An Investigation of the Latent Semantic Analysis Technique for Document Retrieval

STUDENT PROJECT REPORT

SUBMITTED BY

David Muchangi Mugo

Matri.No:34348

SUPERVISED BY

Prof. Dr. rer. nat. Ralf Möller

MSc. Atila Kaya

Abstract

Latent semantic analysis (LSA) application in information retrieval promises to offer better performance by overcoming some limitations that plagues traditional term-matching techniques. These term-matching techniques have always relied on matching query terms with document terms to retrieve the documents having terms matching the query terms. However, by use of these traditional retrieval techniques, users' needs have not been adequately served. While users want to search through information based on conceptual content, natural languages have limited the expression of these concepts. They present synonymy problem (a situation where several words may have the same meaning) and polysemy problem (a situation where a word may have several meanings). Due to these natural language problems, individual words contained in users' queries, may not explicitly specify the intended user's concept, which may result in the retrieval of some irrelevant documents. LSA seems to be a promising technique in overcoming these natural language problems especially synonymy problem. It deals with exploiting the global relationships between terms and documents and then mapping these documents and terms in a proximity space, where terms and documents that are closely related are mapped close to each other in this space. Queries are then mapped to this space with documents being retrieved based on similarity measures. In this report, LSA performance in documents retrieval is investigated and compared with traditional term-matching techniques.

Declaration

I hereby declare that I have personally authored this entire project report and that no other sources other than the ones listed at the end of the report have been used. The report is therefore, original and it has not been published in Germany or in any other country. All the registered trademarks, names, logos and icons appearing in this report are the property of their respective owners.

Signed:…………………………………………………….Date:…………………………….

Acknowledgement

I am very grateful to Prof. Ralf Möller, for his lecture on Multimedia Content Management which provided a strong basis for the start of this project. I acknowledge also in a special way, the assistance and guidance I received from Mr. Atila Kaya throughout the project. Finally, I thank the management of Körber Ag, particularly Dr. Dirk Friedrich who was my company supervisor, for providing a convenient environment and support I required for the execution of the project.

Table of Contents

1. Motivation

1.1 Introduction

Information retrieval (IR) [3] is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).We are living in an era where information is becoming a very important determinant for the success of governments, corporations and individuals. In the same breadth, information technology is growing at a tremendous pace to cater for the ever increasing information needs. In the World Wide Web, we have witnessed unprecedented growth in the number of web pages and the webpage structure as new technologies emerge. More growth is even expected as the need to make content on the web machine-readable as envisioned in the semantic web goes closer to reality. As a result of many enabling technologies and the need to serve information needs better, we have seen a rapid increase in the electronic content as more organisations and individuals shift from traditional filing systems to electronic content storages. If users' information needs have to be served better, then more research is required to have better retrieval techniques for relevant documents.

In many areas, document retrieval has relied on literally matching terms contained in documents with those found in queries. However, natural languages present some challenges that have inevitably rendered these existing term-matching methods inaccurate. This basically arises from the fact that in many natural languages a word can have several meanings (polysemy) and the fact that many natural languages words can have same meaning (synonymy). It is for this reason that latent semantic analysis [1] promises to overcome the problems of lexical matching by using statistically derived conceptual indices instead of individual words for retrieval. Latent semantic analysis promises to address this by harnessing the global relationships between the documents and terms with a view of establishing a common semantic space from which queries can be answered. It assumes that there is some underlying latent semantic structure within a collection of documents which is obscured by the randomness of the words used in the documents. Statistical techniques are used to estimate this semantic structure and then to map it on a proximity space with a view to querying documents better based on their semantic content as opposed to pure terms.

Latent semantic analysis seems to provide strong application prospects in the area of junk E-mail filtering, semantic classification of documents, clustering of documents and information retrieval among others. In information retrieval, documents and queries are projected into a common proximity space or semantic space, from where similarity measures are applied for retrieval purposes as well as for ranking the results.

1.2 Background

As companies try to move from paperless offices to electronic resources, the need to have a better document management arises. The management of electronic documents addresses the following fundamental concerns:

a) The location of documents
b) Filing of documents
c) Retrieval of documents
d) Security aspects of documents
e) Recovery of documents in case of disasters
f) Retention period for the documents
g) Archiving
h) Workflow
i) Creation
j) Authentication

This project focuses on documents retrieval based on the current scenario within Körber Group. The main idea is to investigate LSA performance and thereafter recommend a software tool for document retrieval within the company based on this LSA performance.

1.2.1 Körber Ag

Körber Ag is a management holding company with over 30 independent, internationally active companies in ten countries in Europe, America and Asia with numerous sales and service locations throughout the world. The Körber Group, with 9,200 employees, achieved sales of EUR 1.6 billion in 2006. The Group companies develop, produce and sell precision machines in the sectors of tobacco, paper and machine tools as well as pharmaceutical packaging systems and electronics components.

1.2.2 Current challenge within engineering environment

Within the Engineering Environment of the Körber Group, there is a huge amount of documentation as projects are planned and executed for production of various products. Some of these documents are printed and filed using physical filing systems whereas many others are just stored as soft copies in computer discs. Despite the huge amount of text documents produced within the company, there still exists no central content management system for these documents necessitating an investigation of a tool that could be employed in the future to better manage the documents. The investigation of latent semantic analysis for document retrieval is based on the hypothesis that LSA technique performs better in retrieving the highest number of relevant documents as opposed to term matching techniques and that a software tool implementing the technique exists.

1.3 Project Objectives

i. To investigate the application of LSA for document retrieval based on performance as compared with term-matching techniques
ii. To analyse the state of the art in software tools with a view towards recommending a suitable solution for document retrieval within the company on the basis of LSA performance.

1.4 Document structure

This report starts with a brief motivation in chapter one that leads to the execution of the project. Chapter two opens with an introduction to the LSA technique and further explains LSA foundations in more details. Chapter three presents the experiments carried out and the discussion of the results from these experiments. Finally chapter four follows with a brief conclusion and points out areas for future work.

2. Latent Semantic Analysis

2.1 Latent Semantic Analysis Overview

Latent semantic analysis (LSA) commonly known as Latent Semantic Indexing in the context of information retrieval, [9] is a fully automatic mathematical/statistical technique for extracting and inferring relations of expected contextual usage of words in passages of discourse. It is based on the application of a particular mathematical technique, called Singular Value Decomposition (SVD), to a word-by-document matrix .The word-by-document matrix is formed from LSA inputs that consist of raw text parsed into words defined as unique character strings and separated into meaningful passages or samples such as sentences or paragraphs. This application provides a way of viewing the global relationship between terms in the whole documents' collection enabling the semantic structures within the collection to be unearthed. LSA application in information retrieval is motivated by the challenges encountered in natural language processing where a word may have several meanings (polysemy) and several words may mean the same thing (synonymy) thereby presenting ambiguities in expressing users' concepts. For example [4], several empirical studies show that the likelihood of two people choosing the same keyword for a familiar object is less than 15%. It is due to these challenges that mere keywords searching techniques are inadequate in addressing user queries. LSA [2] enables retrieval on the basis of conceptual content, instead of merely matching words between queries and documents. By use of dimensionality reduction which makes complex natural languages problems tractable and the intrinsically global outlook of the approach which tends to complement the local optimization performed by more conventional techniques, it appears to be a much better approach for information retrieval. LSA also seems particularly attractive due to the mapping of discrete entities onto a continuous parameter space, where efficient machine learning algorithms can be applied.

LSA can be applied in many areas as long as there exists a set of identifiable individual units and a set of collections for these units. In information retrieval, it uses a set of individual terms (words) which are contained in a set of documents belonging to a document collection. LSA assumes that there exist latent semantic structures that are obscured by randomness of words in documents and which could be revealed by the application of a suitable technique. The process involves the analysis of the document collection to extract individual terms likely to be queried by users and then constructing a matrix of dimension m (number of individual terms) by n (number of documents in the repository). SVD is then applied to reveal the semantic structures using this term-document matrix. Apart from this, SVD is applied to reduce the dimensionality of the term document matrix which is very sparse, therefore optimizing the storage as well as removal of noise in the data. User queries are answered based on this reduced space. In this reduced space, LSA is able through [13] the pattern of co-occurrences of words to infer the structure of relationships between documents and words. More details on LSA steps are explained later.

2.2 Measuring the performance of information retrieval systems

A retrieval system's performance is described based on two measures namely: *recall* and *precision.*

- *Recall:* gives the fraction of the relevant documents in the collection that a system returns [3]. Therefore, it measures the ability of the retrieval system to present all relevant items. A recall of 100% is achieved if all documents are retrieved.

$$\text{Recall} := \frac{\left|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}\right|}{\left|\{\text{relevant documents}\}\right|}$$

- *Precision:* gives the fraction of the returned results that are relevant to the information needs [3].It therefore measures the accuracy of the retrieval system to satisfy the users' needs.

$$\text{Precision} := \frac{\left|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}\right|}{\left|\{\text{retrieved documents}\}\right|}$$

2.3 Deficiencies of the term-matching retrieval technique

Current term-matching retrieval systems try to answer queries based on the matching of query terms and document terms. This to some extent, satisfies user needs, however many documents that are relevant to the user may be left out in the result set or irrelevant documents may be retrieved. The following reasons which arise from natural language limitations (polysemy and synonymy) explain this poor performance:

i. Indexes do not contain a combination of all the terms that users might use to search documents but a subset of these terms.
ii. Techniques that have been used to solve synonymy like thesaurus may present more problems, for instance, added words may have different meanings than the intended, therefore causing more degradation on precision measure.
iii. There is no adequate automatic method for dealing with polysemy. Approaches to use controlled vocabularies and human intermediaries to act as translators have been rendered extremely expensive and less effective.
iv. Bag of words model: Each word type [6] is treated as independent of any other thus matching both of two terms that almost always occur together is counted as heavily as matching two that are rarely found in the same document. This fails to take redundancy into account which may result to distortion of the results to an unknown degree.

2.4 Latent Semantic Analysis process

2.4.1 Pre-processing stage

Documents contain many terms some of which are very common while others are very rare with their occurrence depended on the document topic. Due to these common terms, documents are analysed in this stage to extract only the key words to be used in the construction of the term-document matrix. This involves several processes:

Elimination of stop words

This step involves elimination of common words which have less discrimination power for documents on answering users' queries. A strategy to eliminate stop words may involve extracting all terms that appear in the whole document collection and then eliminating those with high occurrence frequency in each document. For instance in English language, the following words may have a high occurrence frequency in a document collection and therefore would be discarded in the construction of term-document matrix:

> *a, an, and, are, as, at, be, by, for, from, has, he, in, is, it, its, of, on, that, the, to, was, were, will & with.*

Discarding of stop words may however affect the performance of a retrieval system negatively. This is due to the fact that some stop words may contribute to the topical meaning of a document, for example some song titles might be composed of stop words only, e.g. *To be or not to be, Let It Be* and *I don't want to be* among others. Elimination of stop words in this case results in the exclusion of the affected documents from the retrieval process.

Stemming and lemmatization

In natural languages, we have words taking different structures but with almost the same basic meaning. For instance, in English language, we can have within the same document words like: *organize*, *organizing* & *organizes* which would make the dimension of the term-document matrix unnecessarily large. It would be reasonable to just store the word *organize* instead of the three words. Stemming and lemmatization tries to achieve this goal. Stemming [3] usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal most of the time and often it may include the removal of derivational affixes. Lemmatization [3] refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. There are many algorithms that exist for doing stemming and lemmatization and the choice depends on the application authors. In stemming English words the most common and the most empirically active algorithm is Porter's algorithm. This algorithm has been described briefly in [3]. An example of stemming is shown below:

connection		
connections		
connective	--->	*connect*
connected		
connecting		

Stemming and lemmatization strategies used may affect the performance of the retrieval system. E.g. where a Porter stemmer is used, the following words might all be stemmed to *oper* which affects the querying for *operating system, operative dentistry* and *operational research*:

operate, operating, operates, operation, operative, operatives, operational

2.4.2 Weighing terms

The construction of the term-document matrix A (of dimensions m x n) follows after the extraction of the dictionary from the documents. Each entry $A_{i,j}$ of the term-document matrix represents the weight of i-th term in j-th document. There are many approaches that can be used for weighting terms in a collection. The following approaches are common:

a) Term frequency ($tf_{i,j}$)

This approach assigns weights to terms based on how many times a term *i* occurs in a document *j*. The critical problem faced by this approach is that all terms used are considered to be of equal relevance to the query answering process. This is worse where we have a great variation between the common terms and the infrequent terms. Consider the document shown on the next page:

ParseTreeView defines a view that is used to display the abstract syntax tree which is resulted by parsing the text content of an editor

If the above document is taken as input, the term *view* and *ParseTreeView* would have the same weight based on term frequency though *ParseTreeView* deserves more weight. The next problem encountered arises from differences in documents' lengths with terms likely to score highly within long documents where they are likely to occur more frequently as opposed to shorter documents. To prevent a bias towards large documents where terms may have unnecessarily high frequency, it is desirable that term frequency ($tf_{i,j}$) is normalized with a per document factor λ_j.

$$A_{i,j} = \frac{tf_{i,j}}{\lambda_j}$$

Depending on the application, different ways may be used to calculate λ_j *factor.*

Examples of λ_j *calculation*

I. $\lambda_j = \left|\left|tf_{i,j}\right|\right|1 := \sum_i \left|tf_{i,j}\right|$ where $\left|\left|.\right|\right|1$ is one-norm

II. $\lambda_j = \left|\left|tf_{i,j}\right|\right| := \sqrt{\sum_i f^2_{i,j}}$ where $\left|\left|.\right|\right|$ is two-norm

b) Tf-idf weighting

This weighting scheme combines the idea of term frequency ($tf_{i,j}$) and inverse document frequency (idf_i). Inverse document frequency tries to come up with a better discriminating mechanism for query answering by considering document-level statistics. The idea is to have terms that appear in many documents weigh less on a query than those terms which appear in fewer documents. The inverse document frequency of a term *i* in the collection of *N* documents is given by:

$$idf_i = log\,\frac{N}{df_i}$$

Where idf_i is the document frequency (number of documents in the whole collection that contains a term *i*).

For illustration purposes we use the example given in [3], which is from Reuters-RCVI collection of 806,791 documents. Logarithms are given to base 10.

i-thTerm	df_i	idf_i
Car	18,165	1.65
Auto	6723	2.08
Insurance	19,241	1.62
Best	25,235	1.5

Table 1. Illustrating inverse document frequency

From the table, it can be seen clearly that terms with high occurrence in many documents score less in terms of their idf_i. For the purpose of calculating the weights of each *i-th* term, we now combine the idea of document frequency with term frequency.

$$Tf\text{-}idf_{i,j} = tf_{i,j} \times idf_i$$

The above equation means that $tf\text{-}idf_{i,j}$ [1] assigns to a particular term *i* a weight in a document *j* that is:

- Highest when term *i* occurs many times within a small number of documents
- Lower when the term *i* occurs fewer times in a document or occurs in many documents
- Lowest when the term occurs in virtually all documents.

This approach of assigning weights is better than the term frequency discussed earlier since it combines both local and global statistics within a documents collection to predict the weight of *i-th* term in a *j-th* document.

2.4.3 Vector space model mapping

Once, a weighting scheme has been selected documents and their terms are mapped on to a vector space model (VSM). VSM is an algebraic model deployed to represent documents as vectors. This model can be visualized as a matrix A of dimension n (number of documents in the collection) by m (number of key terms in the collection). The vector space model involves every document being mapped to a vector c_j where $1 \leq j \leq n$ and c_j represents the columns of term-document matrix A. This is represented below:

$$A := (c_1, c_2, \ldots\ldots, c_n) = (r_1, r_2, \ldots, r_m)^T \in \mathbb{R}^{m\times n}\#$$

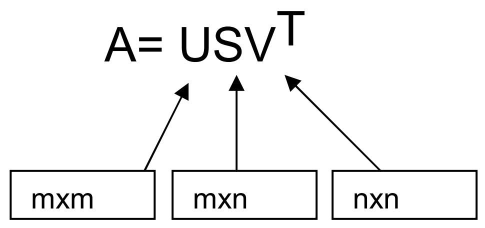
In the above equation $r_1, r_2, \ldots, r_m$ are the rows of the matrix *A* showing weights of terms in each document and T denotes transposition.

2.4.4 Dimensionality reduction

Dimensionality reduction is a strategy aimed at ensuring economical representation of data as well as better semantic representation. In LSA, the occurrence matrix (term-document matrix) is very sparse and large to the effect that it would demand more storage space in computer memory. Singular Value Decomposition (SVD) is used to address this problem of sparsity as well as to unearth the latent semantic structures from the matrix.

2.4.4.1 Singular value decomposition (SVD)

Singular value decomposition is a technique closely related to eigenvector decomposition and factor analysis (more properly the mathematical generalization of which factor analysis is a special case). It is an important factorization of a rectangular real or complex matrix as used in linear algebra with several applications now in signal processing and statistics. The underlying theorem in SVD is that, *for an m-by-n matrix A of rank r there exists a factorization (Singular Value Decomposition) of the form:*



Where:

- U is a matrix of dimensions *m* by *m* whose columns are orthogonal eigenvectors of AA^T matrix
- S is a matrix of dimensions *m* by *n* whose diagonal values are singular values of matrix A and with nonnegative numbers on the diagonal
- V^T is a transpose of V where V is a matrix of dimensions *n* by *n* whose columns are eigenvectors of A^TA matrix.

Example

Given Matrix A below:

1.000	0.000	0.000	0.000	2.000
0.000	0.000	3.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000
0.000	4.000	0.000	0.000	0.000

A singular value decomposition of this matrix is given by decomposing the matrix into the following 3 matrices.

U:

0.000	0.000	1.000	0.000
0.000	1.000	0.000	0.000
0.000	0.000	0.000	-1.000
1.000	0.000	0.000	0.000

S:

4.000	0.000	0.000	0.000	0.000
0.000	3.000	0.000	0.000	0.000
0.000	0.000	2.236	0.000	0.000
0.000	0.000	0.000	0.000	0.000

V^T:

0.000	1.000	0.000	0.000	0.000
0.000	0.000	1.000	0.000	0.000
0.447	0.000	0.000	0.000	0.894
0.000	0.000	0.000	1.000	0.000
-0.894	0.000	0.000	0.000	0.447

Dimensionality reduction then involves truncating the three matrices obtained by full SVD. Basically the first *k* columns of *U*, the first k rows of V^T and the first *k* rows and *k* columns of *S* are retained. The underlying principle is to have an *M x N* matrix A_k of rank at most *k,* so as to minimize the Frobenius norm of the matrix difference $X=A - A_k$ which measures the discrepancy between matrix A_k and matrix *A* and defined as:

$$\left\| X \right\|_F = \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} X_{ij}^2}$$

The goal in this case is to find a matrix A_k that minimizes this discrepancy, while constraining A_k to have a rank of at most *k.* This process of finding *k* such that the matrix A_k has a rank lower than that of the original matrix A is called low-rank approximation. There has been no agreed general strategy so far for deciding the optimal *k* to use for these retained dimensions; it is rather an empirical issue and depends on methods used for the evaluation of the retrieval results. If *k* is too large we may have more noise in the vector space while too low *k* may lead to factors loosing important information. The reduction process can be demonstrated as below:

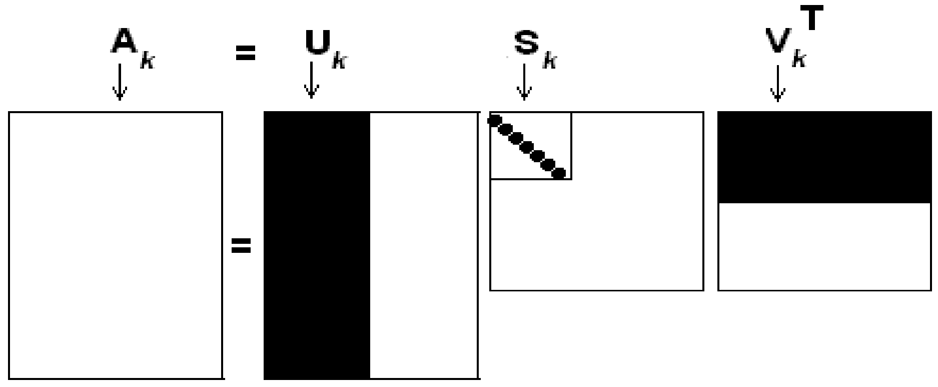


Figure 1. Illustrating dimensionality reduction

Dimensionality reduction will yield a new representation for both terms and documents in the collection. While the vector space model discussed above is able to treat queries and documents uniformly, dimensionality reduction reveals more semantic structures that would not be revealed by vector space model alone. It helps in revealing more relationships between terms and documents which helps in higher performance compared to vector space model.

2.4.5 Document retrieval

In the reduced *k*-space, terms which occur in similar documents are mapped close to each other even though they may not co-occur in the same document. Query terms are also mapped in the same *k-space* where similarity metrics can be used to measure distances between query terms and documents. It is on this comparison of documents and queries on the same *k-space* that LSA is used for information retrieval.

2.4.5.1 Query Mapping on k-vector space

At this point we need a query mapping mechanism on the *k-vector space* and a scoring strategy to rank the documents. The idea here is to transform the query into a vector representation based on the number of dimensions used (*k*) in the semantic space to allow for comparison between the documents and the queries. A query vector $\vec{q}$ is represented in the *k-vector space* by the transformation below:

$$\vec{q}_k = S^{-1}{}_k U^T{}_k \vec{q}$$

2.4.5.2 Similarity Metrics

Once a query has been mapped into the reduced vector space (*k-vector space*) it can be compared against the documents in the vector space using a similarity metric. In the vector space model, to calculate the distance from a document *d* to any query *q* we use *angle cosine*. This method can also calculate the distance between any two documents or terms. Measuring the distance from a document to a query involves the following:

Assume we denote by $\vec{V}(d)$ the vector representing document *d,* with every dictionary (all key words used in the generation of term document matrix) term represented. Suppose we also denote a query by vector $\vec{V}(q)$. Then the angle cosine which is used to show the score of query *q* on document *d* is calculated as shown on the next page.

$$\text{score}(q, d) = \frac{\vec{V}(q).\,\vec{V}(d)}{\left|\vec{V}(q)\right|\left|\vec{V}(d)\right|}$$

From the above equation the numerator represents the dot product of the two vectors $\vec{V}(q)$ and $\vec{V}(d)$ which would sufficiently be enough to measure the score if document *d* and query *q* were of the same length. However, since this is not the case, the denominator is introduced as a way of length normalization. This is called Euclidean normalization, since it involves the calculation of Euclidean lengths for each vector being considered. To rank the other documents based on the query the score is calculated as above for each document and based on the result set, the document that score highest is ranked first. It can be illustrated as shown below.

Document vector	score (q, d)	rank
$\vec{v}(d_1)$	0.8882	1
$\vec{v}(d_2)$	0.3338	3
$\vec{v}(d_3)$	0.5983	2

Table 2. An example of three documents, $\vec{v}(d_1)$, $\vec{v}(d_2)$ and $\vec{v}(d_3)$ ranked on their angle cosine.

3. Experiments

3.1 State-of-the-art analysis

Efforts to find any LSA-based software tool to carry out extensive LSA experiments in this project were fruitless. This could be attributed to the fact that LSA remains an expensive technique especially due to the SVD computation inherent in it. According to [3], there have been no successful experiments with over one million documents done so far and this may explain why its performance has not greatly convinced users to migrate from the lexical searching techniques which have been extensively implemented. LSA has therefore continued to exist in research areas and though there are many claims about its superiority in performance compared to lexical searching techniques, practical applications in real environments are yet to be seen. However there are still some projects either going on or already done based on LSA concept.

The semantic Indexing Project

It is an on-going project to have software solutions based on LSA concept. The goal of this project [11] is to find patterns in unstructured data (documents without descriptors such as keywords or special tags) and to use these patterns to offer more effective search and categorization services. At the moment, there is an incomplete downloadable software tool kit which is still on development stage.

SenseClusters

This is an open source program that allows a user to cluster similar contexts together using unsupervised knowledge-lean methods. Unsupervised knowledge-lean methods [15] rely strictly on the knowledge that is automatically identifiable within the text being processed which avoids dependence on external knowledge sources. The program implements this by latent semantic analysis and native SenseClusters techniques. Native SenseClusters techniques [16] uses a freely available word sense discrimination system that takes a purely unsupervised clustering approach to cluster instances of a given target word based only on their mutual contextual similarities. More technical details regarding SenseClusters project can be found under <http://senseclusters.sourceforge.net/>.

Text to Matrix Generator

This is a MATLAB Toolbox which was developed at the department of Computer Engineering and Informatics, University of Patras, Greece for the purposes of data mining and information retrieval based on LSA concept. The technical details are found under <*scgroup.hpclab.ceid.upatras.gr/scgroup/Projects/TMG/*>.

3.2 Experiment 1: using Text to Matrix Generator (TMG)

3.2.1 Experiment Overview

Text to Matrix Generator (TMG), a MATLAB Toolbox, was used to demonstrate the technical steps involved in the construction of the term-document matrix on which singular value decomposition is applied for dimensionality reduction to reveal existing semantic structures within the document collection. TMG uses the following steps in the construction of the term document matrix:

- Removal of stop words
- Stemming based on the Porter Stemming Algorithm
- Removal of short and long terms based on prior specifications
- Removal of frequent / infrequent terms
- Term weighting and normalization

3.2.2 Inputs

In the experiments conducted a collection of 10 documents has been used. The contents of these documents are as follows:

a) *Programming is the craft of transforming requirements into something that a computer can execute*
b) *The invention of the Von Neumann architecture allowed computer programs to be stored in computer memory*
c) *Debugging is often done with IDEs like Visual Studio, NetBeans, and Eclipse.*
d) *Computer programmers are those who write computer software*
e) *MATLAB allows computation of intensive tasks faster than traditional programming languages.*
f) *Cancer is a class of diseases in which a group of cells display uncontrolled growth*
g) *Immunity refers to resistance of an organism to infection, disease, or other unwanted biological invasion*
h) *The brain is the center of the nervous system in all vertebrate*
i) *The human brain contains roughly 100 billion neurons*

j) *The brains of vertebrates are made of very soft tissue with a texture that has been compared to Jello*

3.2.3 Settings

The settings were made as follows:
Min length(1), Max Length(30), Min Local Frequency (1), Max Local Frequency (inf) ,Min Global Frequency(1), Max Global Frequency (inf), local term weighting (term frequency), Global term weighting (Inverse document Frequency), normalization (checked) and stemming turned on.

3.2.4 Results

The ten documents were transformed into a term document matrix A, with the dimensions 58 by 10. In constructing the Matrix 31 stop words were removed and 5 terms were eliminated using the stemming algorithm. The dictionary and the term-document matrix produced by TMG are shown below:

allow, architectur, billion, biolog, brain, cancer, cell, center, class, compare, comput, craft, debug, diseas, display, eclips, execute,faster, group, growth, human, id, immune, infect, intens, invas, invent, jello, language, MATLAB, memori, nervous, netbean, neumann, neuron, organ, program, programm, refer, requir, resist, roughli, soft , softwar, store, studio, system, task, textur , tissu, tradit, transform, uncontrol, unwant , vertebr, visual, von, write

Table 3. Dictionary produced by TMG

```
(11,1) 0.1890, (12,1) 0.4750, (17,1) 0.4750, (37,1) 0.2484, (40,1) 0.4750, (52,1) 0.4750, (1,2)   0.3556,
(2,2)   0.3556, (11,2) 0.2830, (31,2) 0.3556, (37,2) 0.1860, (45,2) 0.3556, (57,2) 0.3556, (13,3) 0.4082,
(16,3) 0.4082, (22,3) 0.4082, (33,3) 0.4082, (46,3) 0.4082, (56,3) 0.4082, (11,4) 0.4175, (38,4) 0.5246,
(44,4) 0.5246, (58,4) 0.5246, (11,5) 0.1569, (18,5) 0.3943, (25,5) 0.3943, (29,5) 0.3943, (30,5) 0.3943,
(37,5) 0.2062, (48,5) 0.3943, (51,5) 0.3943,  (6,6)  0.3654,  (7,6)  0.3654,   (9,6) 0.3654, (14,6) 0.2554,
(15,6) 0.3654, (19,6) 0.3654, (20,6) 0.3654, (53,6) 0.3654,   (4,7) 0.3432, (14,7) 0.2399, (23,7) 0.3432,
(24,7) 0.3432, (26,7) 0.3432, (36,7) 0.3432, (39,7) 0.3432, (41,7) 0.3432, (54,7) 0.3432,  (5,8)  0.2696,
(8,8)   0.5156, (32,8) 0.5156, (47,8) 0.5156, (55,8) 0.3604,  (3,9)  0.4837,   (5,9) 0.2529, (21,9) 0.4837,
(35,9) 0.4837, (42,9) 0.4837, (5,10) 0.2178, (10,10) 0.4166, (28,10) 0.4166, (43,10) 0.4166,
(49,10) 0.4166, (50,10) 0.4166, (55,10) 0.2912
```

Table 4. The term-document matrix produced by Text to Matrix Generator showing the non-zero entries, with each entry (i,j) corresponding to the weight of term i in document j. All other entries are zero.

3.3 Experiment 2: Comparing term-matching and LSA performance

3.3.1 Experiment Overview

JDesktopSearch has been used as an example of a software application that employs the term-matching technique. It is a Java implementation of a desktop search engine based on Apache Lucene. It can be used to [12] index Html-documents, XML-documents, plain text files, PDF documents, plain text files and open office files. For LSA approach software code written and running on MATLAB environment was used. The software code used is shown at the end of this report. The basic idea was to investigate which technique performed better based on the capability to retrieve the highest number of relevant documents from a document collection.

3.3.2 Data

The experiment involved preparation of 100 short text documents (consisting of between one to three lines of text). The 100 documents were drawn from the following broad topics with 10 documents from each topic: *sociology, psychology, medicine, mathematics, politics, computer science, sports, economics, business* and *religion.* To prepare the dictionary (index terms) to use in LSA, common terms (terms with less discrimination power over queries) were listed in a separate text file which was used to eliminate these terms from the documents. After the elimination of stop words a term-document matrix with dimensions *695 by 100* was constructed based on raw term frequency. To deal with varying lengths of the documents used (one to three lines of text), normalization was done. Dimensionality reduction was done by empirically choosing *k* (the number of factors) to be 10. The selection of *k* as *10* is explained later in this report.

A number of terms likely to be associated with each broad topic were then subjectively selected and used to retrieve documents using LSA approach on MATLAB and then to compare with term-matching retrieval using the same terms. The following terms were used for each broad topic considered:

- Psychology: *psychology, cognition*
- Sociology: *sociology, culture*
- Mathematics: *mathematics, calculus*
- Politics: *politics, government*
- Religion: *religion, supernatural*
- Economics: *inflation, economics*
- Sports: athletics, sport
- Medicine: *medicine, diseases*
- Computer: *computer, program*
- Business: *business, customer*

3.3.3 Results

Using LSA approach on MATLAB, all the documents scoring positively on each of the query terms were retrieved. For evaluation, only the first 10 documents with highest score based on each of the query terms were considered with the relevant documents from these 10 being subjectively recorded. In the term-matching technique all retrieved documents were considered and from these the relevant ones were subjectively recorded. The results are summarized in the table shown on the next page.

Term	*Total number of relevant documents in the collection*	*Number of documents retrieved using term-matching*		*Number of documents retrieved by LSA approach*	
		Total	*Relevant*	*Total*	*Relevant*
Psychology	10	9	9	10	9
Sociology	10	8	8	10	8
Mathematics	10	7	7	10	8
Politics	10	2	2	10	4
Religion	10	3	3	10	7
Economics	10	3	1	10	6
Sport	10	2	2	10	8
Medicine	10	3	3	10	8
Computer	10	3	3	10	8
Business	10	10	8	10	9
Cognition	4	3	2	10	4
Culture	2	2	2	10	2
Calculus	2	2	2	10	2
Government	10	10	6	10	6
supernatural	1	1	1	10	1
Inflation	3	2	1	10	3
Athletics	3	1	1	10	3
Disease	5	7	3	10	5
Program	5	3	2	10	5
Customer	2	2	1	10	1

Table 5. Comparison of LSA approach and Term-Matching approach

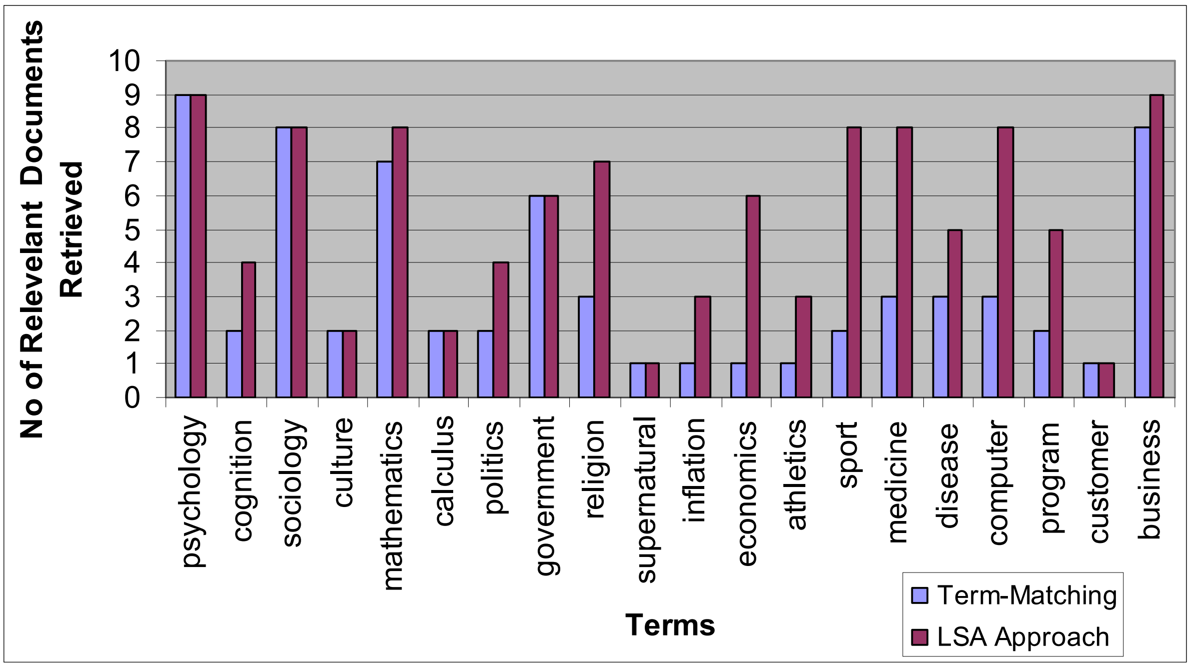


Figure 2. Visualizing experiment 2 results

3.4 Experiment 4: Investigating Synonymy and Polysemy

3.4.1 Experiment Overview

A further experiment was done on LSA approach to investigate how it treats the polysemy and synonymy problem in comparison with the term-matching approach. A number of terms with almost similar meanings were selected for synonymy problem whereas for polysemy problem terms with more than one meaning in different contexts were selected.

3.4.2 Synonymy problem

For synonymy problem the following terms were used:

- *soccer* and *football*
- *funds, finance, money* and *income*
- *traits* and *behaviour*

3.4.3 Results

While term-matching approach retrieved only documents with the exact terms as typed, LSA approach was able to retrieve documents that had terms closely related with query terms. For instance the term *soccer* retrieved only one document (document with the term *soccer*) using term-matching technique whereas in LSA approach two documents (one with the term *soccer* and another with the term *football*) were retrieved. The table below gives a summary of the results.

Term	*Documents retrieved using Term-Matching technique*	*Documents retrieved using LSA Approach*
Soccer	1 document with term 'soccer'	2 documents with either 'soccer' or 'football'
Funds	1 document with term 'funds'	4 documents with either 'funds', 'finance', 'money' or 'income'
Traits	1 document with term 'traits'	4 documents with either 'traits' or 'behaviour'

Table 6. *Demonstrating synonymy problem*

3.4.4 Polysemy problem

For polysemy problem the following terms were used both in LSA approach and in the term-matching approach:

- *press*
- *run*
- *Value*

3.4.2 Results

The term, *press* that can be used to mean, *to exert steady weight or force against* in the context of sports or *to try to influence by insistent arguments* in the context of politics led to only one document about *sports* being retrieved using the term-matching approach. In LSA approach, three documents on *politics* were retrieved with none from *sports*. Likewise, the term, *run,* which can be used in the context of *athletics* or in the case of *contesting for political office* led to retrieval of only one document about *politics* using term-matching approach whereas in LSA approach four documents on *sports* and three on *politics* were retrieved. The term *value* which can be used in economics to express *a fair price or return* and in sociology to express *quality considered worthwhile* led to the retrieval of one document about *Economics* and one document on *Sociology* using term-matching (both documents had the term *value*) whereas LSA approach retrieved only the one document on *Sociology*.

3.5 Discussion

According to the experiments done above and other experiments that have been conducted by other researchers, it can be shown that LSA presents a number of advantages that makes it more superior compared to traditional term matching techniques. The technique still has some limitations that will also be discussed later.

3.5.1 Advantages of LSA

Better performance

In our second experiment whose main focus was to find out which of the techniques could retrieve the highest number of relevant documents, LSA approach was found to be more superior in terms of retrieving the highest number of relevant documents compared to simple term matching. In general the LSA MATLAB application used retrieved on average more than 42 documents (high recall) though only the first 10 documents were considered in each case. It was difficult to measure the exact precision and recall values due to the limitation of the tools used and the size of the document collection which would give results that are not comprehensive. Better results on performance could have been shown also if our LSA MATLAB experiment allowed for searching of more than one dictionary terms using a single query and if the document corpus was large enough with many topics. Based on other experiments that have been done on MED (a collection of medical abstracts) and CISI (a set of 1460 information science abstracts) among other datasets as described in [6], LSA performance has generally been found to be superior compared to simple term matching in terms of recall and precision performance. Using MED, for instance, LSA was shown to present a 13% improvement on precision over raw term matching. LSA superiority is traced to its ability to correctly match queries with relevant documents based on topical meaning even if queries and documents use different terms. The preprocessing step inherent in LSA also improves LSA performance since the overall distribution of a word over its usage contexts, independent of its correlations with other words, is considered.

Synonymy

LSA unlike traditional term matching methods is able to deal with synonymy problem to some extent. Individual terms [6] are replaced as the descriptors of documents by independent “artificial concepts” that can be specified by any one of several terms (or documents) or combinations thereof. This allows retrieval of relevant documents that do not contain the terms of the query, or whose contained terms are qualified by other terms in the query or document but not both. In LSA, terms that occur in context of similar meanings even though they may never occur in the same documents are located close to each other in the reduced dimension space. This makes it possible to retrieve documents in the neighbourhood (the cosine defining a neighbour is relative based on dimensions used and set of data used) by use of any of these terms. In our fourth experiment, it was found that some terms in the query led to the retrieval of not only documents containing these terms but also documents containing other similar or related terms e.g. for the term *soccer* documents with the term *football* were also retrieved. This implies that the term *soccer* and *football* are located close to each other in the semantic space and therefore will retrieve similar documents. This relationship between *soccer* and *football* comes from the transitive relation [14]: *soccer* co-occurs with *football* and *football* co-occurs with *human*. Stemming done in the preprocessing stage of LSA also helps in handling synonymy problem since it assists in capturing the likely synonyms. In the illustration below the queries q1 and q2 contain related terms and therefore will retrieve similar documents which otherwise would not be possible with term matching techniques.

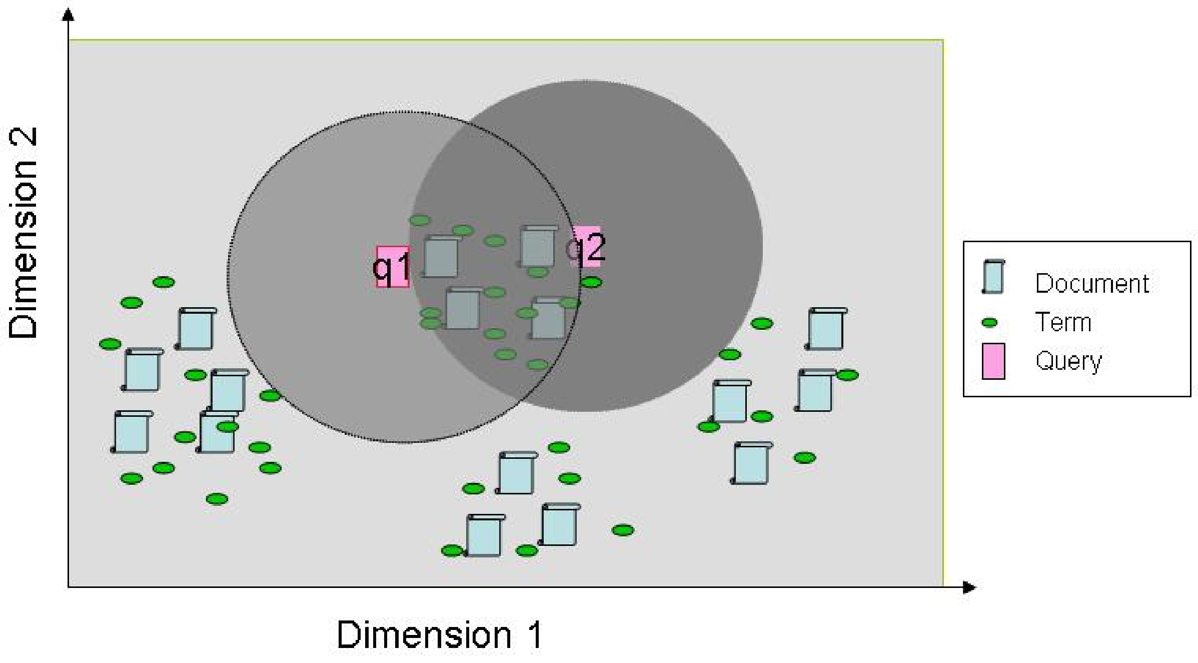


Figure 3. Illustration of synonymy handling by LSA using queries q1 and q2. The grey circles show documents likely to be retrieved by each query. Due to the intersection, 4 documents will be retrieved by either of the queries.

Storage space minimization

Comparing LSA approach with normal vector space model for documents, LSA presents better storage utilization. The removal of common terms (terms with less discriminatory power), the stemming process shown in the first experiment and the dimensionality reduction inherent in the LSA approach reduces greatly the storage requirements for a large document collection. In our first experiment, it was found that the term document matrix A contains zeros in many entries (approximately 89%).In vector space model this matrix would be stored with all these zeros. This problem is better handled by dimensionality reduction in LSA which also eliminates the noise in the semantic space and which makes LSA outperform vector space models in document retrieval.

LSA mimic human knowledge

The similarity measures based on LSA have been shown to closely mimic human judgements. In [9] an LSA model was used to simulate human performance using TOEFL (Test of English as a Foreign Language) and comparing with average test-takers. LSA was found to score 65% correct which was identical to the average score of a large sample of students applying for college entrance in the United States of America from non-English speaking countries.

3.5.2 Limitations of the LSA Approach

Updating

One problem faced in LSA approach is the updating of the term-document matrix when new documents arrive. Updating can involve recomputing the SVD whenever new documents arrive, but due to time constraints and computing resources (memory and processor speed) required, SVD recomputation becomes very expensive. An approach of folding in new documents and queries where new documents are located at the centroids of their terms and new terms are located at the centroids of the documents in which they appear, has been suggested in [6] but it still remains unknown for how many times this can be done without having to redo full SVD. This is because the addition of new documents through this “folding in” approach fails to capture co-occurrences in the newly added documents and also ignores new terms that may be contained in these documents leading to degradation of the LSA quality in representation.

Determination of K (number of factors) in dimensionality reduction

For low-rank approximations, *k* dimensions have to be selected for the representation of terms, documents and queries. In the experiments done, *k* was empirically selected to be 10 after a number of tests that were based on finding the optimum documents representation. In the tests, the search of the term *medicine* with *k* values set as 50, 40, 30, 20, 10 and 5 retrieved 4, 6, 6, 8, 8 and 7 relevant documents respectively, thereby making 10 to be a suitable value for *k*. The selection of *k* as 10 was also motivated by the expected topic clusters in the reduced semantic space based on the 10 topics from which documents were extracted. Therefore, there is still no specific technique or algorithm for selecting *k,* the number of dimensions to retain and therefore it remains an empirical issue. If *k* is too large we may have more noise in the vector space while too low *k* may lead to factors loosing important information.

Polysemy

In LSA, any term is mapped as a point in the latent semantic space. For a term with several meanings, the mapping is done based on the weighted average of the different meanings. This implies that a serious distortion may occur where none of the real meanings is like the average meaning. This may lead either to the retrieval of high number of documents having different topical meanings, some relevant and some irrelevant to the user (in case clusters affected are close to each other) or even the retrieval of very few or no documents (in case clusters affected are far from each other). In experiment 4, the search for the term *press* retrieved some documents on *politics* and some on *sports.* Given these results, it is difficult to tell automatically which topic satisfies the user expected meaning and therefore retrieve only the very relevant documents based on that meaning. The illustration of polysemy problem is shown below; where the query term (q) is located almost between two clusters (topics) therefore retrieving differing documents in terms of their topics.

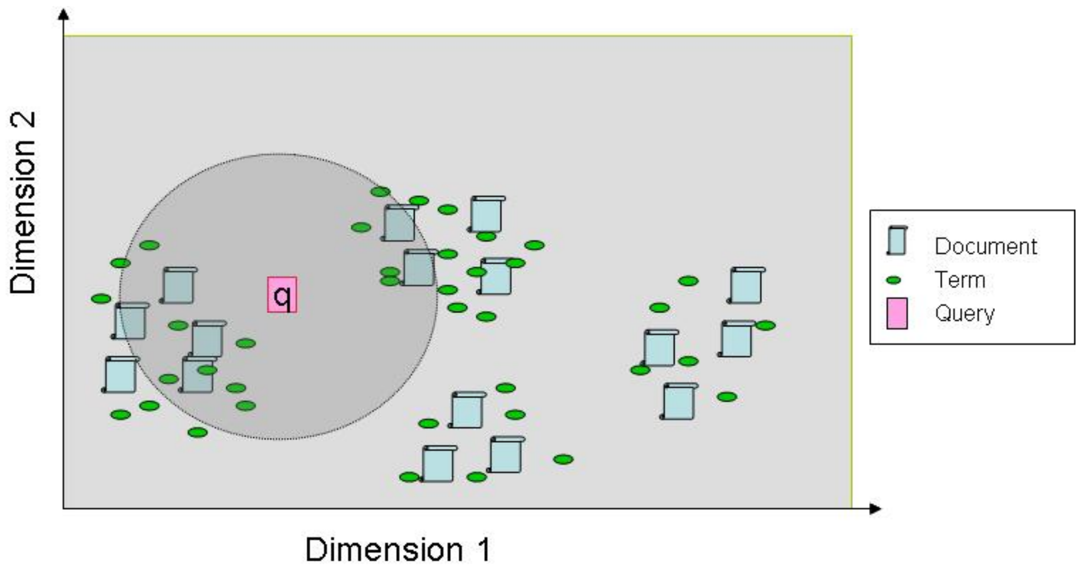


Figure 4. Illustration of polysemy handling by LSA

The solution to the polysemy problem would be to realize that a term has several distinct meanings and to subcategorize it and place it in several points in the latent semantic space. However, but it has not been achieved in this project.

Word order

LSA's "bag of words" nature does not make use of word order which may lead to the ignorance of syntactical, logical and non-linguistic pragmatic entailments. LSA therefore skips the meanings produced by the order of words to focus only on how differences in word choice and differences in passage meanings are related. This can result in incomplete representation and thus retrieval errors. For instance, the two documents below would be treated equally:

The old scenery in the small city

The old small city in the scenery

This problem can be handled to some extent by making use of biword and phrase indexes. This involves including of biwords (2 terms making single meaning that would not be realized if they were treated individually) and phrases (several terms making a single meaning that would not be realized if they were treated individually) in the term-document matrix. To achieve this all biwords and phrases occurring in the documents must be known before the process begins. This process is rather expensive to realize.

4. Conclusion and future work

4.1 Conclusion

In this project the application of the latent semantic technique has been investigated based on performance and compared with term-matching technique. From the project we conclude that LSA out-performs term-matching techniques by retrieving a higher number of relevant documents from a document collection. The state-of-the-art analysis was also done to find out suitable software tools implementing LSA in an actual scenario involving document retrieval. The conclusion derived from the analysis is that though many experiments have been done that shows higher performance for LSA, software tools that can be bought off the shelf do not yet exist. There are some software firms claiming to employ LSA in their products but this cannot be confirmed since detailed technical information is missing.

4.2 Future work

Having investigated the application of LSA in information retrieval the following limitations still need to be pursued in future work.

Polysemy solution

Despite the fact that polysemy and synonymy problems in keyword matching motivated the development of LSA, the problem of polysemy has not yet been solved. Attempts must be made in the future to map terms that have different meanings in their contexts as distinct points on the latent semantic space based on their meanings.

Determination of number of factors

A possible solution is still needed to address the problem of selecting *k* dimensions to retain in the reduced dimension space. Currently, it is done empirically depending on the methods used for the evaluation of the retrieval results.

Word Order

A solution to treat the word order in LSA should be further researched. This will help in having the LSA gain substantial cognitive abilities that human beings use to construct and apply knowledge from experience, in particular the ability to use detailed and complex order information such as that expressed by syntax and used in logic.

Probabilistic Latent Semantic Analysis (PLSA)

PLSA uses a generative latent class model to perform probabilistic mixture decomposition. It has been argued that this approach results in a [8] more principled approach with a solid foundation in statistical inference. On the other hand [10], the methodological foundation of LSA remains to a large extent unsatisfactory and incomplete. Research should focus more in comparing the two approaches, LSA and PLSA to come-up with better performing tools for information retrieval.

Tools used

Text To Matrix Generator (TMG)

TMG [5] is a MATLAB Toolbox for Data Mining and Information Retrieval (IR) tasks. The diagram below shows the structure and the modules of this tool. It has five modules namely: indexing, dimensionality reduction, clustering, classification and retrieval as shown in the diagram below.

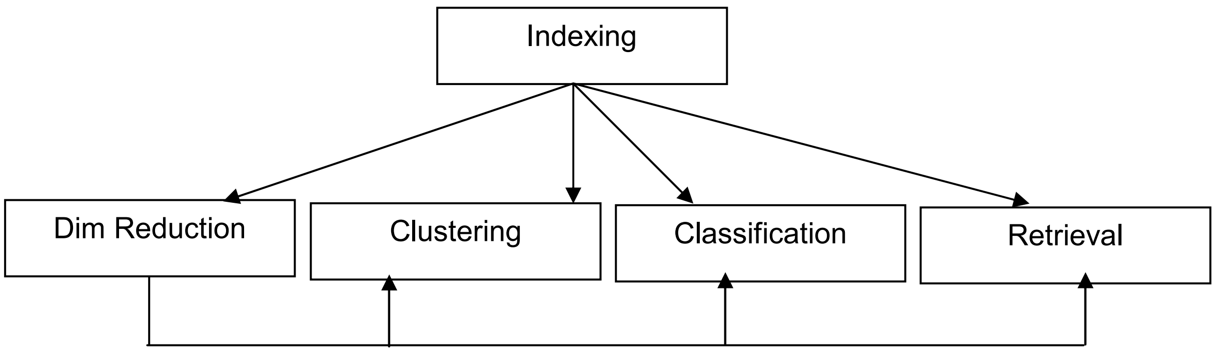


Figure 5. Structure and dependencies of GUI modules of TMG

JDesktopSearch

JdesktopSearch is a Java implementation of a desktop search engine based on Apache Lucene. It can be used to index Html-documents, XML-documents, plain text files, PDF documents, plain text files and open office files. It uses term matching technique to retrieve documents.

MATLAB

MATLAB (registered trademark of The MathWorks, Inc) was used to implement LSA. It is [7] an interactive system whose basic data element is an array that does not require dimensioning. The codes shown in the next page were used:

```
%% Input documents
n = 0;
D = cell(0);
while 1
  filename = ['documents/'  int2str(n + 1) '.txt'];
  fid = fopen(filename, 'r');%...open file for reading
  if (fid == -1), break, end
  n = n + 1;
  D{n} = (fread(fid, '*char'))';
  fclose(fid);
end

%% Simple document preprocessing
for j = 1:n
  D{j} = lower(D{j});  %...converts string read from files into lower case
end

%% Removal of stop words
C = textread('stop words/stopwords.txt', '%s'); %...Reading the stop lists
i=0; %...Keeps track of documents
P=0;
while (i<n)
  docname=['documents/'  int2str(i + 1) '.txt'];
  W= lower(textread(docname, '%s'));
  P=cat(1,P,setdiff(W,C));
 i=i+1;
end
P(1) = [];
Dict=unique(P);
i=length(Dict);
save 'dictionary/dictionary.mat' Dict;

%% Input terms
m =length(Dict);
%%Dictionary Statistics
fprintf(1, '%s \n','DICTIONARY TERMS');
for j=1:m
fprintf(1,'%d%s%s\n ', j, ' : ', Dict{j});
end
```

```
 %% Occurrence matrix
A = zeros(m, n);
for i = 1:m
 for j = 1:n
   A(i, j) = size(findstr(D{j}, Dict{i}), 2); %...Construction of Term-Document Matrix based on raw term
frequency
 end
end

 %% Normalization process
for j = 1:n
  A(:, j) = A(:, j) / norm(A(:, j));
end

%% Dimension reduction
[U,S,V] = svd(A);
Vt=V';
k =10 ;
Sk = S .* [ones(m, k) zeros(m, n-k)]; % retain k singular values
Ak = U * Sk * V';

%saving results
save 'matrices/A.mat' A;
save 'matrices/U.mat' U;
save 'matrices/S.mat' S;
save 'matrices/Vtranspose.mat' Vt;
save 'matrices/Ak.mat' Ak;
```

```
%% Document retrieval
%To search assign the term index to search_request var
search_request =492;
q = zeros(m, 1);
q(search_request) = 1;
r = (q'*Ak)';
[sr, id] = sort(r);
 fprintf(1, '\n                                        search request: "%s"\n\n', Dict{search_request});


 fprintf(1, '%s %s \n', 'Score',' Documents');


 fprintf(1, '%s  \n','__________________________________________________________');


 for j = n:-1:1
     if (sr(j)>0)
       fprintf(1, '%5.6f  %s\n', sr(j), D{id(j)})
     end
  end
```

References

[1] Rosario Barbara, *Latent Semantic Indexing: An overview*, INFOSYS 240 Spring 2000 Final Paper, 2000

[2] Bellegarda R. Jerome, *Latent Semantic Mapping: Principles & Applications*, Morgan & Claypool, 2007

[3] Manning D. Christopher, Raghavan Prabhakar and Schütze Hinrich, *Introduction to Information Retrieval*, Cambridge University Press, New York, 2008.

[4] Yang Kichoon, *Latent semantic indexing and linear relevance feedback in text information retrieval theory*, 1999.

[5] Zeimpekis Dimitrios, Gallopoulos Efstratios, Text To Matrix Generator-User's Guide, 2007.

[6] Deerwester, S., Dumais, S. T., Furnas, G., Landauer, T., & Harshman, R. *Indexing by latent semantic analysis*. Journal of American Society for Information Science, 391–407, 1990.

[7] The Math Works, Inc. *MATLAB Documentation, 2007*

[8] Hofmann, T. *Probabilistic latent semantic analysis*. In Proceedings of 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 50-57, 1999.

[9] Landauer, T.K., Foltz, P.W., & Laham, D. Introduction to latent Semantic Analysis. Discourse Processes, 25, 259-284, 1998.

[10] Hofmann Thomas. *Unsupervised Learning by Probabilistic Latent Semantic Analysis*, Machine Learning, 42, 177-196, 2001.

[11] The semantic Indexing Project. Available from: < http://knowledgesearch.org/> [Accessed 06 February 2009]

[12] JdesktopSearch Online Help. Available from: <http://jdesktopsearch.wiki.sourceforge.net/Online+help?f=print> [Accessed 06 February 2009]

[13] Foltz P. W. *Using Semantic Indexing for Information Filtering*, Proceedings of the Conference on Office Information Systems, Cambridge, MA, 40-47.

[14] Kontostathis April, Pottenger M. William. *A Framework for Understanding Latent Semantic Indexing (LSI) Performance*, Elsevier Science, 2004

[15] Pedersen Ted and Bruce Rebecca. *Knowledge Lean Word-Sense Disambiguation*, Proceedings of the Fifteenth National Conference on Artificial Intelligence, Madison, WI, 1998.

[16] Pedersen Ted and Purandare Amruta. *SenseClusters – Finding Clusters that Represent Word Senses.* Available from: < http://www.aclweb.org/anthology-new/N/N04/N04-3008.pdf> [Accessed 23rd February 2009]