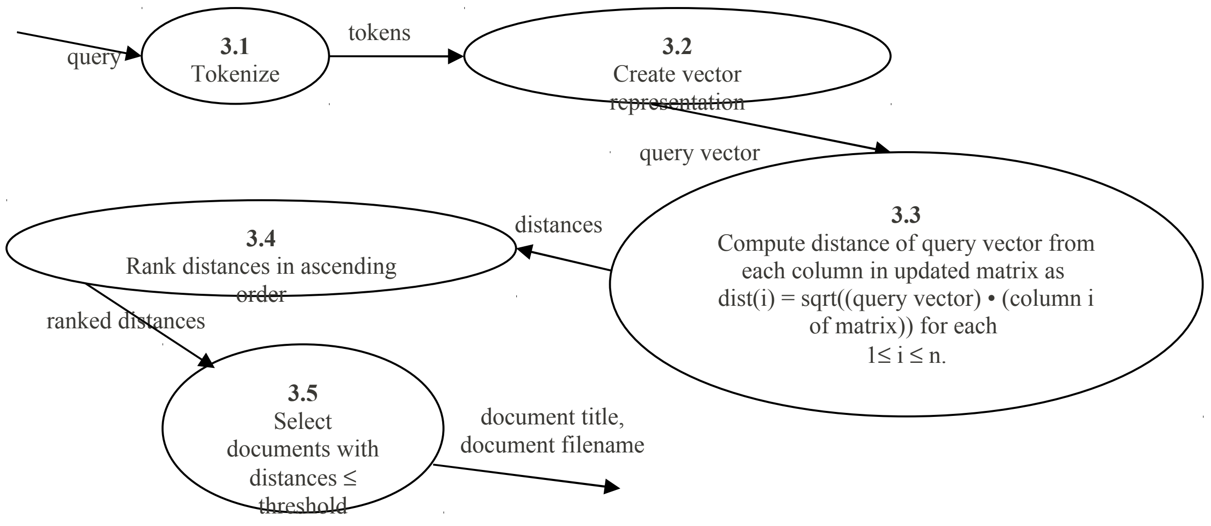
The updated matrix is stored for later use in the Submit Query process. Queries are processed by treating them as pseudo-documents. Since a query is composed of a group of words, it can be represented as a vector **v**, whose elements are defined as follows:

$$v_i = \begin{cases} 1 \text{ if word in } i\text{th row of content table is in query} \\ 0 \text{ otherwise} \end{cases}$$

$$\forall i \; 1 \leq i \leq m,$$

where m is the number of content words in the content table.

The first step for this process is to tokenize the query. Each token is looked up at the content table for its index. (Tokens that are not found in the content table are ignored.) These indices are used to represent the query as a vector by indicating which elements in the vector have a value of 1. If it happens that the entries in v are all 0's, meaning it doesn't have any content words existing in the collection, the query is not processed. The search results will simply display a message indicating that there was no match found. Otherwise, the distance of the query vector from each column in the updated matrix is computed. These documents whose distances have the lowest values not greater than the set threshold are selected for output as search results. Figure 6 shows the sub-explosion of the Submit Query process.



query
3.1
Tokenize
tokens
3.2
Create vector representation
query vector
3.3
Compute distance of query vector from each column in updated matrix as dist(i) = sqrt((query vector) • (column i of matrix)) for each 1≤ i ≤ n.
distances
3.4
Rank distances in ascending order
ranked distances
3.5
Select documents with distances ≤ threshold
document title, document filename

Figure 6. Sub-explosion of Process Apply LSA,
A Text Retrieval System Using Latent Semantic Analysis

The code for the main function was written using the Java programming language. Most mathematical functions ware written using Matlab which has predefined functions for operations such as the extraction of eigenvalues and the singular value decomposition. Microsoft Access was used to create and store the database. The JMatLink class, a third-party software, was added in the import list of the source code of the main java class to integrate the Matlab engine with the java application.

V. TECHNICAL ARCHITECTURE

The project is a standalone application that will run with the following:

- Windows XP
- Pentium II or higher
- The Java Virtual Machine
- Matlab 6.1 or better
- 256 MB RAM
- 1 MB disk space for the class, function and database files

VI. RESULTS

The main menu is shown in Figure 6. The application has four major functions. It allows the user to submit a query using Search, add a document title using Insert Document, edit a document title using Modify Document and delete a document using Delete Document. It also gives the user the option to close the program using **Quit**.

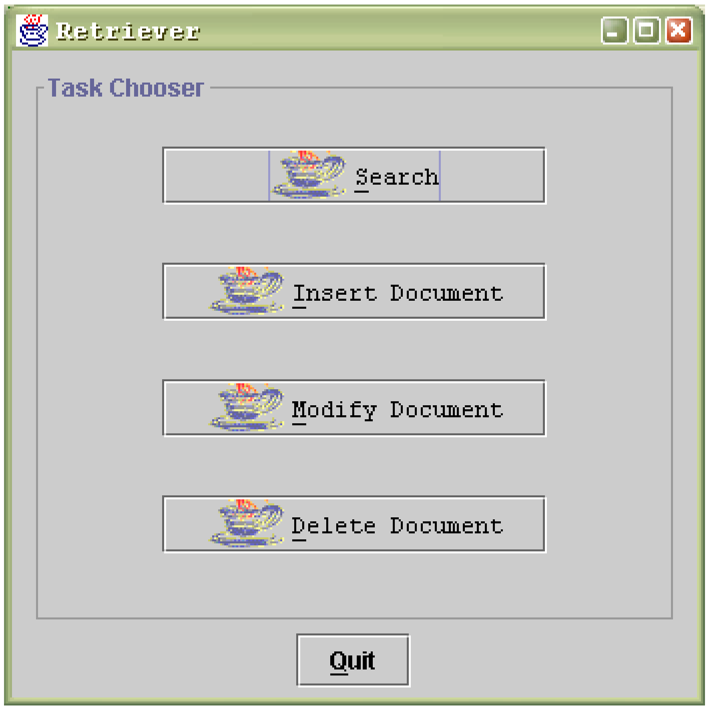


Figure 6. The **Task Chooser**, A Text Retrieval System Using Latent Semantic Analysis

The main feature of the system is allowing users to submit a query. Figure 7 shows the dialog that appears as the user presses the Search button prompting the user to type a query. After typing the query, the user may start the search by clicking on **Go** or cancel the search by choosing **Stop**. An empty query will cause the following dialog to appear telling the user to type a query as shown in Figure 8. If **OK** is pressed, the user returns to the interface for search.

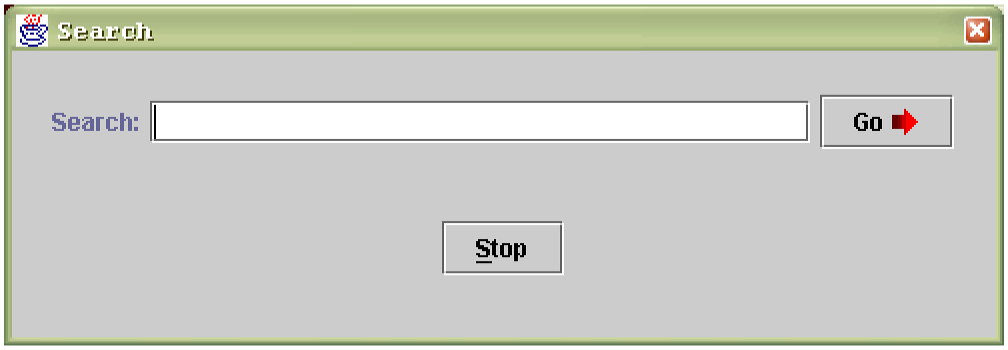
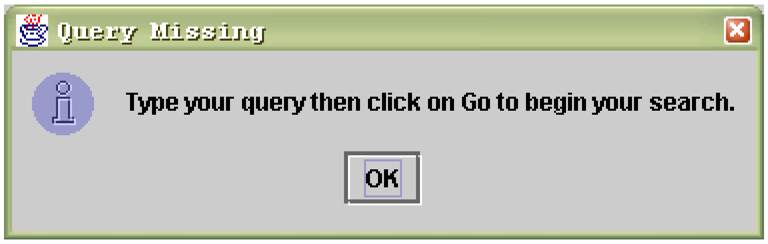


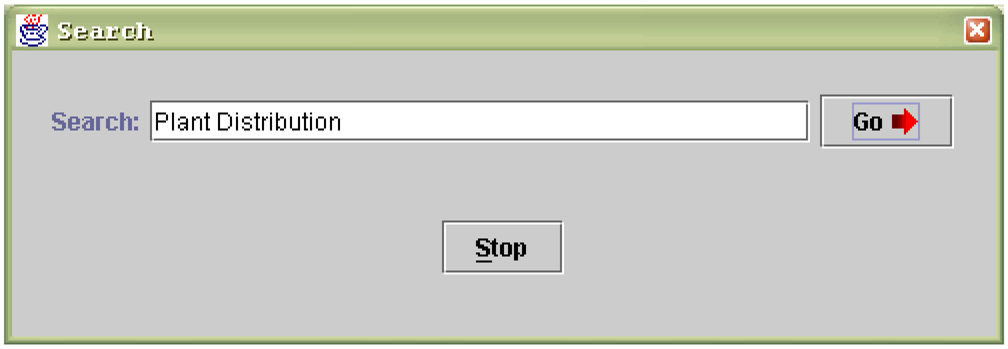
Figure 7. Interface for Search, A Text Retrieval System Using Latent Semantic Analysis


Query Missing

Type your query then click on Go to begin your search.

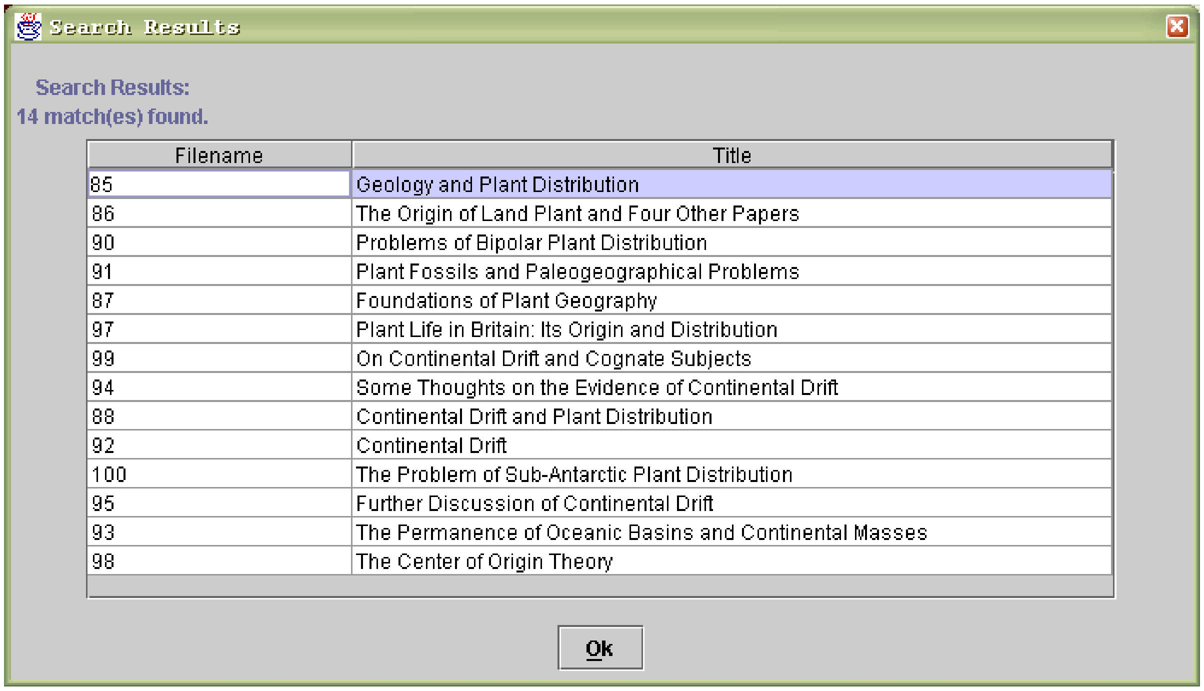
Figure 8. Information Dialog for Missing Query,
A Text Retrieval System Using Latent Semantic Analysis

One of the queries submitted, “Plant Distribution”, is shown in Figure 9. The system returned 14 matches including 5 document titles that did not contain any of the words “plant” or “distribution”. One such document title is “Some Thoughts on the Evidence of Continental Drift”. The concepts “plant distribution” and “continental drift” are related as they both appear in the document title “Continental Drift and Plant Distribution” (*See* Figure 10).



Plant Distribution

Figure 9. The query "Plant Distribution", A Text Retrieval System Using Latent Semantic Analysis

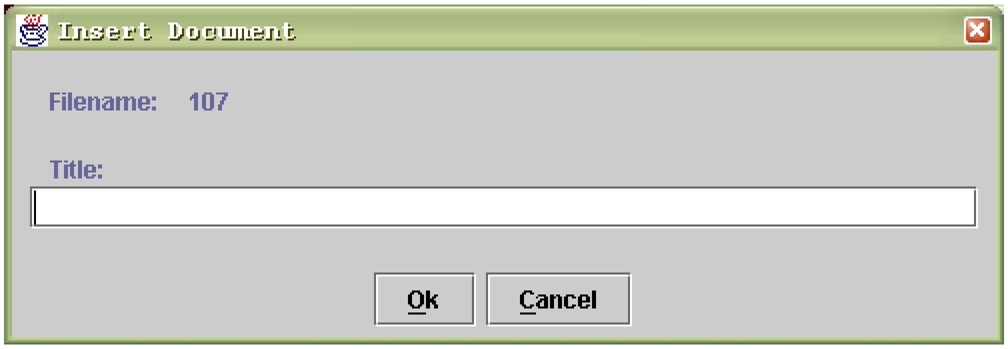


Search Results:
14 match(es) found.

Filename	Title
85	Geology and Plant Distribution
86	The Origin of Land Plant and Four Other Papers
90	Problems of Bipolar Plant Distribution
91	Plant Fossils and Paleogeographical Problems
87	Foundations of Plant Geography
97	Plant Life in Britain: Its Origin and Distribution
99	On Continental Drift and Cognate Subjects
94	Some Thoughts on the Evidence of Continental Drift
88	Continental Drift and Plant Distribution
92	Continental Drift
100	The Problem of Sub-Antarctic Plant Distribution
95	Further Discussion of Continental Drift
93	The Permanence of Oceanic Basins and Continental Masses
98	The Center of Origin Theory

Figure 10. Search results for the query "Plant Distribution",
A Text Retrieval System Using Latent Semantic Analysis

The second functionality of the system is allowing users to add document title. The filename has already been generated by the system. The user is prompted to enter a document title and is given the option to cancel the addition of new document title. This is shown in figure 11.

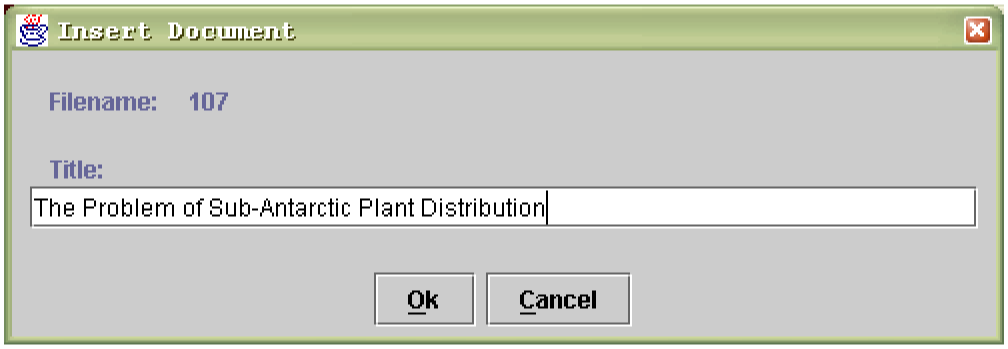

Insert Document

Filename: 107

Title:

Figure 11. Interface for adding a new document title, A Text Retrieval System Using Latent Semantic Analysis

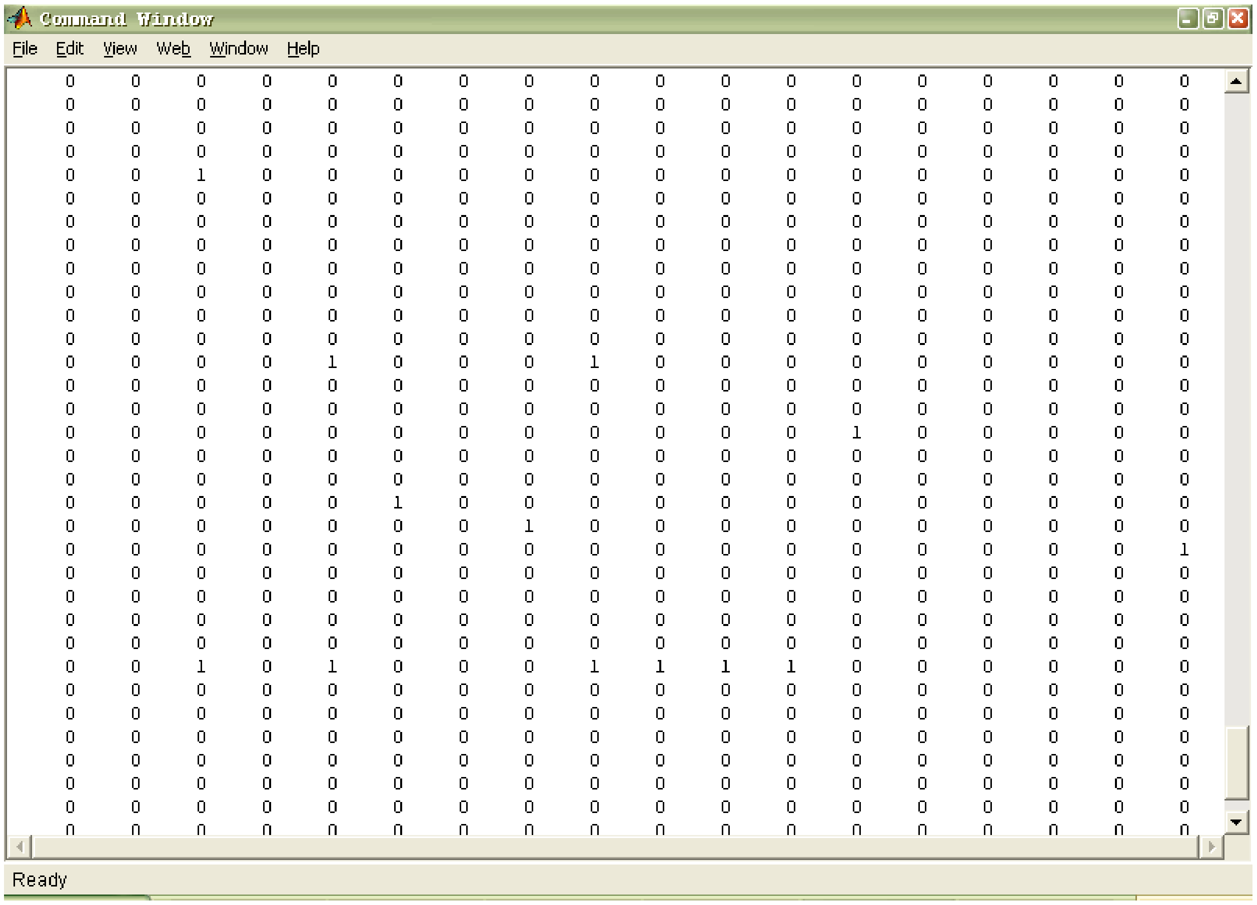
The addition of a new document title such as in Figure 12 causes a new document record having the generated filename and the document title supplied by the user as fields. The update matrix process which is completely hidden from the user takes place. Figure 13 shows a portion of the term-by-document matrix generated after the addition of new document and Figure 14 shows a portion of the matrix after the Latent Semantic Analysis using truncated Singular Value Decomposition. These can be seen by loading the matrices from Matlab’s command window.


Insert Document

Filename: 107

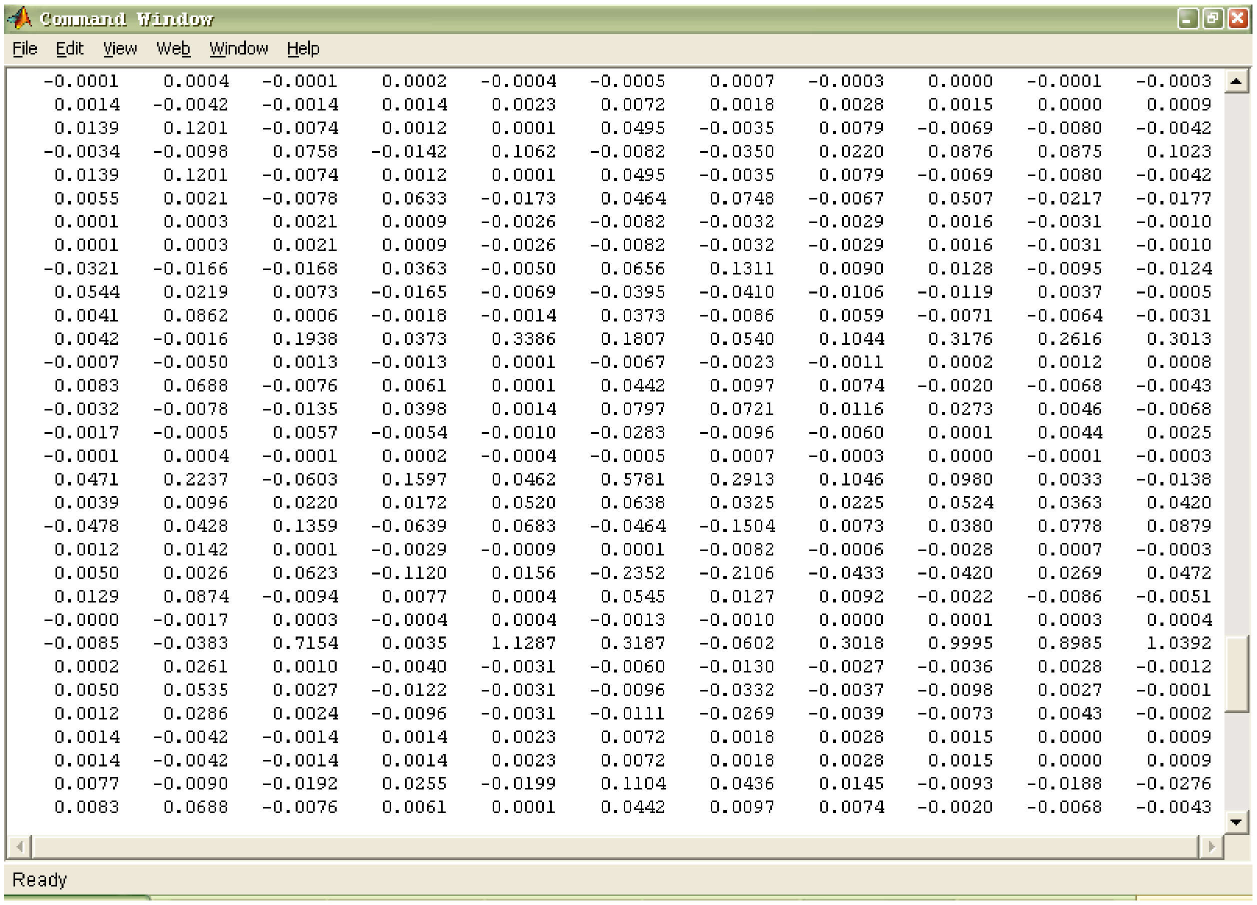
Title: The Problem of Sub-Antarctic Plant Distribution

Figure 12. A new document title is added, A Text Retrieval System Using Latent Semantic Analysis



```
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     1     0     0     0     1     1     1     1     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
```

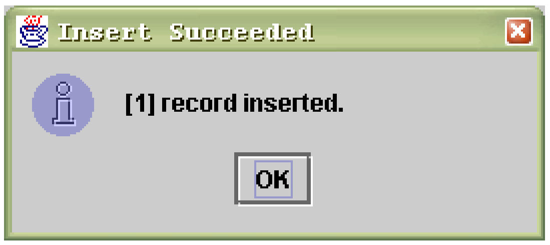
Figure 13. A portion of the term-by-document matrix from the Matlab command window, A Text Retrieval System Using Latent Semantic Analysis



```
  -0.0001    0.0004   -0.0001    0.0002   -0.0004   -0.0005    0.0007   -0.0003    0.0000   -0.0001   -0.0003
   0.0014   -0.0042   -0.0014    0.0014    0.0023    0.0072    0.0018    0.0028    0.0015    0.0000    0.0009
   0.0139    0.1201   -0.0074    0.0012    0.0001    0.0495   -0.0035    0.0079   -0.0069   -0.0080   -0.0042
  -0.0034   -0.0098    0.0758   -0.0142    0.1062   -0.0082   -0.0350    0.0220    0.0876    0.0875    0.1023
   0.0139    0.1201   -0.0074    0.0012    0.0001    0.0495   -0.0035    0.0079   -0.0069   -0.0080   -0.0042
   0.0055    0.0021   -0.0078    0.0633   -0.0173    0.0464    0.0748   -0.0067    0.0507   -0.0217   -0.0177
   0.0001    0.0003    0.0021    0.0009   -0.0026   -0.0082   -0.0032   -0.0029    0.0016   -0.0031   -0.0010
   0.0001    0.0003    0.0021    0.0009   -0.0026   -0.0082   -0.0032   -0.0029    0.0016   -0.0031   -0.0010
  -0.0321   -0.0166   -0.0168    0.0363   -0.0050    0.0656    0.1311    0.0090    0.0128   -0.0095   -0.0124
   0.0544    0.0219    0.0073   -0.0165   -0.0069   -0.0395   -0.0410   -0.0106   -0.0119    0.0037   -0.0005
   0.0041    0.0862    0.0006   -0.0018   -0.0014    0.0373   -0.0086    0.0059   -0.0071   -0.0064   -0.0031
   0.0042   -0.0016    0.1938    0.0373    0.3386    0.1807    0.0540    0.1044    0.3176    0.2616    0.3013
  -0.0007   -0.0050    0.0013   -0.0013    0.0001   -0.0067   -0.0023   -0.0011    0.0002    0.0012    0.0008
   0.0083    0.0688   -0.0076    0.0061    0.0001    0.0442    0.0097    0.0074   -0.0020   -0.0068   -0.0043
  -0.0032   -0.0078   -0.0135    0.0398    0.0014    0.0797    0.0721    0.0116    0.0273    0.0046   -0.0068
  -0.0017   -0.0005    0.0057   -0.0054   -0.0010   -0.0283   -0.0096   -0.0060    0.0001    0.0044    0.0025
  -0.0001    0.0004   -0.0001    0.0002   -0.0004   -0.0005    0.0007   -0.0003    0.0000   -0.0001   -0.0003
   0.0471    0.2237   -0.0603    0.1597    0.0462    0.5781    0.2913    0.1046    0.0980    0.0033   -0.0138
   0.0039    0.0096    0.0220    0.0172    0.0520    0.0638    0.0325    0.0225    0.0524    0.0363    0.0420
  -0.0478    0.0428    0.1359   -0.0639    0.0683   -0.0464   -0.1504    0.0073    0.0380    0.0778    0.0879
   0.0012    0.0142    0.0001   -0.0029   -0.0009    0.0001   -0.0082   -0.0006   -0.0028    0.0007   -0.0003
   0.0050    0.0026    0.0623   -0.1120    0.0156   -0.2352   -0.2106   -0.0433   -0.0420    0.0269    0.0472
   0.0129    0.0874   -0.0094    0.0077    0.0004    0.0545    0.0127    0.0092   -0.0022   -0.0086   -0.0051
  -0.0000   -0.0017    0.0003   -0.0004    0.0004   -0.0013   -0.0010    0.0000    0.0001    0.0003    0.0004
  -0.0085   -0.0383    0.7154    0.0035    1.1287    0.3187   -0.0602    0.3018    0.9995    0.8985    1.0392
   0.0002    0.0261    0.0010   -0.0040   -0.0031   -0.0060   -0.0130   -0.0027   -0.0036    0.0028   -0.0012
   0.0050    0.0535    0.0027   -0.0122   -0.0031   -0.0096   -0.0332   -0.0037   -0.0098    0.0027   -0.0001
   0.0012    0.0286    0.0024   -0.0096   -0.0031   -0.0111   -0.0269   -0.0039   -0.0073    0.0043   -0.0002
   0.0014   -0.0042   -0.0014    0.0014    0.0023    0.0072    0.0018    0.0028    0.0015    0.0000    0.0009
   0.0014   -0.0042   -0.0014    0.0014    0.0023    0.0072    0.0018    0.0028    0.0015    0.0000    0.0009
   0.0077   -0.0090   -0.0192    0.0255   -0.0199    0.1104    0.0436    0.0145   -0.0093   -0.0188   -0.0276
   0.0083    0.0688   -0.0076    0.0061    0.0001    0.0442    0.0097    0.0074   -0.0020   -0.0068   -0.0043
```

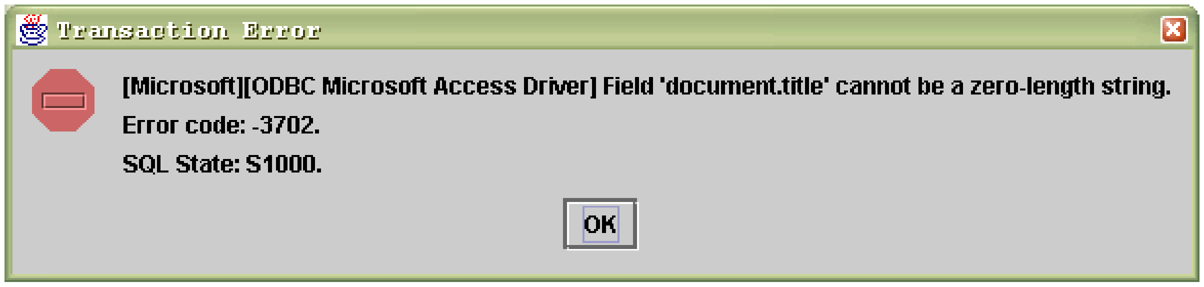
Figure 14. A portion of the matrix after LSA using truncated SVD,
A Text Retrieval System Using Latent Semantic Analysis

If no problems are encountered by the system, the user is informed of the success of addition of new document title as shown in figure 15. If a problem is encountered, the system informs the user of the failure. Figures 16 and 17 shows dialogs for these.



[1] record inserted.

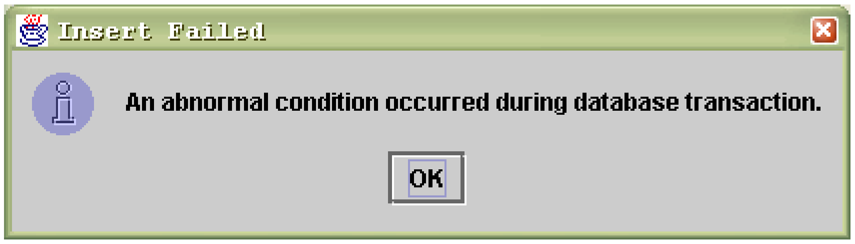
Figure 15. Insert Succeeded, A Text Retrieval System Using Latent Semantic Analysis



Transaction Error

[Microsoft][ODBC Microsoft Access Driver] Field 'document.title' cannot be a zero-length string.
Error code: -3702.
SQL State: S1000.

Figure 16. A transaction error, A Text Retrieval System Using Latent Semantic Analysis

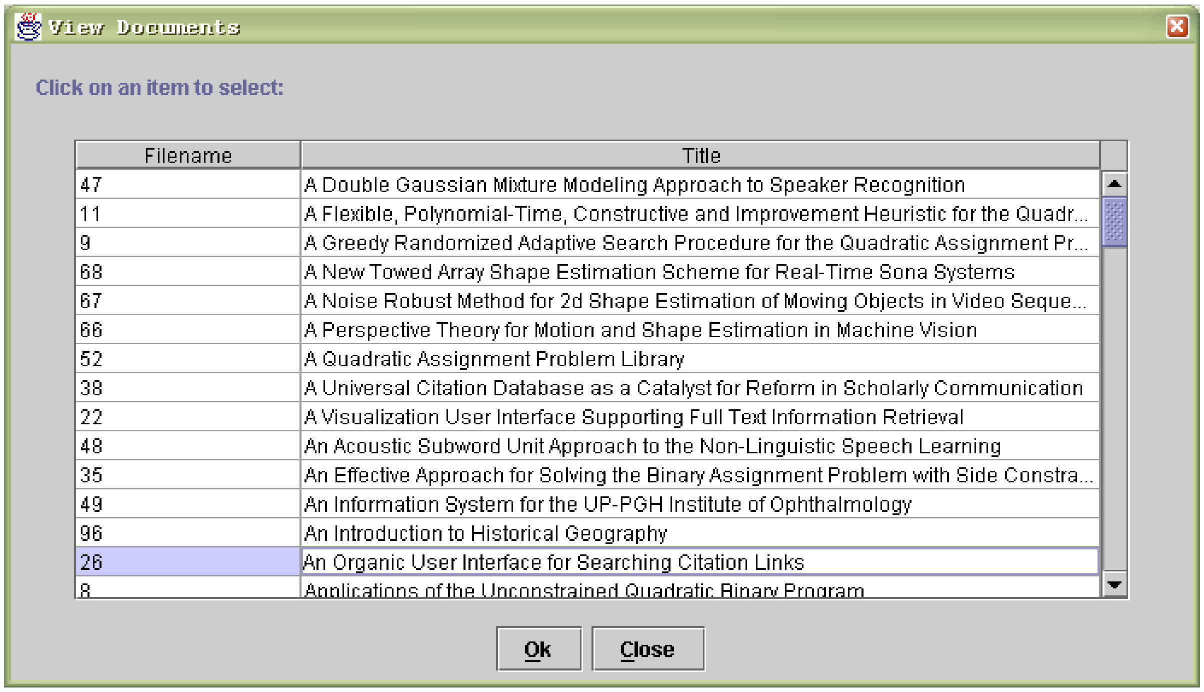


Insert Failed

An abnormal condition occurred during database transaction.

Figure 17. Database transaction error causes addition of new document title to fail, A Text Retrieval System Using Latent Semantic Analysis

Choosing to edit a document title displays all the documents from the document table such as shown in Figure 18. Once a document is selected such as in Figure 19 the document record having the document filename and the document title selected by the user is displayed. The document filename is not editable. Only the document title can be replaced by the user. This record is saved in the document table in place of the original.

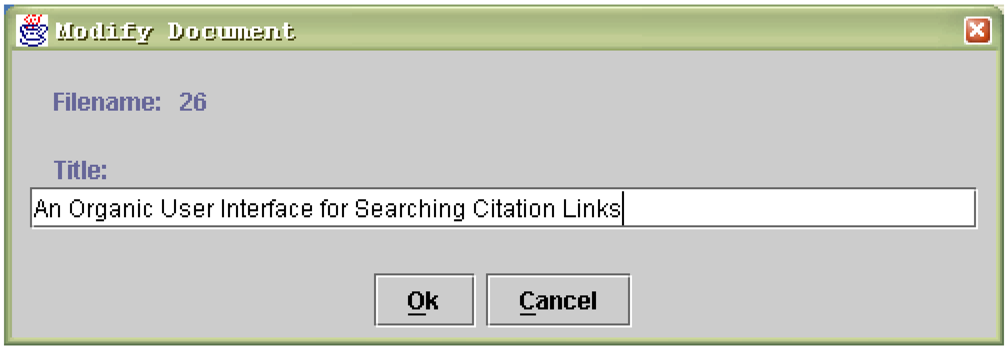


View Documents

Click on an item to select:

Filename	Title
47	A Double Gaussian Mixture Modeling Approach to Speaker Recognition
11	A Flexible, Polynomial-Time, Constructive and Improvement Heuristic for the Quadr...
9	A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Pr...
68	A New Towed Array Shape Estimation Scheme for Real-Time Sona Systems
67	A Noise Robust Method for 2d Shape Estimation of Moving Objects in Video Seque...
66	A Perspective Theory for Motion and Shape Estimation in Machine Vision
52	A Quadratic Assignment Problem Library
38	A Universal Citation Database as a Catalyst for Reform in Scholarly Communication
22	A Visualization User Interface Supporting Full Text Information Retrieval
48	An Acoustic Subword Unit Approach to the Non-Linguistic Speech Learning
35	An Effective Approach for Solving the Binary Assignment Problem with Side Constra...
49	An Information System for the UP-PGH Institute of Ophthalmology
96	An Introduction to Historical Geography
26	An Organic User Interface for Searching Citation Links
8	Applications of the Unconstrained Quadratic Binary Program

Figure 18. The documents from the document table, A Text Retrieval System Using Latent Semantic Analysis



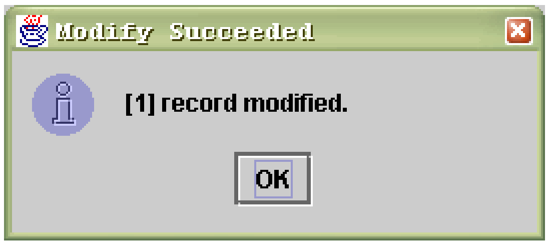
Modify Document

Filename: 26

Title: An Organic User Interface for Searching Citation Links

Figure 19. Interface for Editing a Document Title, A Text Retrieval System Using Latent Semantic Analysis

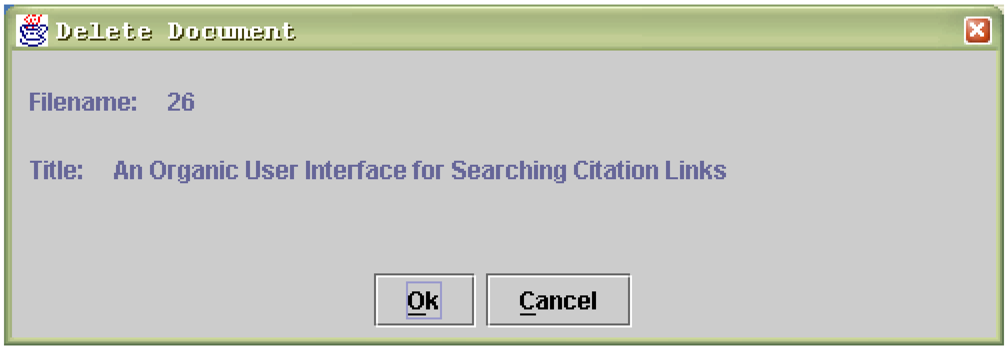
If no problems are encountered by the system, the user is informed of the success of editing the document title. The update matrix process which is completely hidden from the user takes place. The matrices generated follow the LSA mechanism using SVD resulting in matrices similar to the ones shown in Figures 13 and 14. The user is informed of the success of editing of document title as shown in Figure 20. If a problem is encountered, the system informs the user of the failure with dialogs similar to Figures 16 and 17.



[1] record modified.

Figure 20. Modify Succeeded, A Text Retrieval System Using Latent Semantic Analysis

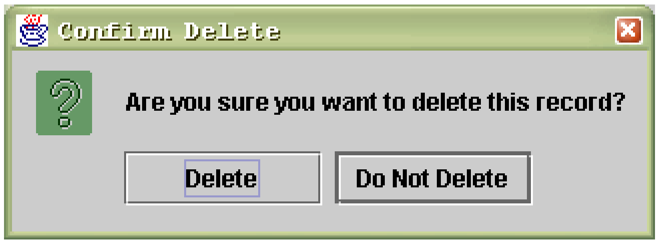
The last main functionality of the system is allowing them to delete a document record. For this process, all the documents in the document table are displayed as in Figure 18. Once a document is selected such as in Figure 21 the document record having the document filename and the document title selected by the user is displayed. Both the document filename and the document title are not editable. The user is given the option to cancel the deletion. When the **Ok** button is pressed, the user is prompted to confirm delete action as shown in Figure 22.



Filename: 26

Title: An Organic User Interface for Searching Citation Links

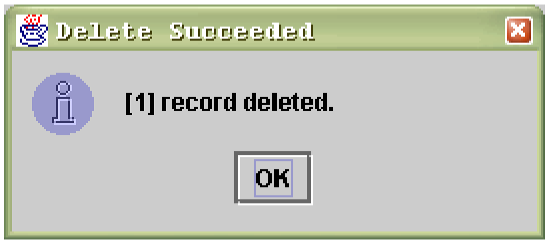
Figure 21. Interface for Deleting a Document Title, A Text Retrieval System Using Latent Semantic Analysis



Are you sure you want to delete this record?

Figure 22. Confirm Delete Action, A Text Retrieval System Using Latent Semantic Analysis

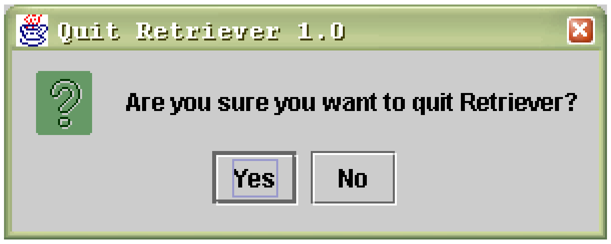
If **Delete** is clicked on and no problems are encountered by the system, the user is informed of the success of deleting the document title. The update matrix process which is completely hidden from the user takes place. The matrices generated follow the LSA mechanism using SVD resulting in matrices similar to the ones shown in Figures 13 and 14. The user is informed of the success of deleting of document title as shown in Figure 23. If a problem is encountered, the system informs the user of the failure with dialogs similar to Figures 16 and 17.


Delete Succeeded

[1] record deleted.

Figure 23. Delete Succeeded, A Text Retrieval System Using Latent Semantic Analysis

If the user opts to leave the program, the dialog shown in figure 24 appears. Choosing **No** returns the user to the main menu while choosing **Yes** closes the program.


Quit Retriever 1.0

Are you sure you want to quit Retriever?

Figure 24. Quit Retriever 1.0, A Text Retrieval System Using Latent Semantic Analysis

For this study, the index is obtained by recombining the decomposition $U\Sigma V^T$ taking only k orthogonal factors of each of the matrices U, Σ, and V. Following the observation that applications using the singular value decomposition for latent semantic analysis usually take about 10% of the total number of documents in the document collection such as in the works of Dumais et al. [3, 6], k is set to k = round (number of documents ×

10%) + 1. A value of one has been added so that the *k* will never be equal to zero such as in the case that the number of documents is less than ten. In such case, the resulting matrices will be empty and the system fails. It has been observed that raising this threshold results in too many documents that are irrelevant to the query while decreasing it results in very few relevant documents in the search results. For this reason, the threshold that determines which documents qualify as search results is set to 1.3.

Nine queries have been passed to evaluate the performance of the system. For each query, the documents have been judged as either relevant or irrelevant. Recall is given by number of relevant documents returned divided by the total number of relevant documents in the collection. Precision is given by the number of relevant documents divided by the total number of returned documents. The following table gives details of some queries passed into the system:

Query	**Total # of documents relevant**	**Total # of documents retrieved**	**Total # of relevant documents retrieved**	**Recall**	**Precision**
quadratic assignment problem	11	10	9	0.81	0.90
information visualization	17	13	11	0.64	0.84
continental drift	6	4	4	0.66	1.00
Plant distribution	14	14	13	0.92	0.92
fossil floras	10	4	4	0.40	1.00
paleogeographic American	5	2	2	0.40	1.00
shape size	6	1	1	0.16	1.00
text-independent speaker	4	1	1	0.25	1.00
shape estimation	4	2	2	0.50	1.00
Average				**0.52**	**0.96**

Note: Figures have been rounded down to 2 decimal places.

Table 5. Queries passed into the system, A Text Retrieval System Using Latent Semantic Analysis

The average recall for the data set is 52% and the average precision is 96%. In past studies, it has been anticipated that recall would fall somewhere around 20% and precision about 50% using keyword matching. Basing on the above data, it can be inferred that latent semantic indexing might have bettered performance.

VII. DISCUSSION

The system A Text Retrieval System Using Latent Semantic Analysis allows adding a new document title, editing an existing document title and deletion of an existing document title. These functionalities provided basis for the system to generate an index following the SVD, thus, creating a new representation of the document collection. Adding a new document title, for instance, created a new vector in the term space and possibly created new dimensions with the introduction of new content words. Editing an existing document title changes the position of its representing vector in the term space based on possibly a new set of dimensions resulting from introduction of new content words or removal of content words that are not contained in any document after editing. Similarly, deletion of an existing document title removes a vector from the term space and possibly removes a number of dimensions following a possible removal of content words that are not contained in any other document after deletion.

The system's main functionality, allowing users to submit a query, makes use of the index generated after the above mentioned functionalities. Basing on sampled queries shown in the previous section, recall was 52% and precision was 96% compared with an anticipated recall of 20% and anticipated precision of 50% using keyword matching. The application was able to return relevant document titles that do not have some or all of the query terms. This results from indexing using the approximate term-by-document matrix generated by truncated SVD rather than the original term-by-document matrix from which the former was derived. These two matrices are directly related to the indexing using LSA and indexing using keyword matching, respectively.

Existing systems fail where documents that suppose to match a query do not contain any of the query words accounting to the failure of mere term overlapping schemes to address the problem of synonymy [7]. It can be drawn from the previous section that the system was able to return more than twice of what keyword matching could have delivered. Also, the system delivered search results that are about twice more relevant than what most current raw keyword matching retrieval systems could have given. This is indicated by the 96% precision for the set of queries passed in to the system.

Deerwester et al. reported that precision was highest at higher levels of recall which shows that indexing using LSA deals better with the problem of synonymy rather than polysemy [7]. Our study produced a different result where precision was almost perfect at the lower levels of recall. This can be accounted for the smallness of the document collection which may not have adequately sampled a typical document collection. Relevancy judgments may have also played a vital role in the outcome of the above numbers.

It can be anticipated that sets containing tens of thousands of documents will be extremely difficult to manage since the extraction of the singular value decomposition of a very large matrix can be much computationally expensive. This is the reason why said technique is hardly used in the internet where search engines index millions of documents and the collections need be updated from time to time. Even so, improvements on recall and precision can not be overlooked. It can be inferred that LSA is indeed a promising tool for uncovering the latent semantic structure to improve information retrieval.

VIII. CONCLUSION

A document retrieval system is created that is capable of both high precision and high recall through a model that estimates term-to-document similarity with the use of the LSA.

The retrieval system is able to:

d. enable users to add, modify and delete document titles in database;

e. generate an index following the SVD with term-by-document matrix as initial input;

f. allow queries on document titles and return search results based on LSA.

IX. RECOMMENDATIONS

While the singular value decomposition model produced satisfactory results for the latent semantic analysis, there are other models that may be used to achieve faster results accounting to computational inexpensiveness but are less accurate [13]. Indexing using LSA may be readily used with other methods, such as stemming and weighting, for increased performance.

The application may also be improved by making use of a more powerful processor to increase speed of manipulating very large matrices. Furthermore, instead of the described means of selection of document titles for output as search results, other methods may be explored as described by Deerwester et al. [7] like finding the documents that have the highest cosines of the angle between its representing vector and the query vector.

The choice of the number of dimensions, *k*, to which the matrices are truncated, is still an open issue. The threshold that serves as cut-off between distances selected as being near or not to a particular query may also be increased or decreased. Such adjustments, however, might result changes in recall and precision.

The data set may be replaced as any user might wish to use the application as a specialized retrieval system.

X. BIBLIOGRAPHY

[1] Michael L. Littman, Susan T. Dumais, Thomas K. Landauer, "Automatic Cross Language Information Retrieval Using Latent Semantic Indexing", *Proceedings of the ACM SIGIR '96 Workshop on Cross-Linguistic Information Retrieval*, August 1996.

[2] "Latent Semantic Indexing", National Institute for Technology in Liberal Education, http://www.nitle.org, 2002.

[3] Steve Lawrence, C. Lee Giles, "Tips for Searching the Web", *Science*, Volume 280, Number 5360, pp. 98-100. 1998.

[4] Thomas K. Landauer, Susan T. Dumais, "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge".

[5] John C. Dunn, Osvaldo P. Almeida, Lee Barclay, Anna Waterreus and Leon Flicker, "Latent Semantic Analysis: A New Method to Measure Prose Recall", *Journal of Clinical and Experimental Neuropsychology*, Volume 24, Number 1, pp. 26-35. 2002.

[6] Peter W. Foltz, "Using Latent Semantic Indexing for Information Filtering" In R.B. Allen (Ed.), *Proceedings of the Conference on Office Information Systems*, Cambridge, MA, pp. 40-47. 1990.

[7] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Richard Harshman, "Indexing by Latent Semantic Analysis", *Journal of the Society for Information Science*, 41(6), pp. 391-407. 1990.

[8] "Telcordia™ Latent Semantic Indexing Software (LSI). Beyond Keyword Retrieval", Telcordia Technologies, http://www.telcordia.com, 1997-2001.

[9] Susan T. Dumais, "Using Latent Semantic Indexing (LSI) for Information Retrieval, Information Filtering and Other Things", *Cognitive Technology Conference*, April 1997.

[10] David C. Lay, *Linear Algebra and Its Applications*, Addison Wesley Publishing Company, Inc., 1994.

[11] Bernard Kolman, David R. Hill, *Elementary Linear Algebra*, Prentice-Hall, Inc., 2000, 1996.

[12] Mohamed Amine Khamsi, "Eigenvalues and Eigenvectors Technique", MathMedics, LLC, http://www.sosmath.com, 1999-2003.

[13] Zoran Pecenovic, "Image Retrieval Using Latent Semantic Analysis", Final Year Graduate Thesis, May 1997.

XI. APPENDICES

A. **`retriever1.java`** Source Code

```
import java.io.*;
import java.awt.*;
import java.sql.*;
import java.lang.*;
import java.math.*;
import java.util.*;
import java.awt.event.*;
import java.awt.Graphics;
import java.lang.String.*;
import javax.swing.*;
import javax.swing.event.*;
public class retriever1 extends JFrame implements ActionListener
      {
      boolean ok;
      double length1;
      int rpint1, temp2, temp4, gint1;
      CallableStatement getNewID;
      Connection conn;
      Font rfont1;
      ImageIcon ricon1, sicon1;
      JButton rbutton1, rbutton2, rbutton3, rbutton4, rbutton5;
      JLabel label01, label02, label03, label04, label05, label06,
label07, label08;
      JMatLink engine1;
      JPanel panel4, panel5, panel6;
      PreparedStatement insertdocument;
      String record, temp1, temp3, temp5;
      class record1
            {
            int FILENAME;
            String TITLE;
            }
      class representation extends Vector
            {
            int tint1, tint2, tint3, tint4, tint5;
            String delimetersset, token, tstring1;
            StringTokenizer machine;
            Vector tvector1;
            int[] t1int1;
            public representation(String ptstring1)
                  {
                  tint1 = 0;
                  tint5 = 0;
                  tvector1 = new Vector();
                  delimetersset = ".,;:?!#&* \t\n\r";
                  machine = new StringTokenizer(ptstring1,
delimetersset);
```

```
                  while(machine.hasMoreTokens() == true)
                        {
                        token = machine.nextToken();
                        tvector1.add(tint1, token);
                        tint1++;
                        }
                  for(tint2 = 0; tint2 < tint1; tint2++)
                        {
                        tstring1 = (String)
      (tvector1.elementAt(tint2));
                        tint3 = tint2;
                        while((tint3 > 0) &&
(tvector1.elementAt(tint3 - 1).toString().compareTo(tstring1) > 0))
                              {
                              tvector1.set(tint3,
tvector1.elementAt(tint3-1));
                              tint3 = tint3 - 1;
                              }
                        tvector1.set(tint3, tstring1);
                        }
                  for(tint2 = 0; tint2 < tint1; tint2++)
                        {
            if(find("content", tvector1.elementAt(tint2).toString())
== true)
                        {
                        this.add(tint5, new
Integer(getindex(tvector1.elementAt(tint2).toString())));
                        tint5++;
                        }
                  }
            }
      }
class display_interface extends JDialog implements ActionListener
      {
      int index1, i0, selectedRow;
      record1 r0, r1;
      JButton b1, b2;
      JLabel vlabel1;
      JList list1;
      JPanel rpanel1, rpanel2, rpanel3;
      JScrollPane pane1;
      JTable vtable1;
      ListSelectionModel lsm, rowsm;
      Object selectedFilename, selectedTitle;
      PreparedStatement st1;
      ResultSet rs;
      String vstring1;
      Vector vector1, vector2, vector3;
      public display_interface(retriever1 mother)
            {
            super(mother, "View Documents", true);
            setSize(700, 400);
            getContentPane().setLayout(new BorderLayout());
            index1 = 0;
            vlabel1 = new JLabel("     Click on an item to select:");
            b1 = new JButton("Ok");
            b2 = new JButton("Close");
            rpanel1 = new JPanel();
            rpanel2 = new JPanel();
            rpanel3 = new JPanel();
            vector1 = new Vector();
            vector2 = new Vector();
            b1.addActionListener(this);
            b1.setMnemonic(KeyEvent.VK_O);
            b2.addActionListener(this);
            b2.setMnemonic(KeyEvent.VK_C);
            vector1.add(0, "Filename");
            vector1.add(1, "Title");
            try
                  {
                  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                  }
            catch(ClassNotFoundException e)
                  {
                  ok = false;
                  JOptionPane.showMessageDialog(this,
e.getMessage(), "Driver Class Not Found Error",
JOptionPane.ERROR_MESSAGE);
                  }
            try
                  {
                  conn = DriverManager.getConnection("jdbc:odbc:MS
Access Database");
                  st1 = conn.prepareStatement("SELECT * FROM
document ORDER BY title");
                  rs = st1.executeQuery();
                  i0 = 0;
                  while(rs.next() == true)
                        {
                        vector3 = new Vector();
                        r0 = new record1();
                        r0.FILENAME = rs.getInt("filename");
                        r0.TITLE = rs.getString("title");
                  vector3.addElement(String.valueOf(r0.FILENAME));
                        vector3.addElement(r0.TITLE);
                        vector2.add(i0, vector3);
                        i0++;
                        }
                  conn.close();
                  }
            catch(SQLException f)
                  {
                  JOptionPane.showMessageDialog(this, f.getMessage()
+ "\nError Code: " + f.getErrorCode() + ".\nSQL State: " +
f.getSQLState() + ".", "Vector Representation Error",
JOptionPane.ERROR_MESSAGE);
                  ok = false;
                  }
            vtable1 = new JTable(vector2, vector1);
            vtable1.setPreferredScrollableViewportSize(new
Dimension(600, 250));
      vtable1.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
      vtable1.getColumnModel().getColumn(0).setPreferredWidth(50);
            pane1 = new JScrollPane(vtable1);
            rowsm = vtable1.getSelectionModel();
            pane1.createHorizontalScrollBar().setEnabled(true);
            rpanel1.setLayout(new BoxLayout(rpanel1,
BoxLayout.Y_AXIS));
            rpanel1.add(new JLabel(" "));
            rpanel1.add(vlabel1);
            rpanel1.add(new JLabel(" "));
            rpanel2.add(pane1);
            rpanel3.add(b1);
            rpanel3.add(b2);
            getContentPane().add(rpanel1, BorderLayout.NORTH);
            getContentPane().add(rpanel2, BorderLayout.CENTER);
            getContentPane().add(rpanel3, BorderLayout.SOUTH);
            getContentPane().add(new JLabel("   "),
      BorderLayout.EAST);
            getContentPane().add(new JLabel("   "),
      BorderLayout.WEST);
            rowsm.addListSelectionListener(new
      ListSelectionListener()
                  {
                  public void valueChanged(ListSelectionEvent e)
                        {
                        if(e.getValueIsAdjusting()) return;
                        lsm = (ListSelectionModel)e.getSource();
                        selectedRow = lsm.getMinSelectionIndex();
                        selectedFilename =
vtable1.getValueAt(selectedRow, 0);
                        selectedTitle =
vtable1.getValueAt(selectedRow, 1);
                        vstring1 = selectedFilename.toString();
                        temp2 = Integer.parseInt(vstring1);
                        temp1 = selectedTitle.toString();
                        }
                  });
            }
      public void actionPerformed(ActionEvent e)
            {
            if(e.getActionCommand() == temp1)
                  {
                  modify_interface that1 = new
      modify_interface(this);
                  that1.setVisible(true);
                  }
            if(e.getSource() == b1)
                  {
                  if(temp2 != 0)
                        {
                        modify_interface that2 = new
modify_interface(this);
                        that2.setVisible(true);
                        }
                  else JOptionPane.showMessageDialog(this, "Select a
record to modify then click OK.", "Selection Missing",
JOptionPane.INFORMATION_MESSAGE);
```

```
                        }
                if(e.getSource() == b2) this.dispose();
                }
        }
class display_interface0 extends JDialog implements ActionListener
        {
        int index1, i0, selectedRow;
        record1 r0, r1;
        JButton b1, b2;
        JLabel vlabel1;
        JList list1;
        JPanel rpanel1, rpanel2, rpanel3;
        JScrollPane pane1;
        JTable vtable1;
        ListSelectionModel lsm, rowsm;
        Object selectedFilename, selectedTitle;
        PreparedStatement st1;
        ResultSet rs;
        String vstring1;
        Vector vector1, vector2, vector3;
        public display_interface0(retriever1 mother)
                {
                super(mother, "View Documents", true);
                setSize(700, 400);
                getContentPane().setLayout(new BorderLayout());
                index1 = 0;
                vlabel1 = new JLabel("     Click on an item to select:");
                b1 = new JButton("Ok");
                b2 = new JButton("Close");
                rpanel1 = new JPanel();
                rpanel2 = new JPanel();
                rpanel3 = new JPanel();
                vector1 = new Vector();
                vector2 = new Vector();
                b1.addActionListener(this);
                b1.setMnemonic(KeyEvent.VK_O);
                b2.addActionListener(this);
                b2.setMnemonic(KeyEvent.VK_C);
                vector1.add(0, "Filename");
                vector1.add(1, "Title");
                try
                        {
                        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                        }
                catch(ClassNotFoundException e)
                        {
                        ok = false;
                        JOptionPane.showMessageDialog(this,
e.getMessage(), "Driver Class Not Found Error",
JOptionPane.ERROR_MESSAGE);
                        }
                try
                        {
                        conn = DriverManager.getConnection("jdbc:odbc:MS
Access Database");
```

```
                  st1 = conn.prepareStatement("SELECT * FROM
document ORDER BY title");
                  rs = st1.executeQuery();
                  i0 = 0;
                  while (rs.next() == true)
                        {
                        vector3 = new Vector();
                        r0 = new record1();
                        r0.FILENAME = rs.getInt("filename");
                        r0.TITLE = rs.getString("title");
                  vector3.addElement(String.valueOf(r0.FILENAME));
                        vector3.addElement(r0.TITLE);
                        vector2.add(i0, vector3);
                        i0++;
                        }
                  conn.close();
                  }
            catch(SQLException f)
                  {
                  JOptionPane.showMessageDialog(this, f.getMessage()
+ "\nError Code: " + f.getErrorCode() + ".\nSQL State: " +
f.getSQLState() + ".", "Vector Representation Error",
JOptionPane.ERROR_MESSAGE);
                  ok = false;
                  }
            vtable1 = new JTable(vector2, vector1);
            vtable1.setPreferredScrollableViewportSize(new
Dimension(600, 250));
      vtable1.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
      vtable1.getColumnModel().getColumn(0).setPreferredWidth(10);
      vtable1.getColumnModel().getColumn(1).setPreferredWidth(30);
            pane1 = new JScrollPane(vtable1);
            rpanel1.setLayout(new BoxLayout(rpanel1,
BoxLayout.Y_AXIS));
            rpanel1.add(new JLabel(" "));
            rpanel1.add(vlabel1);
            rpanel1.add(new JLabel(" "));
            rpanel2.add(pane1);
            rpanel3.add(b1);
            rpanel3.add(b2);
            getContentPane().add(rpanel1, BorderLayout.NORTH);
            getContentPane().add(rpanel2, BorderLayout.CENTER);
            getContentPane().add(rpanel3, BorderLayout.SOUTH);
            getContentPane().add(new JLabel("   "),
      BorderLayout.EAST);
            getContentPane().add(new JLabel("   "),
      BorderLayout.WEST);
            rowsm = vtable1.getSelectionModel();
            rowsm.addListSelectionListener(new
      ListSelectionListener()
                  {
                  public void valueChanged(ListSelectionEvent e)
                        {
                        if(e.getValueIsAdjusting()) return;
                        lsm = (ListSelectionModel)e.getSource();
                        selectedRow = lsm.getMinSelectionIndex();
                        selectedFilename =
vtable1.getValueAt(selectedRow, 0);
                        selectedTitle =
vtable1.getValueAt(selectedRow, 1);
                        vstring1 = selectedFilename.toString();
                        temp4 = Integer.parseInt(vstring1);
                        temp5 = selectedTitle.toString();
                        }
                  });
            }
      public void actionPerformed(ActionEvent e)
            {
            if(e.getActionCommand() == temp1)
                  {
                  delete_interface that1 = new
      delete_interface(this);
                  that1.setVisible(true);
                  }
            if(e.getSource() == b1)
                  {
                  if(temp4 != 0)
                        {
                        delete_interface that2 = new
delete_interface(this);
                        that2.setVisible(true);
                        }
                  else JOptionPane.showMessageDialog(this, "Select a
record to delete then click OK to start deletion.", "Selection
Missing", JOptionPane.INFORMATION_MESSAGE);
                  }
            if(e.getSource() == b2) this.dispose();
            }
      }
class insert_interface extends JDialog implements ActionListener
      {
      int dindex;
      JButton ibutton1, ibutton2;
      JLabel ilabel1, ilabel2;
      JPanel ipanel1, ipanel2;
      JTextField itextfield1;
      String rpstring1, rpstring2;
      Vector ivector1;
      public insert_interface(retriever1 mother)
            {
            super(mother, "Insert Document", true);
            setSize(500, 170);
            getContentPane().setLayout(new BorderLayout());
            rpint1 = getnewfilename();
            rpstring2 = "";
            ipanel1 = new JPanel();
            ipanel2 = new JPanel();
            ilabel1 = new JLabel("Filename:     " +
String.valueOf(rpint1));
            ilabel2 = new JLabel("Title:  ");
            itextfield1 = new JTextField(20);
            ibutton1 = new JButton("Ok");
            ibutton2 = new JButton("Cancel");
            ibutton1.addActionListener(this);
            ibutton1.setMnemonic(KeyEvent.VK_O);
            ibutton2.addActionListener(this);
            ibutton2.setMnemonic(KeyEvent.VK_C);
            ipanel1.setLayout(new BoxLayout(ipanel1,
BoxLayout.Y_AXIS));
            ipanel1.add(new JLabel(" "));
            ipanel1.add(ilabel1);
            ipanel1.add(new JLabel(" "));
            ipanel1.add(ilabel2);
            ipanel1.add(itextfield1);
            ipanel1.add(new JLabel(" "));
            ipanel2.add(ibutton1);
            ipanel2.add(ibutton2);
            getContentPane().add(ipanel1, BorderLayout.CENTER);
            getContentPane().add(ipanel2, BorderLayout.SOUTH);
            getContentPane().add(new JLabel("   "),
      BorderLayout.EAST);
            getContentPane().add(new JLabel("   "),
      BorderLayout.WEST);
            }
      public void actionPerformed(ActionEvent e)
            {
            if(e.getSource() == ibutton1)
                  {
                  ok = true;
                  rpstring1 = itextfield1.getText().trim();
            dindex = database_system("INSERT INTO document VALUES(?,
?)", rpint1, rpstring1, 1, 2);
                  ivector1 = new text_processor(rpstring1);
                  semantic_analyzer(1, dindex, ivector1);
                  if(ok = true) JOptionPane.showMessageDialog(this,
"[1] record inserted.", "Insert Succeeded",
JOptionPane.INFORMATION_MESSAGE, null);
                  this.dispose();
                  }
            if(e.getSource() == ibutton2) this.dispose();
            }
      }
class modify_interface extends JDialog implements ActionListener
      {
      double yy;
      int dindex, k, newtemp1, size1, newint1, newint2, tint2, tint3;
      JButton mbutton1, mbutton2;
      JLabel mlabel1, mlabel2;
      JPanel mpanel1, mpanel2;
      JTextField mtextfield1;
      String rpstring1, rpstring2, newstring1, delimetersset,
tempstring1, tstring1;
      StringTokenizer newmachine1, newmachine2;
      Vector mvector1, newvector1, newvector2, newvector3;
      public modify_interface(display_interface mother)
            {
            super(mother, "Modify Document", true);
            setSize(500, 170);
            getContentPane().setLayout(new BorderLayout());
            rpint1 = temp2;
            rpstring2 = "";
            mpanel1 = new JPanel();
            mpanel2 = new JPanel();
            mlabel1 = new JLabel("Filename:   " +
String.valueOf(rpint1));
            mlabel2 = new JLabel("Title:  ");
            mtextfield1 = new JTextField(String.valueOf(temp1), 20);
            newstring1 = String.valueOf(temp1);
            mbutton1 = new JButton("Ok");
            mbutton2 = new JButton("Cancel");
            mbutton1.addActionListener(this);
            mbutton1.setMnemonic(KeyEvent.VK_O);
            mbutton2.addActionListener(this);
            mbutton2.setMnemonic(KeyEvent.VK_C);
            mpanel1.setLayout(new BoxLayout(mpanel1,
BoxLayout.Y_AXIS));
            mpanel1.add(new JLabel(" "));
            mpanel1.add(mlabel1);
            mpanel1.add(new JLabel(" "));
            mpanel1.add(mlabel2);
            mpanel1.add(mtextfield1);
            mpanel1.add(new JLabel(" "));
            mpanel2.add(mbutton2);
            getContentPane().add(mpanel1, BorderLayout.CENTER);
            getContentPane().add(mpanel2, BorderLayout.SOUTH);
            getContentPane().add(new JLabel("   "),
      BorderLayout.EAST);
            getContentPane().add(new JLabel("   "),
      BorderLayout.WEST);
            }
      public void actionPerformed(ActionEvent e)
            {
            if(e.getSource() == mbutton1)
                  {
                  ok = true;
                  rpstring1 = mtextfield1.getText().trim();
                  dindex = database_system("UPDATE document SET
title = ? WHERE filename = ?", rpint1, rpstring1, 2, 1);
                  mvector1 = new text_processor(rpstring1);
                  newvector1 = new Vector();
                  newvector2 = new Vector();
                  newvector3 = new Vector();
                  newint1 = 0;
                  newint2 = 0;
                  delimetersset = ".,;:?!#&* \t\n\r";
                  newmachine1 = new StringTokenizer(newstring1,
delimetersset);
                  while(newmachine1.hasMoreTokens() == true)
                        {
                        tempstring1 = newmachine1.nextToken();
                        newvector1.add(newint1, tempstring1);
                        newint1++;
                        }
                  for(tint2 = 0; tint2 < newint1; tint2++)
```

```
                        {
                        tstring1 = (String)
(newvector1.elementAt(tint2));
                        tint3 = tint2;
                        while((tint3 > 0) &&
(newvector1.elementAt(tint3 - 1).toString().compareTo(tstring1) > 0))
                              {
                              newvector1.set(tint3,
newvector1.elementAt(tint3-1));
                              tint3 = tint3 - 1;
                              }
                        newvector1.set(tint3, tstring1);
                        }
                  newmachine2 = new StringTokenizer(rpstring1,
delimetersset);
                  while(newmachine2.hasMoreTokens() == true)
                        {
                        tempstring1 = newmachine2.nextToken();
                        newvector2.add(newint2, tempstring1);
                        newint2++;
                        }
                  size1 = newvector1.size();
                  k = 0;
                  for (newtemp1 = 0; newtemp1 < size1; newtemp1++)
                        {
                        if(newvector2.contains(newvector1.elementAt(
newtemp1)) == false)
                              {
                              if(find("content", (String)
(newvector1.elementAt(newtemp1))) == true)
                                    {
            newvector3.addElement(newvector1.elementAt(newtemp1));
                                    int newint3 = getindex((String)
(newvector1.elementAt(newtemp1)));

engine1.engEvalString("determine(" + String.valueOf(newint3-k) +
")");
                                    engine1.engEvalString("load
newvar1");
                                    yy =
engine1.engGetScalar("newvar1");
                                    if(yy == (double)(0))
                                          {

engine1.engEvalString("deleteword(" + String.valueOf(newint3-k) +
")");
                                          k++;
                                          }
                                    }
                              }
                        }
                  newvoid1(newvector3);
                  mvector1 = new text_processor(rpstring1);
                  semantic_analyzer(2, dindex, mvector1);
                  if(ok = true) JOptionPane.showMessageDialog(this,
"[1] record modified.", "Modify Succeeded",
JOptionPane.INFORMATION_MESSAGE, null);
                  this.dispose();
                  }
            if(e.getSource() == mbutton2) this.dispose();
            }
      }
class delete_interface extends JDialog implements ActionListener
      {
      double yy;
      int dindex, k, n, newint1, newint2, newint3, newtemp1, size1,
tint2, tint3;
      JButton dbutton1, dbutton2;
      JLabel dlabel1, dlabel2;
      JPanel dpanel1, dpanel2;
      String rpstring2, delimetersset, newstring1, tempstring1,
tstring1;
      StringTokenizer newmachine1, newmachine2;
      Vector dvector1, newvector1, newvector2, newvector3;
      int[] d1int1 = {};
      public delete_interface(display_interface0 mother)
            {
            super(mother, "Delete Document", true);
            setSize(500, 170);
            getContentPane().setLayout(new BorderLayout());
            rpstring2 = "";
            dpanel1 = new JPanel();
            dpanel2 = new JPanel();
            dlabel1 = new JLabel("Filename:     " +
String.valueOf(temp4));
            dlabel2 = new JLabel("Title:     " + temp5);
            dbutton1 = new JButton("Ok");
            dbutton2 = new JButton("Cancel");
            dbutton1.addActionListener(this);
            dbutton1.setMnemonic(KeyEvent.VK_O);
            dbutton2.addActionListener(this);
            dbutton2.setMnemonic(KeyEvent.VK_C);
            dpanel1.setLayout(new BoxLayout(dpanel1,
BoxLayout.Y_AXIS));
            dpanel1.add(new JLabel(" "));
            dpanel1.add(dlabel1);
            dpanel1.add(new JLabel(" "));
            dpanel1.add(dlabel2);
            dpanel1.add(new JLabel(" "));
            dpanel2.add(dbutton1);
            dpanel2.add(dbutton2);
            getContentPane().add(dpanel1, BorderLayout.CENTER);
            getContentPane().add(dpanel2, BorderLayout.SOUTH);
            getContentPane().add(new JLabel("   "),
      BorderLayout.EAST);
            getContentPane().add(new JLabel("   "),
      BorderLayout.WEST);
            }
      public void actionPerformed(ActionEvent e)
            {
```

```
            if(e.getSource() == dbutton1)
                  {
                  Object[] options = {"Delete", "Do Not Delete"};
                  n = JOptionPane.showOptionDialog(this, "Are you
sure you want to delete this record?", "Confirm Delete",
JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null,
options, options[1]);
                  if(n == JOptionPane.YES_OPTION)
                        {
                        ok = true;
                        rpint1 = temp4;
                        dindex = database_system("DELETE FROM
document WHERE filename = ?", rpint1, "", 1, 0);
                        newvector1 = new Vector();
                        newvector2 = new Vector();
                        newvector3 = new Vector();
                        newint1 = 0;
                        newint2 = 0;
                        delimetersset = ".,;:?!#&* \t\n\r";
                        newmachine1 = new StringTokenizer(temp5,
delimetersset);
                        while(newmachine1.hasMoreTokens() == true)
                              {
                              tempstring1 = newmachine1.nextToken();
                              newvector1.add(newint1, tempstring1);
                              newint1++;
                              }
                        for(tint2 = 0; tint2 < newint1; tint2++)
                              {
                              tstring1 = (String)
(newvector1.elementAt(tint2));
                              tint3 = tint2;
                              while((tint3 > 0) &&
(newvector1.elementAt(tint3 - 1).toString().compareTo(tstring1) > 0))
                                    {
                                    newvector1.set(tint3,
newvector1.elementAt(tint3-1));
                                    tint3 = tint3 - 1;
                                    }
                              newvector1.set(tint3, tstring1);
                              }
                        while(newmachine2.hasMoreTokens() == true)
                              {
                              tempstring1 = newmachine2.nextToken();
                              newvector2.add(newint2, tempstring1);
                              newint2++;
                              }
                        k = 0;
                        size1 = newvector1.size();
                        for(newtemp1 = 0; newtemp1 < size1;
      newtemp1++)
                              {
                              if(find("content", (String)
(newvector1.elementAt(newtemp1))) == true)
                                    {
```

```
                                    newint3 = getindex((String)
(newvector1.elementAt(newtemp1)));
            newvector3.addElement(newvector1.elementAt(newtemp1));
                                    newvector2.addElement(new
Integer(newint3));

engine1.engEvalString("determine(" + String.valueOf(newint3-k) +
")");
                                    engine1.engEvalString("load
newvar1");
                                    yy =
engine1.engGetScalar("newvar1");
                                    if(yy == (double)(0))
                                          {

engine1.engEvalString("deleteword(" + String.valueOf(newint3 - k) +
")");
                                    k++;
                                          }
                                    }
                              }
                        newvoid1(newvector3);
                        semantic_analyzer(3, dindex, dvector1);
                        if(ok = true)
JOptionPane.showMessageDialog(this, "[1] record deleted.", "Delete
Succeeded", JOptionPane.INFORMATION_MESSAGE, null);
                        this.dispose();
                        }
                  }
            if(e.getSource() == dbutton2) this.dispose();
            }
      }
class search_interface extends JDialog implements ActionListener
      {
      JButton sbutton1, sbutton2;
      JLabel slabel1, slabel3, slabel4, slabel5;
      JPanel spanel1, spanel2;
      JTextField stextfield1;
      String rpstring1;
      Vector rsvector1;
      double[] sarraya;
      public search_interface(retriever1 mother)
            {
            super(mother, "Search", true);
            setSize(500, 170);
            getContentPane().setLayout(new
BoxLayout(getContentPane(), BoxLayout.Y_AXIS));
            sicon1 = new ImageIcon("right_arrow.gif");
            sbutton1 = new JButton("Go", sicon1);
            sbutton2 = new JButton("Stop");
            slabel1 = new JLabel("Search:");
            stextfield1 = new JTextField(30);
            spanel1 = new JPanel();
            spanel2 = new JPanel();
            sbutton1.addActionListener(this);
            sbutton1.setVerticalTextPosition(AbstractButton.CENTER);
```

```
sbutton1.setHorizontalTextPosition(AbstractButton.LEADING);
            sbutton2.addActionListener(this);
            sbutton2.setMnemonic(KeyEvent.VK_S);
            spanel1.setLayout(new FlowLayout());
            spanel1.add(slabel1);
            spanel1.add(stextfield1);
            spanel1.add(sbutton1);
            spanel2.add(sbutton2);
            getContentPane().add(new JLabel(" "));
            getContentPane().add(spanel1);
            getContentPane().add(spanel2);
            }
      public void actionPerformed(ActionEvent e)
            {
            if(e.getSource() == sbutton1)
                  {
                  rpstring1 = stextfield1.getText();
                  if(rpstring1.trim().length() == 0)
JOptionPane.showMessageDialog(this, "Type your query then click on Go
to begin your search.", "Query Missing",
JOptionPane.INFORMATION_MESSAGE, null);
                  else
                        {
                        double[][] s;
                        length1 = 0;
                        rsvector1 = new representation(rpstring1);
                        if(rsvector1.isEmpty() == false)
                              {
                              search_engine that = new
search_engine(this, rsvector1);
                              that.setVisible(true);
                              }
                        else JOptionPane.showMessageDialog(this, "No
match found.", "Search Results", JOptionPane.INFORMATION_MESSAGE,
null);
                        }
                  }
            if(e.getSource() == sbutton2) this.dispose();
            }
      }
class text_processor extends Vector
      {
      int rtint1, rtint2;
      String delimetersset, token;
      StringTokenizer machine;
      text_processor(String ptstring1)
            {
            delimetersset = ".,;:?!#&* \t\n\r";
            machine = new StringTokenizer(ptstring1, delimetersset);
            while(machine.hasMoreTokens() == true)
                  {
                  token = machine.nextToken();
                  if(find("stop", token) == false)
                        {
                        if(find("content", token) == false)
```

```
                              {
                              rtint2 = insertword(token);
                        engine1.engEvalString("einsertword(" +
String.valueOf(rtint2) +")");
                              }
                        }
                  }
            this.addAll(new representation(ptstring1));
            }
      }
class search_engine extends JDialog implements ActionListener
      {
      int i, index1, i0, length1, selectedRow;
      record1 r0, r1;
      double[] sarray1;
      Double rsdouble1;
      JButton b1;
      JLabel vlabel1;
      JList list1;
      JPanel rpanel1, rpanel2, rpanel3;
      JScrollPane pane1;
      JTable vtable1;
      ListSelectionModel lsm, rowsm;
      Object selectedFilename, selectedTitle;
      PreparedStatement st1;
      ResultSet rs;
      String vstring1;
      Vector vector1, vector2, vector3, vector4, v1;
      public search_engine(search_interface mother, Vector rsvector1)
            {
            super(mother, "Search Results", true);
            setSize(700, 400);
            getContentPane().setLayout(new BorderLayout());
            index1 = 0;
            vlabel1 = new JLabel("     Search Results:");
            b1 = new JButton("Ok");
            rpanel1 = new JPanel();
            rpanel2 = new JPanel();
            rpanel3 = new JPanel();
            vector1 = new Vector();
            vector2 = new Vector();
            b1.addActionListener(this);
            b1.setMnemonic(KeyEvent.VK_O);
            vector1.add(0, "Filename");
            vector1.add(1, "Title");
            try
                  {
                  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                  }
            catch(ClassNotFoundException e)
                  {
                  ok = false;
                  JOptionPane.showMessageDialog(this,
e.getMessage(), "Driver Class Not Found Error",
JOptionPane.ERROR_MESSAGE);
                  }
            try
                  {
                  conn = DriverManager.getConnection("jdbc:odbc:MS
Access Database");
                  st1 = conn.prepareStatement("SELECT * FROM
document ORDER BY filename");
                  rs = st1.executeQuery();
                  i0 = 0;
                  while (rs.next() == true)
                        {
                        vector3 = new Vector();
                        r0 = new record1();
                        r0.FILENAME = rs.getInt("filename");
                        r0.TITLE = rs.getString("title");
                  vector3.addElement(String.valueOf(r0.FILENAME));
                        vector3.addElement(r0.TITLE);
                        vector2.add(i0, vector3);
                        i0++;
                        }
                  conn.close();
                  }
            catch(SQLException f)
                  {
                  JOptionPane.showMessageDialog(this, f.getMessage()
+ "\nError Code: " + f.getErrorCode() + ".\nSQL State: " +
f.getSQLState() + ".", "Search Engine Failure",
JOptionPane.ERROR_MESSAGE);
                  ok = false;
                  }
            vector4 = new Vector();
            v1 = new Vector();
            sarray1 = search(rsvector1);
            if((int)sarray1[0] > 0)
                  {
                  for(i = 1; i <= (int)(sarray1[0]); i++)
                        {
                        v1.add(i - 1, new Double(sarray1[i] - 1));
                        }
                  for(i = 1; i <= sarray1[0]; i++)
                        {
                        rsdouble1 = (Double)(v1.elementAt(i - 1));
                        vector4.add(i - 1,
vector2.elementAt(rsdouble1.intValue()));
                        }
                  }
            vtable1 = new JTable(vector4, vector1);
            vtable1.setPreferredScrollableViewportSize(new
Dimension(600, 250));
      vtable1.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

vtable1.getColumnModel().getColumn(0).setPreferredWidth(1);
            if((int)sarray1[0] > 0)
                  {
                  pane1 = new JScrollPane(vtable1);
                  }
            rpanel1.setLayout(new BoxLayout(rpanel1,
BoxLayout.Y_AXIS));
            rpanel1.add(new JLabel(" "));
            rpanel1.add(vlabel1);
            rpanel1.add(new JLabel(" " +
String.valueOf((int)sarray1[0]) + " match(es) found."));
            if((int)sarray1[0] > 0) rpanel2.add(pane1);
            rpanel3.add(b1);
            getContentPane().add(rpanel1, BorderLayout.NORTH);
            getContentPane().add(rpanel2, BorderLayout.CENTER);
            getContentPane().add(rpanel3, BorderLayout.SOUTH);
            getContentPane().add(new JLabel("   "),
      BorderLayout.EAST);
            getContentPane().add(new JLabel("   "),
      BorderLayout.WEST);
            rowsm = vtable1.getSelectionModel();
            rowsm.addListSelectionListener(new
      ListSelectionListener()
                  {
                  public void valueChanged(ListSelectionEvent e)
                        {
                        if(e.getValueIsAdjusting()) return;
                        lsm = (ListSelectionModel)e.getSource();
                        selectedRow = lsm.getMinSelectionIndex();
                        selectedFilename =
vtable1.getValueAt(selectedRow, 0);
                        selectedTitle =
vtable1.getValueAt(selectedRow, 1);
                        vstring1 = selectedFilename.toString();
                        temp4 = Integer.parseInt(vstring1);
                        temp5 = selectedTitle.toString();
                        }
                  });
            }
      public void actionPerformed(ActionEvent e)
            {
            if (e.getSource() == b1) this.dispose();
            }
      }
public boolean find(String table, String word)
      {
      boolean found;
      int count1;
      PreparedStatement st1;
      ResultSet rs;
      count1 = 0;
      found = false;
      try
            {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            }
      catch(ClassNotFoundException e)
            {
            ok = false;
            JOptionPane.showMessageDialog(this, e.getMessage(),
"Driver Class Not Found Error", JOptionPane.ERROR_MESSAGE);
```

```
            }
      try
            {
            conn = DriverManager.getConnection("jdbc:odbc:MS Access
Database");
            st1 = conn.prepareStatement("SELECT COUNT(*) FROM " +
table + " WHERE " + table + "word = ?");
            st1.setString(1, word);
            rs = st1.executeQuery();
            rs.next();
            count1 = rs.getInt(1);
            conn.close();
            }
      catch(SQLException f)
            {
            JOptionPane.showMessageDialog(this, f.getMessage() +
"\nError Code: " + f.getErrorCode() + ".\nSQL State: " +
f.getSQLState() + ".", "Database Search Failed",
JOptionPane.ERROR_MESSAGE);
            ok = false;
            }
      if(count1 == 1) found = true;
      return found;
      }
public int getindex(String word)
      {
      int temp;
      PreparedStatement st1;
      ResultSet rs;
      temp = 0;
      try
            {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            }
      catch(ClassNotFoundException e)
            {
            ok = false;
            JOptionPane.showMessageDialog(this, e.getMessage(),
"Driver Class Not Found Error", JOptionPane.ERROR_MESSAGE);
            }
      try
            {
            conn = DriverManager.getConnection("jdbc:odbc:MS Access
Database");
            st1 = conn.prepareStatement("SELECT COUNT(*) FROM
content WHERE contentword < ?");
            st1.setString(1, word);
            rs = st1.executeQuery();
            rs.next();
            temp = rs.getInt(1) + 1;
            conn.close();
            }
      catch(SQLException f)
            {
            JOptionPane.showMessageDialog(this, f.getMessage() +
"\nError Code: " + f.getErrorCode() + ".\nSQL State: " +
f.getSQLState() + ".", "Word Insert Failed",
JOptionPane.ERROR_MESSAGE);
            ok = false;
            }
      return temp;
      }
public int getnewfilename()
      {
      int temp, temp0, temp1;
      PreparedStatement st1, st2;
      ResultSet rs, rs2;
      temp = 0;
      try
            {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            }
      catch(ClassNotFoundException e)
            {
            ok = false;
            JOptionPane.showMessageDialog(this, e.getMessage(),
"Driver Class Not Found Error", JOptionPane.ERROR_MESSAGE);
            }
      try
            {
            conn = DriverManager.getConnection("jdbc:odbc:MS Access
Database");
            st1 = conn.prepareStatement("SELECT * FROM document
ORDER BY filename");
            rs = st1.executeQuery();
            while(rs.next() == true)
                  {
                  temp = rs.getInt(1);
                  }
            conn.close();
            temp = temp + 1;
            }
      catch(SQLException f)
            {
            JOptionPane.showMessageDialog(this, f.getMessage() +
"\nError Code: " + f.getErrorCode() + ".\nSQL State: " +
f.getSQLState() + ".", "Word Insert Failed",
JOptionPane.ERROR_MESSAGE);
            ok = false;
            }
      return temp;
      }
public int insertword(String word)
      {
      int temp;
      PreparedStatement st1, st2;
      ResultSet rs;
      temp = 0;
      try
            {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            }
      catch(ClassNotFoundException e)
            {
            ok = false;
            JOptionPane.showMessageDialog(this, e.getMessage(),
"Driver Class Not Found Error", JOptionPane.ERROR_MESSAGE);
            }
      try
            {
            conn = DriverManager.getConnection("jdbc:odbc:MS Access
Database");
            st1 = conn.prepareStatement("INSERT INTO content
VALUES(?)");
            st1.setString(1, word);
            st1.executeUpdate();
            st2 = conn.prepareStatement("SELECT COUNT(*) FROM
content WHERE contentword < ?");
            st2.setString(1, word);
            rs = st2.executeQuery();
            rs.next();
            temp = rs.getInt(1) + 1;
            conn.close();
            }
      catch(SQLException f)
            {
            JOptionPane.showMessageDialog(this, f.getMessage() +
"\nError Code: " + f.getErrorCode() + ".\nSQL State: " +
f.getSQLState() + ".", "Word Insert Failed",
JOptionPane.ERROR_MESSAGE);
            ok = false;
            }
      return temp;
      }
public int database_system(String stmt, int dfilename, String
dtitle, int p1, int p2)
      {
      boolean right;
      int dindex;
      PreparedStatement st1, st2;
      ResultSet rs;
      String message1;
      dindex = 0;
      right = true;
      try
            {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            }
      catch(ClassNotFoundException e)
            {
            ok = false;
            JOptionPane.showMessageDialog(this, e.getMessage(),
"Driver Class Not Found Error", JOptionPane.ERROR_MESSAGE);
            }
      try
            {
            conn = DriverManager.getConnection("jdbc:odbc:MS Access
Database");
```

```
            st1 = conn.prepareStatement(stmt);
            st1.setInt(p1, dfilename);
            if(p2 != 0) st1.setString(p2, dtitle);
            st1.executeUpdate();
            st2 = conn.prepareStatement("SELECT COUNT(*) FROM 
document WHERE filename < ?");
            st2.setInt(1, dfilename);
            rs = st2.executeQuery();
            rs.next();
            dindex = rs.getInt(1) + 1;
            conn.close();
            }
      catch(SQLException f)
            {
            message1 = "";
            ok = false;
            right = false;
            dindex = -1;
            if(String.valueOf(f.getErrorCode()) == "0" && 
String.valueOf(f.getSQLState()) == "S1000") message1 = "The title you 
have entered may already exist in the database.  Try entering a 
different title.";
            JOptionPane.showMessageDialog(this, f.getMessage() + 
"\nError code: " + f.getErrorCode() + ".\nSQL State: " + 
f.getSQLState() + ".\n" + message1, "Database System Failure", 
JOptionPane.ERROR_MESSAGE);
            }
      if(right == true) switch(p2)
            {
            case 2:
                  {
                  engine1.engEvalString(gint1, "einsert1(" + 
String.valueOf(dindex) + ")");
                  break;
                  }
            case 1:
                  {
                  engine1.engEvalString(gint1, "emodify1(" + 
String.valueOf(dindex) + ")");
                  break;
                  }
            case 0:
                  {
                  engine1.engEvalString(gint1, "edelete1(" + 
String.valueOf(dindex) + ")");
                  break;
                  }
            }
      return dindex;
      }
public void actionPerformed(ActionEvent e)
      {
      int n;
      Object[] options = {"Yes", "No"};
      ok = true;
      if(e.getSource() == rbutton1)
```

```
            {
            insert_interface that1 = new insert_interface(this);
            that1.setVisible(true);
            }
      if(e.getSource() == rbutton2)
            {
            display_interface that2 = new display_interface(this);
            that2.setVisible(true);
            }
      if(e.getSource() == rbutton3)
            {
            display_interface0 that3 = new display_interface0(this);
            that3.setVisible(true);
            }
      if(e.getSource() == rbutton4)
            {
            search_interface that4 = new search_interface(this);
            that4.setVisible(true);
            }
      if(e.getSource() == rbutton5)
            {
            n = JOptionPane.showConfirmDialog(this, "Are you sure
you want to quit Retriever?", "Quit Retriever 1.0",
JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null);
            if(n == JOptionPane.YES_OPTION)
                  {
                  engine1.engClose(gint1);
                  engine1.kill();
                  System.exit(0);
                  }
            }
      }
public void newdelete(String word)
      {
      boolean found;
      int count1;
      PreparedStatement st1;
      ResultSet rs;
      found = false;
      count1 = 0;
      try
            {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            }
      catch(ClassNotFoundException e)
            {
            ok = false;
            JOptionPane.showMessageDialog(this, e.getMessage(),
"Driver Class Not Found Error", JOptionPane.ERROR_MESSAGE);
            }
      try
            {
            conn = DriverManager.getConnection("jdbc:odbc:MS Access
Database");
            st1 = conn.prepareStatement("DELETE FROM content WHERE
contentword = ?");
```

```
            st1.setString(1, word);
            st1.executeUpdate();
            conn.close();
            }
      catch(SQLException f)
            {
            JOptionPane.showMessageDialog(this, f.getMessage() +
"\nError Code: " + f.getErrorCode() + ".\nSQL State: " +
f.getSQLState() + ".", "Database Search Failed",
JOptionPane.ERROR_MESSAGE);
            ok = false;
            }
      }
public void newvoid1(Vector newvector3)
      {
      boolean newboolean1;
      int newtemp2, size2;
      size2 = newvector3.size();
      for(newtemp2 = 0; newtemp2 < size2; newtemp2++)
            {
            newdelete((String)(newvector3.elementAt(newtemp2)));
            }
      }
public void semantic_analyzer(int source1, int dindex, Vector
rsvector1)
      {
      switch(source1)
            {
            case 1:
                  {
                  engine1.engEvalString(gint1, "eassign(" +
String.valueOf(dindex) + ", " + String.valueOf(rsvector1) + ")");
                  break;
                  }
            case 2:
                  {
                  engine1.engEvalString(gint1, "eassign(" +
String.valueOf(dindex) + ", " + String.valueOf(rsvector1) + ")");
                  break;
                  }
            case 3:
                  {
                  engine1.engEvalString(gint1, "eanalyze1");
                  break;
                  }
            }
      }
public static void main(String[] args)
      {
      retriever1 that = new retriever1();
      that.setVisible(true);
      }
public double[] search(Vector rsvector1)
      {
      double[] rsvector3;
      double[][] rsvector2;
      engine1.engEvalString(gint1, "rsearch(" +
String.valueOf(rsvector1) + ")");
      engine1.engEvalString(gint1, "load retriever.mat evector1");
      rsvector2 = engine1.engGetArray(gint1, "evector1");
      rsvector3 = rsvector2[0];
      return rsvector3;
      }
public retriever1()
      {
      super("Retriever");
      setSize(350, 350);
      addWindowListener(new WindowAdapter()
            {
            public void windowClosing(WindowEvent e)
                  {
                  engine1.engClose(gint1);
                  engine1.kill();
                  System.exit(0);
                  }
            });
      getContentPane().setLayout(new BorderLayout());
      rfont1 = new Font("Courier", Font.PLAIN, 12);
      ricon1 = new ImageIcon("v4_java_logo.gif");
      rbutton1 = new JButton("Insert Document", ricon1);
      rbutton2 = new JButton("Modify Document", ricon1);
      rbutton3 = new JButton("Delete Document", ricon1);
      rbutton4 = new JButton("Search", ricon1);
      rbutton5 = new JButton("Quit");
      label01 = new JLabel("     ");
      label02 = new JLabel("");
      label03 = new JLabel("                    ");
      label04 = new JLabel("                    ");
      label05 = new JLabel("");
      label06 = new JLabel("");
      label07 = new JLabel("");
      label08 = new JLabel("");
      panel4 = new JPanel();
      panel5 = new JPanel();
      panel6 = new JPanel();
      rbutton1.addActionListener(this);
      rbutton1.setVerticalTextPosition(AbstractButton.CENTER);
      rbutton1.setHorizontalTextPosition(AbstractButton.TRAILING);
      rbutton1.setFont(rfont1);
      rbutton1.setMnemonic(KeyEvent.VK_I);
      rbutton2.addActionListener(this);
      rbutton2.setVerticalTextPosition(AbstractButton.CENTER);
      rbutton2.setHorizontalTextPosition(AbstractButton.TRAILING);
      rbutton2.setFont(rfont1);
      rbutton2.setMnemonic(KeyEvent.VK_M);
      rbutton3.addActionListener(this);
      rbutton3.setVerticalTextPosition(AbstractButton.CENTER);
      rbutton3.setHorizontalTextPosition(AbstractButton.TRAILING);
      rbutton3.setFont(rfont1);
      rbutton3.setMnemonic(KeyEvent.VK_D);
      rbutton4.addActionListener(this);
      rbutton4.setVerticalTextPosition(AbstractButton.CENTER);
```

```
        rbutton4.setHorizontalTextPosition(AbstractButton.TRAILING);
        rbutton4.setFont(rfont1);
        rbutton4.setMnemonic(KeyEvent.VK_S);
        rbutton5.addActionListener(this);
        rbutton5.setMnemonic(KeyEvent.VK_Q);
        panel6.setLayout(new GridLayout(8, 1));
        panel6.add(rbutton4);
        panel6.add(label05);
        panel6.add(rbutton1);
        panel6.add(label06);
        panel6.add(rbutton2);
        panel6.add(label07);
        panel6.add(rbutton3);
        panel4.setLayout(new BorderLayout());
        panel4.setBorder(BorderFactory.createCompoundBorder(BorderFacto
ry.createTitledBorder("Task
Chooser"),BorderFactory.createEmptyBorder()));
        panel4.add(label01, BorderLayout.NORTH);
        panel4.add(label02, BorderLayout.SOUTH);
        panel4.add(label03, BorderLayout.EAST);
        panel4.add(label04, BorderLayout.WEST);
        panel4.add(panel6, BorderLayout.CENTER);
        panel5.add(rbutton5);
        getContentPane().add(new JPanel(), BorderLayout.NORTH);
        getContentPane().add(panel4, BorderLayout.CENTER);
        getContentPane().add(panel5, BorderLayout.SOUTH);
        getContentPane().add(new JPanel(), BorderLayout.EAST);
        getContentPane().add(new JPanel(), BorderLayout.WEST);
        engine1 = new JMatLink();
        gint1 = engine1.engOpenSingleUse();
        }
```

B. **einsert1.m Source Code**

```
function e1 = einsert1(dindex)
idindex = dindex;
load retriever.mat matrix1 matrix2 evector1;
if ~(isempty(matrix1))
      [m, n] = size(matrix1)
      matrixa = zeros(m, n + 1)
      matrixa = [matrix1 zeros(m, 1)]
end
matrix1 = matrixa
save retriever.mat matrix1 matrix2 evector1;
```

C. **emodify1.m Source Code**

```
function e2 = emodify1(dindex)
idindex = dindex;
load retriever.mat matrix1 matrix2 evector1;
if ~(isempty(matrix1))
      [m, n] = size(matrix1)
      matrix1(:, dindex) = 0
end
save retriever.mat matrix1 matrix2 evector1;
```

D. **edelete1.m Source Code**

```
function e3 = edelete1(dindex)
ddindex = dindex;
load retriever.mat matrix1 matrix2 evector1;
[m, n] = size(matrix1);
if n == 1
      matrixa = [];
else
      if ddindex == 1
            matrixa = [matrix1(:, ddindex + 1:end)];
      elseif ddindex == n
            matrixa = [matrix1(:, 1:ddindex - 1)];
      elseif ddindex > 1 & ddindex < n
            matrixa = [matrix1(:, 1:ddindex - 1) matrix1(:, ddindex
+ 1:end)];
      end
end
matrix1 = matrixa;
save retriever.mat matrix1 matrix2 evector1;
clear all;
```

E. **einsertword.m Source Code**

```
function e1 = einsertword(row)
row1 = row;
load retriever.mat matrix1 matrix2 evector1;
[m, n] = size(matrix1)
if isequal(n, 0)
      n = 1
end
matrixa = zeros(m + 1, n)
if isequal(row, 1)
      matrixa = [zeros(1, n); matrix1]
elseif isequal(row, m + 1) & ~isequal(m, 0)
      matrixa = [matrix1; zeros(1, n)]
```

```
elseif ~isequal(row, 1) & ~isequal(row, m + 1)
      matrixa = [matrix1(1: row - 1, :); zeros(1, n);
matrix1(row:end, :)]
end
matrix1 = matrixa
save retriever.mat matrix1 matrix2 evector1;
clear all;
```

F. **determine.m Source Code**

```
function found = determine(row)
row1 = row;
load retriever.mat matrix1 matrix2 evector1;
[m, n] = size(matrix1)
x = 1;
d1 = 0;
found = 0;
matrixa = matrix1;
while ~isequal(found, 1) & x <= n
      if matrixa(row1, x)
            found = 1;
      end
      x = x + 1
end
newvar1 = found
save newvar1
save retriever.mat matrix1 matrix2 evector1;
clear all;
```

G. **deleteword.m Source Code**

```
function d1 = deleteword(row)
row1 = row;
load retriever.mat matrix1 matrix2 evector1;
[m, n] = size(matrix1)
matrixa = zeros(m-1, n)
if isequal(row1, 1)
      matrixa = [matrix1(2:end, :)]
elseif isequal(row1, m)
      matrixa = [matrix1(1:m-1, :)]
elseif ~isequal(row1, 1) & ~isequal(row1, m) & ~isequal(row1, 0)
      matrixa = [matrix1(1:row-1, :); matrix1(row+1:end, :)]
end
matrix1 = matrixa
save retriever.mat matrix1 matrix2 evector1;
```

```
clear all;
```

H. **eassign.m Source Code**

```
function assign1 = eassign(dindex, rarray)
idindex = dindex;
iarray = rarray;
load retriever.mat matrix1 matrix2 evector1;
ea1 = length(rarray)
[m, n] = size(matrix1)
r1 = 1
for r2 = r1:ea1
      matrix1(rarray(r2), dindex) = 1
end
[m, n] = size(matrix1)
ivar2 = round(.1*n) + 1
[U, S, V] = svds(matrix1, ivar2)
matrix2 = U*S*V'
save retriever.mat matrix1 matrix2 evector1;
clear all;
```

I. **eanalyze1.m Source Code**

```
function es1 = eanalyze1
load retriever.mat matrix1 matrix2 evector1;
[m, n] = size(matrix1)
ivar2 = round(.1*n) + 1
[U, S, V] = svds(matrix1, ivar2)
matrix2 = U*S*V'
save retriever.mat matrix1 matrix2 evector1;
clear all;
```

J. **rsearch.m Source Code**

```
function a = rsearch(rsvector1)
load retriever.mat matrix1 matrix2 evector1;
[m, n] = size(matrix2)
matrix3 = zeros(m, n)
vector1 = zeros(m, 1)
matrix4 = zeros(1, n)
```

```
p = length(rsvector1)
for s1 = 1:p
      vector1(rsvector1(s1)) = 1;
end
for s2 = 1:n
      matrix3(1:end, s2) = matrix2(1:end, s2) - vector1
      matrix4(s2) = sqrt(matrix3(1:m, s2)'*matrix3(1:m, s2))
end
threshold = 1.3; %temporary
[dvalues, dindex1] = sort(matrix4)
indeces = length(matrix4(matrix4 < threshold))
dindex = dindex1(1:indeces)
t1 = length(dindex)
t2 = [t1 dindex]
evector1 = t2
save retriever.mat matrix1 matrix2 evector1;
save dindex;
clear all;
```

XII. ACKNOWLEDGEMENT

GOD;

Edgardo Mojar Sr., Nenita Mojar, Karla Mojar, Jennifer Mojar, Edgardo Mojar Jr., David Mojar, Paulo Mojar, Mandy Parra Jr.;

Prof. Gregorio Baes, Dr. Vincent Magboo, Prof. Sheila Magboo, Ms. Celina Umagtang, Mr. Philip Zamora;

Magsico clan, Acuesta family, Despuig family;

Enrico Agodilla, Gabriel Fernandez, Joey Donasco, Josephine Dungog, Joynivee Chua, Lloyd Meily, Melissa Del Poso, Trina Quijano;

Beverly Manzano, Rowena Malinay, Annaliza Miranda, Rebecca Militante;

Annaliza Pagkalinawan, Eddie Parra, Eden Huelgas, Francia Abarquez, John Lat, John Solatario, Lourdes Mojar, Manual Dala, Myra Aquino, Myrna Cañete, Oliver Roxas, Rose Paiso, Ruel Cantila, Ruth Dadia, Shulamita Fino, Ms. Alma, Mrs. Chavez, Mrs. Roxas, Alex, John, Nyde;

Alta Vista Company, Avon Cosmetics, http://citeseer.nj.nec.com, http://forums.devshed.com, http://javaalmanac.com, http://linux.wku.edu, http://www.codeguru.com, http://www.experts-exchange.com, http://www.devx.com, http://www.held-mueller.de, http://www.infopeople.org, http://www.isacat.net, http://www.java.sun.com, http://www.mathworks.com, http://www.microsoft.com, http://www.objectweb.org, http://www.proverbium.com, http://www.qvtrek.com, http://www.sis.buffalo.edu, *IEEE Computer* Journal, *Journal of Medical Internet Research*, NEC Research Institute, Pedro E. Diaz High School Annex, Pfizer Virtual Library, *Science* Journal, The Internet Society, Word International Ministries Alabang Incorporated Young Adults' Ministry, University of the Philippines Manila Department of Physical Sciences and Mathematics, University of the Philippines Manila Office of the College Secretary;

Listana family, Rev. Jun Insigne, Genesis Dadia, Jeffrey Canlas, Jennifer Magsico, Joel Mallillin, very special thanks to Ms. Eleanor Despuig.