

**STUDY OF DOCUMENT RETRIEVAL USING LATENT SEMANTIC
INDEXING (LSI) ON A VERY LARGE DATA SET**

by

A N K Zaman

B.Sc. Computer Science and Technology, University of Rajshahi, Bangladesh
M.Sc. Computer Science and Technology, University of Rajshahi, Bangladesh

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
MATHEMATICAL, COMPUTER, AND PHYSICAL SCIENCES
(COMPUTER SCIENCE)

THE UNIVERSITY OF NORTHERN BRITISH COLUMBIA

January 2010

© A N K Zaman, 2010



Library and Archives
Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file *Votre référence*
ISBN: 978-0-494-60849-4
Our file *Notre référence*
ISBN: 978-0-494-60849-4

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.


Canada

Abstract

The primary purpose of an information retrieval system is to retrieve all the relevant documents, which are relevant to the user query. The Latent Semantic Indexing (LSI) based ad hoc document retrieval task investigates the performance of retrieval systems that search a static set of documents using new questions/queries. Performance of LSI has been tested for several smaller datasets (e.g., MED, CISI abstracts etc) however, LSI has not been tested for a large dataset. In this research, we concentrated on the performance of LSI on large dataset. Stop word list and term weighting schemes are two key parameters in the area of information retrieval. We investigated the performance of LSI by using three different set of stop word lists and, also, without removing the stop words from the test collection. We also applied three different term-weighting (raw term frequency, log-entropy, and tf-idf) schemes to measure retrieval performance of LSI. We observed that, firstly, for a LSI based ad hoc information retrieval system, a tailored stop word list must be assembled for every unique large dataset. Secondly, the use of tf-idf term weighting scheme shows better retrieval performance than log-entropy and raw term frequency weighting schemes even when the test collection became large.

Dedicated
to my parents,
my wife Majida,
and my beloved son Rafan

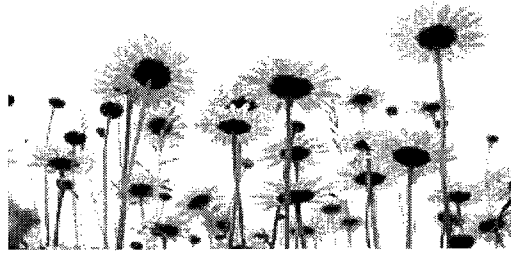


TABLE OF CONTENTS

ABSTRACT	II
LIST OF TABLES.....	VI
LIST OF FIGURES.....	VII
LIST OF FIGURES.....	VII
ACKNOWLEDGEMENT	VIII
ACKNOWLEDGEMENT	VIII
CHAPTER-1	1
INTRODUCTION	1
1.1 BACKGROUND	1
1.2 BASIC TERMINOLOGY	1
1.2.1 TREC.....	1
1.2.2 Term Weighting Scheme.....	2
1.2.3 Stop Words.....	2
1.2.4 Stemming.....	2
1.3 PROBLEM STATEMENT.....	2
1.4 THESIS OUTLINE.....	3
CHAPTER-2	4
THEORIES OF INFORMATION RETRIEVAL TECHNIQUES.....	4
2.1 INTRODUCTION	4
2.2 BACKGROUND AND RELATED THEORY.....	4
2.3 INFORMATION RETRIEVAL VS DATA RETRIEVAL	4
2.4 INFORMATION RETRIEVAL SYSTEMS	5
2.5 INFORMATION RETRIEVAL APPROACHES.....	6
2.5.1 Classical Boolean Approach.....	7
2.5.2 Extended Boolean Approach.....	7
2.5.3 Vector Space Approach.....	8
2.5.4 Latent Semantic Indexing.....	14
2.6 TREC AND RELEVANCE JUDGMENT	20
2.6.1 Text REtrieval Conference-8 (TREC-8) Relevance Judgment	21
2.6.2 Relevance Judgement Problems.....	21
2.7 EVALUATION OF INFORMATION RETRIEVAL PERFORMANCE	22
2.7.1 Recall and Precision	23
2.7.2 Recall-Precision Calculations and Plotting.....	23
2.8 PAIRED SAMPLE T-TEST.....	26
CHAPTER-3	28
ISSUES IN LATENT SEMANTIC INDEXING (LSI) RESEARCH.....	28
3.1 INTRODUCTION	28
3.2 RELATED RESEARCH WORK ON LATENT SEMANTIC INDEXING (LSI).....	28
3.2.1 Using Latent Semantic Analysis to Improve Access to Textual Information.....	28
3.2.2 Indexing by Latent Semantic Analysis.....	29
3.2.3 Large-Scale Information Retrieval Using Latent Semantic Indexing	30
3.2.4 Automatic Cross-Language Information Retrieval using Latent Semantic Indexing.....	31
3.2.5 Latent Semantic Indexing-Based Intelligent Information Retrieval System for Digital Libraries	31
3.2.6 Singular Value Decomposition and Rank K Approximation.....	32
3.2.7 Singular Value Decomposition for Text Retrieval	32
3.3 CONCLUSION	34
CHAPTER-4	35
TEST COLLECTION AND PRE-PROCESSING	35
4.1 INTRODUCTION	35

4.2 STANDARD TEST COLLECTION	35
4.2.1 TREC and Standard Test Collection	35
4.2.2 Los Angeles Times (LA Times) Text Collection (TREC-8)	36
4.3 PRE-PROCESSING OF DATA	38
4.3.1 Method to Extract Documents	38
4.3.2 Stop Word Removal and Stemming	39
4.4 STOP WORD REMOVAL AND STEMMING PROGRAM	42
4.5 QUERY SELECTION	44
4.7 CONCLUSION	45
CHAPTER-5	46
EXPERIMENTAL STUDIES ON STOP WORD LISTS AND TERM WEIGHTING SCHEMES....	46
5.1 INTRODUCTION	46
5.2 GTP (GENERAL TEXT PARSER)	46
5.3 EXPERIMENTAL DESIGN	48
5.3.1 Experiment-1: Study of Stop Words	48
5.3.2 Experiment-2: Study of Weighting Schemes	49
5.4 RESULTS AND EVALUATIONS	49
5.4.1 Result: Experiment-1	49
5.4.2 Result: Experiment-2	53
5.5 CONCLUSION	54
CHAPTER-6	56
CONCLUSION AND FUTURE DIRECTIONS	56
6.1 INTRODUCTION	56
6.2 SUMMARY	56
6.3 POSSIBLE EXTENSIONS OF THE EXPERIMENTS	57
BIBLIOGRAPHY	59
APPENDIX A	67
1. STOP WORDS LIST: GENERATED STOP WORDS LIST (UNIVERSITY OF NORTHERN BRITISH COLUMBIA)	67
2. STOP WORDS LIST: UNIVERSITY OF TENNESSEE	74
3. STOP WORDS LIST: UNIVERSITY OF GLASGOW	76
APPENDIX B	79
SOURCE CODE: TREC FILE EXTRACTOR	79
APPENDIX C	96
GENERAL TEXT PARSER [UNIVERSITY OF TENNESSEE][GILES ET AL 2001]	96
QUERY PROCESSING WITH GTPQUERY	98
APPENDIX D	100
T-TEST CALCULATIONS	100
LEVEL OF SIGNIFICANCE FOR TWO-TAILED T-TEST	102
PUBLICATIONS	103

List of Tables

Table 2.1: Information Retrieval Vs Data Retrieval.....	5
Table 2.2: Term-by-document matrix.....	9
Table 2.3: Angles and their Cosine values.....	20
Table 2.4: Recall-precision of a single query	24
Table 2.5: Interpolated precision of a query	24
Table 2.6: Interpolated average precision of two different systems	25
Table 3.1: A set of datasets [Letsche 1996].....	30
Table 4.1: Input Data Characteristics (LA Times Data Set (1989, 1990))	38
Table 5.1: Different Parameters for the Study of Stop Words.....	49
Table 5.2: Different Parameters for the Study of Weighting Schemes.....	49
Table 5.3: 10-point Interpolated Precision of Four Different Systems	50
Table 5.4: 10-point Interpolated Precision, Mean and Standard Deviation of Four Systems	52
Table 5.5: Comparison among Systems in Terms of T-test	53
Table 5.6: 10-point Interpolated Precision of Three Different Systems	53

List of Figures

Figure 2.1: Framework of an Information Retrieval System.....	6
Figure 2.2: Recall-Precision Graph of a query	25
Figure 2.3: Recall-precision graph of two retrieval systems	26
Figure 3.1: LSI vs. Term Matching on MEDLINE [Husbands et al 2001]	33
Figure 3.2: LSI vs. Term Matching on TREC6 [Husbands et al 2001]	33
Figure 4.2: An extract from the LA Times Collection, file name LA123190, and illustrates one document	37
Figure 4.3: TREC File Extractor.....	39
Figure 4.4: An Input File before Pre-processing	41
Figure 4.5: An Input File after Pre-processing	42
Figure 4.6: Pre-processing Software.....	43
Figure 4.7: First Two Queries (401 and 402) from the TREC-8 Query Set.....	45
Figure 5.1: Latent Semantic Indexing (LSI) Retrieval Systems [GILES ET AL 2001]....	47
Figure 5.2: Recall-Precision Graph for Experiment-1	51
Figure 5.3: Recall-Precision Graph for Experiment-2	54

Acknowledgement

I would like to express my deepest gratitude to my supervisors, Dr. Charles Brown and Dr. Liang Chen, for their active support throughout my master's study at UNBC. Their suggestions, comments, and encouragement helped me to complete this thesis. I also wish to thank Dr. Jennifer Hyndman for her encouragement and valuable advice on my thesis. As well, I would acknowledge the external reviewer for his willingness to serve on my thesis committee.

The work carried out in this thesis is financially partly supported by the Research Assistantship (RA) award from the Computer Science Program, UNBC. I am also very much thankful to the RA committee for granting me the award.

I would also like to extend my thanks to Heath de la Giroday who actively helped me in editing my thesis.

I am indebted to my parents and in-laws for their encouragement and support in everything I have done. Finally, I would like to thank my wife Majida Akter for her constant support and encouragement which made it possible for me to pursue and complete this work.

Chapter-1

Introduction

1.1 Background

Information Retrieval (IR) is the discipline that deals with retrieval of unstructured data, especially textual documents, in response to a query or topic statement. The need for effective methods of automated IR has grown in importance because of the tremendous explosion in the amount of unstructured data in digital form.

IR focuses on retrieving documents based on the content of their unstructured components. An IR request (typically called a “query”) may specify desired characteristics of both the structured and unstructured components of the documents to be retrieved, e.g., “The documents should be about ‘Global Warming’ and their author must be ‘Rosemary’.” In this example, the query asks for documents whose body (the unstructured part) is “about” a certain topic and whose author (a structured part) has a specified value.

IR typically seeks to find documents in a given collection that are “about” a given topic or that satisfy a given information need. The topic or information need is expressed by a query, generated by the user. Documents that satisfy the given query in the judgment of the user are said to be “relevant.” Documents that are not about the given topic are said to be “non-relevant.”

1.2 Basic Terminology

Before mentioning the problem statement we present some terminology associated with this thesis:

1.2.1 TREC

TREC (Text REtrieval Conference) is part of U.S. National Institute of Standards and Technology (NIST), and TREC provides the infrastructure (e.g., organizing text retrieval conferences, provide standard dataset etc) for large-scale testing of text retrieval technology.

1.2.2 Term Weighting Scheme

In the text based IR process term weighting schemes help us to assign a weight/value to every term of a document. Tf-idf, and log-entropy are examples of term weighing schemes. Term weighting schemes are discussed in Chapter-2 in details.

1.2.3 Stop Words

Stop words are the list of words (e.g., Articles: a, an, and the, Prepositions: at, by, in, to, from, and with, Conjunctions: and, but, as, and because, others: become, everywhere, and likely) that are not considered influential during the execution of Latent Semantic Analysis (LSA) process. A stop word list is also called a negative-dictionary.

1.2.4 Stemming

Stemming is a technique to reduce a word to its stem or root form. Stemming improves retrieval performance by reducing distinct index terms. For example:

◆ Plurals:

1. The word *cats* is stemmed as *cat* because of the rule $s \rightarrow \emptyset$
2. The word *stresses* is stemmed as *stress* because of the rules $sses \rightarrow ss$ and $ss \rightarrow ss$

◆ Participles:

1. The word *examined* is stemmed as *examin* because of the rule $ed \rightarrow \emptyset$
2. The word *playing* is stemmed as *play* because of the rule $ing \rightarrow \emptyset$

1.3 Problem Statement

We have reviewed the published work on LSI for the last 20 years, and we found some open questions on LSI, stop word lists, term weighting schemes, and large datasets. So, we are interested in exploring the following issues:

- LSI has not been tested on a large standard dataset with the most accepted weighting schemes tf-idf and log-entropy. So, we are interested in the performance of LSI on a large standard dataset.
- TREC encourages text retrieval research, but TREC does not provide any evidence/standard rule to use stop words in IR research. So, we propose an algorithm to produce a stop word list based on TREC-8 LA Times dataset for our experiments.
- Different information retrieval research groups use different stop words lists. Our aim is to find the effect of stop words in the IR process in the case of a large

standard dataset. Also to find the effect of arbitrary (a stop word lists is not tailored for a unique dataset) use of stop word list on LSI-based retrieval system.

- Term weighting schemes improve retrieval performance, however, we did not find any concrete evidence that mentions which weighting scheme gives better performance in case of a large dataset. So, we decided to find the effects of different term weighting schemes (raw term frequency, tf-idf, and log-entropy) on LSI based IR process in case of large standard dataset.

1.4 Thesis Outline

The rest of the thesis is organized as follows:

Chapter-2 focuses on the theories of IR and LSI as well as the evaluation methods of retrieval performance. Chapter-3 is the literature review of the Latent Semantic Indexing (LSI) research. Chapter-4 focuses on the characteristics of the large test collection and associated query set. It also illustrates the different phases of data pre-processing, e.g., stemming. Chapter-5 focuses on the experimental setup, obtained results, and the evaluations. Chapter-6 contains the concluding remarks.

Chapter-2

Theories of Information Retrieval Techniques

2.1 Introduction

With the exponential development of information technologies, a vast amount of intellectual resources has been recorded in computer-readable forms of information that can be digitally transmitted or processed. The evolution of communication networks has made an immense number of datasets available in the public domain that can be used by ordinary users. However, due to the tremendous volume of the data, it has become extremely difficult for the users to find the desired information. This challenge has led to a huge demand for information management technologies which can facilitate access to the information for users. [Zeng 2005] The objective of this thesis to study the effects of different stop word lists, different term weighting schemes on LSI based IR system with a very large dataset.

2.2 Background and Related Theory

Information retrieval encompasses the techniques for retrieving relevant information from storage in response to a user's request. In the following sections, a brief overview of the field of information retrieval is provided in order to introduce the proposed research work. The following sections present a historical review of a number of IR approaches that are relevant to the thesis topic. The evaluation strategy that is used to estimate the retrieval performance of an IR method is also covered.

2.3 Information Retrieval vs Data Retrieval

When the term *information retrieval (IR)* is mentioned in this thesis, it refers to the automatic information retrieval systems that search unstructured databases for data records related to the user's queries, and inform the user regarding the existence/non-existence as well as the location of the document. Although the term *information* has a close association with the term *data*, they are not equivalent concepts. The differences between *information retrieval* and *data retrieval* are illustrated in Table 2.1. [Rijsbergen 1979]

	Information Retrieval	Data Retrieval
Query Language	Natural	Artificial (not natural)
Query Specification	Incomplete	Complete
Matching	Best Match, Partial Matching	Exact match
Items Wanted	Relevance	Exact matching
Model	Probabilistic or Statistical	Deterministic

Table 2.1: Information Retrieval vs Data Retrieval

In IR, the query statement is usually expressed in natural language and does not necessarily need to be structured. On the contrary, data retrieval usually requires the request to comply with specified syntax and to provide a complete description of the desired information. With regard to evaluating the retrieved records, IR judges relevant items to be good matches, and among them the most relevant one as judged by the user, is determined to be the best match. Whereas, data retrieval only regards exact matches as an acceptable match. Due to the differences between information retrieval documents (unstructured records) and data retrieval documents (structured records), information retrieval engines might, but are not guaranteed to, retrieve all the relevant documents from the storage. Even if IR systems manage to retrieve a list of relevant records, such a list might not be complete. In contrast, data retrieval systems are guaranteed to output all occurrences of the records that match.

2.4 Information Retrieval Systems

An information retrieval system is a “device interposed between a potential user of information and the information collection itself”. [Harter 1986] The purpose of an IR system is to capture wanted items and to filter out unwanted items. [Harter 1986] A typical IR system performs representation, storage, and retrieval of unstructured data, and may contain some or all of the following parts/components: indexing, query operation, matching, output module, and feedback module. The indexing usually contains two primary processes. The first process conducts conceptual analysis of information resources in the collection to obtain the contained concepts. These concepts, usually called effective terms, make up a system vocabulary that is applicable to all the information pieces in this system. The output of the first process flows to the second stage, in which a translation mechanism is employed and a database of information representations is created. When an information request is posed, the query operation

process parses it into its constituent elements. An analysis is conducted over its conceptual content and the query is transformed into a representation that is consistent with all the information items in storage. Given the query representation, the matching mechanism evaluates the relatedness of all the potential targets to the query and obtains a rank of relevance. An ordered set of information items is returned to the user by the output module. When interaction between the user and the retrieval system is available, one can communicate with the system through the feedback module by refining the query during one search session in the light of sample retrieval.

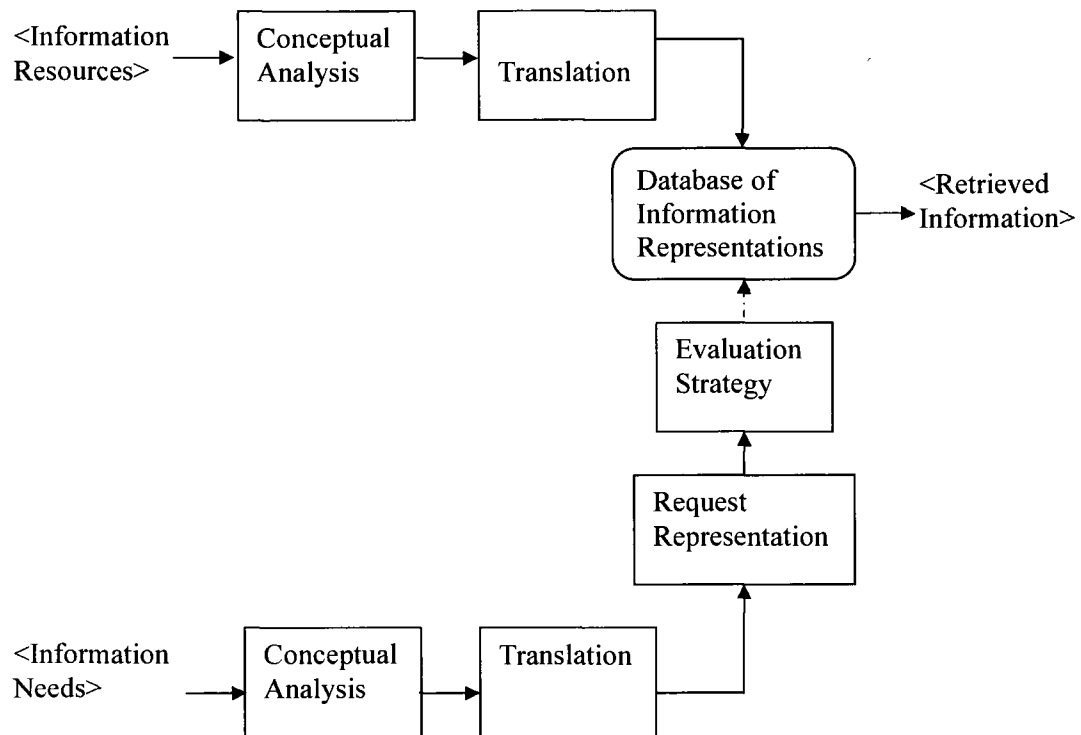


Figure 2.1: Framework of an Information Retrieval System

The user interface serves as a bridge connecting the client to the other modules of the system. The infrastructure of a typical IR system is depicted in Fig. 2.1.

2.5 Information Retrieval Approaches

In the field of information retrieval, there are two major categories of techniques: statistical analysis and semantic analysis. Statistical approaches consider the freeform natural language documents to be pure data records and index them in terms of some statistical measure. Assessment of the relevance between a pair of documents is also

based on a certain statistical metric. Semantic approaches, however, reproduce some degree of understanding of natural language texts that a human may provide.

By far, the greatest amount of work has been devoted to statistical approaches which fall into four categories: classical Boolean, extended Boolean, vector space, and probabilistic. The following sections briefly discuss classical Boolean, extended Boolean, and vector space approaches.

2.5.1 Classical Boolean Approach

The classical Boolean approach is based on the theory of Boolean algebra. A conventional Boolean query combines terms with the classical Boolean operators AND, OR, and NOT, and is evaluated by the classical rules of Boolean algebra. Such a model is very straightforward and relatively tractable to implement.

However, the classical Boolean method has some major limitations due to the characteristics of the standard Boolean model. Like all Boolean expression, the query only has two values: true or false. Correspondingly, a document is either relevant to a query or non-relevant to it. Therefore, no ranking strategy is possible in the classical Boolean method. With regard to the term weighting, only two values are available: 0 for an absent term and 1 for a present term. Such an all-or-nothing condition tends to have the effect that either an intimidating large amount of documents or none at all are retrieved [Harman 1992]. As well, the classical Boolean rules tend to produce counterintuitive results because of this all-or-nothing characteristic. For example, in response to a multi-term OR operation, “a document containing all (or many of) the query terms is not treated better than a document containing one term” [Salton & Buckley 1987]. Similarly, in response to a multi-term AND operation “a document containing all but one query term is treated just as badly as a document containing no query term at all” [Salton & Buckley 1987]. The ranking inefficiency of the classical Boolean model is a considerable issue that makes this method ineffective in document retrieval.

2.5.2 Extended Boolean Approach

As mentioned above, the classical Boolean scheme has ranking inefficiency and to overcome this inefficiency several extended Boolean models that integrate ranking

strategies are discussed in the literature. [Waller & Kraft 1979], [Salton & McGill 1983], [Paice 1984], [Zimmerman 1991], [Grei et al 1997]

The extended Boolean models employ extended Boolean operators, also called soft Boolean operators. These operators assign weights to the terms in each document. They also extend the truth value range from a discrete two-element-set: $\{0, 1\}$ in the case of classical Boolean model to a consecutive range: $[0, 1]$. In other words, the operators evaluate their arguments to a number, corresponding to the estimated degree to which a given logical expression matches the given document. By doing this, the extended Boolean methods are able to provide a ranked output allowing some documents to satisfy the query condition more closely than others. [Lee 1994] Therefore extended Boolean methods manage to overcome the limitation of the classical Boolean approach. From literature it is found that the extended Boolean model can achieve greater IR performance than either the classical Boolean or the vector space model. [Greengrass 2001] However, there is a big price for this performance improvement. Formulating effective extended Boolean queries involves more thought and expertise in the query domain than either the classical Boolean method or the vector space approach. [Greengrass 2001]

2.5.3 Vector Space Approach

A Vector space model (or term vector model) is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as terms. A Vector space model is used for various operations of IR such as information filtering, information retrieval, indexing and relevancy rankings. The vector space approaches have achieved great success in IR by applying the theory of linear algebra in this representation model.

In the traditional vector space method, the union of the effective terms defines a document space so that each distinct term represents one dimension in this space. For a given document, each term is assigned a numeric weight which estimates the usefulness of the term as a descriptor of the given document, i.e., the discriminatory power of the term for the document. By exploiting the weights of all the terms for a document, the document is then encoded as a term vector in the document space. It is worth noting that a query is usually considered to be a pseudo-document and can also be represented as a term vector.

Perspectives of document space and term space can be combined by viewing the entire collection as a term-by-document matrix, also called an indexing matrix. Each row of this matrix represents a term, and each column of this matrix represents a document. The element at row i , column j reflects the importance of term i in representing the characteristics of document j . A data set of d documents and t terms can be represented by the matrix shown below:

$$M = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1d} \\ m_{21} & m_{22} & \cdots & m_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{t1} & m_{t2} & \cdots & m_{td} \end{pmatrix}$$

Please note that any defined denotation, e.g., d , t and M , will be applicable to the rest of this text. An example of term-by-document matrix is given below:

Let us consider three documents (d1, d2, and d3) and each document contains one sentence:

d1: You read magazine.

d2: You play cricket.

d3: You like like like pizza.

The term-by-document matrix takes the following form for the above documents:

Terms	Documents		
	d1	d2	d3
you	1	1	1
read	1	0	0
play	0	1	0
cricket	0	1	0
like	0	0	3
pizza	0	0	1

Table 2.2: Term-by-document Matrix

We can also write down this term-by-document matrix as follows:

$$M_{6 \times 3} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 1 \end{pmatrix}$$

2.5.3.1 Term Weighting

One significant issue is that any vector space model needs to address term weighting, i.e. assigning weight to a term for a document so that the assigned weight properly reflects the contribution of the term in distinguishing the document from other documents. A variety of weighting schemes have been proposed, which basically fall into two categories: local weighting and global weighting.

Local weighting schemes attempt to reflect the importance of a term within a given document by a document-specific statistic. Usually, the local weights assigned to identical terms vary among documents. Some popular local weighting functions include the raw term frequency, binary, and logarithm of the term frequency. Let us denote L_{ij} to be the local weight of term i in document j and denote tf_{ij} to be the frequency with which term i appears in document j . The local weights for three weighting schemes are as follows:

- Raw Term Frequency: $L_{ij} = tf_{ij}$ 1(a)

- Binary $L_{ij} = \begin{cases} 1 & \text{if } tf_{ij} \geq 1 \\ 0 & \text{otherwise} \end{cases}$ 1(b)

- Logarithm $L_{ij} = \begin{cases} 1 + \log tf_{ij} & \text{if } tf_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$ 1(c)

There is a critical problem associated with raw term frequency. In the case of raw term frequency, all terms are considered equally important when it comes to assessing relevancy on a query; as a result, certain terms have little or no discriminating power in determining relevance, and thus have no effect in scoring. For example, a document with 10 occurrences of the term is more relevant than a document with one occurrence of the term but not ten times as relevant.

The problem with the binary weighting scheme is that it fails to capture the fact that some terms might be more important than others as the binary weighting scheme only considers the presence or absence of a term in a document.

It is worth mentioning that the logarithm weighting scheme exploits the logarithm function to transform the raw term frequency, thereby dampening the effects of large differences in frequencies of individual terms. For example, let us consider $tf_1 = 1000$ is the term frequency of a term i in the entire collection and $tf_2 = 10$ is the term frequency of a term j in the entire collection. The ratio of two raw term frequencies is $tf_1/tf_2=100$. If we go for logarithm weighting scheme (please see equation 1(c)) then local weight, L_1 for tf_1 is 4 and local weight, L_2 for tf_2 is 2, so after applying logarithm weighting schemes the ratio of two local weights $L_1/L_2=4/2=2$, so it is clear that the logarithm weighting scheme reduces the large differences in frequencies of individual terms.

Global weighting schemes on the other hand attempts to attenuate the effect of terms that occur too often in the collection to be meaningful in determining relevance of documents against a query. The idea is to scale down the term weights of terms with high collection frequency (defined to be the total number of occurrences of a term in the collection) by reducing the tf (term frequency) weight of a term by a factor that grows with its collection frequency. In addition to estimating the document-specific statistics, characterizing a term's overall importance in the whole collection can also be useful. Global weighting strategies are designed to measure the distribution of a term within the given collection. Thus they are able to estimate how frequently a term occurs in a certain document by chance. Generally, they give less weight to terms that occur frequently or occur in many documents because these terms are not considered to be strong descriptors for any specific document. Four well-known global weighting schemes are: normalized term frequency (or normal for short), inverse document frequency (or *idf* for short), term frequency-idf (or *tf-idf* for short), and entropy. Let G_i be the global weight assigned to term i , let gf_i be the frequency term i occurs in the entire collection, let df_i be the frequency of documents in which term i occurs, and let d be the number of documents in the whole collection.

The evaluation of G_i by the four types of global weighting methods is discussed below:

The normal weighting scheme normalizes the length of the vector for a term to 1, giving high weight to infrequent terms in the collection. However, normal weighting only depends on the sum of the squared frequencies, not the distribution of those frequencies per se. The following equation presents the normal weighting scheme:

$$\circ \text{ Normal: } G_i = \sqrt{\frac{1}{\sum_{j=1}^d tf_{ij}^2}} \quad 2(a)$$

The *tf-idf* and *idf* weighting schemes are closely related because both schemes weight terms inversely by the number of documents in which they appear. Neither method depends on the distribution of terms; rather they depend on the number of different documents in which a term occurs. The *tf-idf* weighting scheme is used to scale the frequency of a term. By nature *tf* assigns high weights to the frequent terms and low weights to the rare terms. On the other hand, *idf* (inverse document frequency) assigns high weight to a rare term and low weight to a frequent term. So, *tf-idf* term weighting scheme produces a composite weighting schemes with the following characteristics:

- Weight of a certain term becomes highest when the term occurs many times within a small number of documents.
- Weight of a certain term becomes lower when the term occurs fewer times in a document or occurs in many documents.
- Weight of a certain term becomes lowest when the term occurs in virtually all documents.
- The *tf-idf* weight values are strictly smaller than 1.

Reviewing the *tf-idf* weighting scheme in the IR literature leads to the following points being noted:

- *tf-idf* attenuates the effect of terms that occur too often in the input dataset.
- The discriminatory power of *tf-idf* allows the retrieval engine to quickly find relevant documents that are likely to satisfy the user.
- *tf-idf* exhibits an overall advantage over other term weighting system. [TREC-3, 1997]

The following equations present the *idf* and *tf-idf* weighting schemes:

- Idf: $G_i = \log\left(\frac{d}{df_i}\right)$ 2(b)

- Tf-idf: $G_i = tf_{ij} \times idf_i$ 2(c)

The entropy scheme is based on information theoretic ideas and is the most sophisticated weighting scheme. It exploits the distribution of terms over documents. The main idea is to assign less weight to terms that are equally distributed over all the documents and assign more weight to terms that are concentrated in a few documents. [Dumais 1991], [Manning et al 2008] The following equation presents the entropy weighting scheme:

- Entropy: $G_i = 1 - \sum_{j=1}^d \frac{p_{ij} \log(p_{ij})}{\log(d)}$ where $p_{ij} = \frac{tf_{ij}}{gf_i}$ 2(d)

All of the global weighting schemes share a principle of assigning less weight to terms that occur frequently. Global weights involve variations in the relative importance of local frequency, global frequency, and document frequency.

We can combine local and global weighting schemes to measure weight of a term as each of the weighting schemes contains advantages and limitations. There is not a fixed solution for choosing a term weighting scheme. Both local weights and global weights are used to measure the term weights; the value of the i th row, j th column element can be evaluated as shown below:

- $m_{ij} = L_{ij} \times G_i$ (3)

If you look at the individual value of m in the matrix M [section 2.5.3] and the local and global weightings equations [1(a), 1(b), 1(c) and 2(a), 2(b), 2(c)], it is clear that the value of m is dependent on both local and global weights of terms.

2.5.3.2 Normalization (equivalence classing of terms)

After getting the tokens from the documents and query, the simple case is if tokens in the query just match with the tokens in the token list of a document. However, there are many

situations when two character sequences are not quite the same but you would like a match to occur. For instance, if we search for UK, we might hope to also match documents containing U.K.

Term/Token normalization is the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens. The standard way to normalize is to implicitly create equivalence classes, which are normally named after one member of the set. For example, if the tokens anti-war and antiwar are both mapped onto the term anti-war, in both the document text and queries, then searches for one term will retrieve documents that contain either. [Manning et al 2008]

2.5.3.3 Discussion

As an efficient model, the traditional vector space scheme is becoming very popular in IR research. Since it has a sound mathematical foundation, it applies a similarity measurement [see section 2.5.4.5] technique between document and query to produce a ranked list of relevant documents.

The traditional vector space approach provides an effective way to approximate the statistical properties of the document set. The major problem that exists with this method is that it assumes all the terms are independent, orthogonal dimensions of the document space so it simply ignores the relationship among terms. However, as it is a fact that there are strong associations among terms in natural languages, the above assumption is never satisfied [Hull 1994]. Another drawback of this model in some applications is that the number of terms that occur in a collection can be quite large; the traditional term-based document space tends to have a large number of dimensions. Some alternative approaches such as LSI based on the traditional vector space model have overcome these limitations.

2.5.4 Latent Semantic Indexing

Latent semantic indexing (LSI) is a method that exploits the idea of vector space model and singular value decomposition (SVD).

2.5.4.1 Motivation of LSI

In the research of retrieving free-form natural language data, it is always useful to analyze the features of human verbal behavior. There are two issues that are discussed the most:

synonymy and polysemy. Synonymy refers to the states when two or more words or expressions have the same or nearly the same meaning in some or all senses. Polysemy describes the cases when one word has multiple meanings. These characteristics of natural languages result in the deficiencies of some IR algorithms that do not offer comparison methods on terms. The Latent semantic indexing method was proposed in order to capture the statistical relationship among terms and accordingly provide an effective solution to the problems of synonymy and polysemy that cannot be tackled by either word-based approaches or the traditional vector space approach.

2.5.4.2 Theory

Latent semantic indexing, also called latent semantic analysis (LSA) method, was first proposed by Deerwester et al. [Deerwester et al 1990]. LSI assumes that, in the textual data, there is some underlying latent semantic structure that is partially obscured by the randomness of word choice with respect to retrieval. This structure can be estimated by statistical techniques and the obscuring noise can be removed.

Like the traditional vector space approach, the LSI method starts with a term-by-document matrix that represents the association of terms to documents. It applies a dimensional reduction scheme, singular value decomposition (SVD), on the matrix to construct a reduced unified semantic space for retrieval. This smaller space, called LSI space, consists of derived dimensions that are assumed to convey truly independent concepts. By employing the dimensional reduction strategy, LSI not only captures most of the important underlying semantic structure in associating terms with documents, but also has a better chance in removing the noise or possible variability in word usage.

2.5.4.3 Singular Value Decomposition

Singular value decomposition is an effective dimensional reduction scheme. It is closely related to a number of mathematical and statistical techniques that have been widely used in other fields, such as the principal component analysis (PCA) for image processing and face recognition. SVD has been proved to be a very good choice for uncovering latent semantic structure (See [Deerwester et al 1990] for a further discussion of SVD and the other alternative models).

SVD can be applied with an arbitrary rectangle matrix to decompose the matrix into three matrices containing singular vectors and/or singular values. These three matrices with special forms show a breakdown of the original matrix into linearly independent components or factors. Many of these components are very small, leading to an approximate model that contains many fewer dimensions. Thus, for information retrieval purposes, SVD provides a reduced model for representing the term-to-term, document-to-document and term-to-document relationships.

By dimension reduction, it is possible for documents with somewhat different profiles of term usage to be mapped into the same vector of factor values [Deerwester et al 1990]. This property helps to eliminate the noise in the original data, thus improving the reliability of the algorithm.

Suppose we obtained a $t*d$ term-by-document matrix M from the collection indexing process of the traditional vector space method. We can apply SVD on M , which is then decomposed into three special matrices U , S , and V . The decomposition can be written as:

$$M = USV^T \quad 4(a)$$

U is the $t*t$ orthogonal matrix ($UU^T=I_t$) having the left singular vectors of M as its columns, and V is the $d*d$ orthogonal matrix ($VV^T=I_d$) having the right singular vectors as its columns, and S is the $t*d$ diagonal matrix having the singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(t,d)}$ of M in order along its diagonal. It should be noted that for any arbitrary matrix, such a factorization exists. For details, please see [Strang 1998].

Generally, in Eq. 6, the matrices U , S and V must all be of full rank. However, one great facilitation is that SVD offers a simple strategy for optimal approximate fit using smaller matrices. [Deerwester et al 1990] If the singular values in S are ordered by size, the first k largest values may be kept and the remaining smaller ones are set to zero. The product of the resulting matrices is a matrix M_k which is only approximately equal to M , and is of rank k . Since zeros were introduced into S , the representation can be simplified by deleting the zero rows and columns of S to obtain a new diagonal matrix S_k , and then deleting the corresponding columns of U and V to obtain U_k and V_k respectively. The rank- k model with the best possible least-squares-fit to M can be written as following:

$$M_k = U_k S_k V_k^T \quad 4(b)$$

Where, M_k is a matrix of size $t*d$, U_k is of size $t*k$, S_k is of size $k*k$, and V_k is of size $k*d$.

SVD provides an optimal solution to dimensionality reduction in that it derives an orthonormal space, where the dimensions are ordered. Therefore, projecting the set of documents onto the k lowest dimensions is guaranteed to have, among all possible projections to a k dimensional space, the lowest possible least-square distance to the original documents. [Schutze & Silverstein 1997]

The following section presents a numerical example of SVD calculation.

Let us consider a $t*d$ ($t>d$) matrix M , where $t=3$ and $d=2$:

$$M = \begin{pmatrix} 1 & -1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

We decompose this matrix by using the equation 6, and we get the following values:

$M=USV^T$ where, U is a $t*t$, S is a $t*d$, and V^T is a $d*d$ matrix.

$$U = \begin{pmatrix} -0.816 & 0.000 & -0.057 \\ 0.408 & -0.707 & -0.057 \\ -0.408 & -0.707 & 0.057 \end{pmatrix}$$

$$S = \begin{pmatrix} 1.732 & 0.000 \\ 0.000 & 1.000 \\ 0.000 & 0.000 \end{pmatrix}$$

$$V^T = \begin{pmatrix} -0.707 & 0.707 \\ -0.707 & -0.707 \end{pmatrix}$$

$$M=USV^T = \begin{pmatrix} -0.816 & 0.000 & -0.057 \\ 0.408 & -0.707 & -0.057 \\ -0.408 & -0.707 & 0.057 \end{pmatrix} \begin{pmatrix} 1.732 & 0.000 \\ 0.000 & 1.000 \\ 0.000 & 0.000 \end{pmatrix} \begin{pmatrix} -0.707 & 0.707 \\ -0.707 & -0.707 \end{pmatrix}$$

The reduced singular value decomposition of matrix M is given below:

$$M=USV^T = \begin{pmatrix} -0.816 & 0.000 \\ 0.408 & -0.707 \\ -0.408 & -0.707 \end{pmatrix} \begin{pmatrix} 1.732 & 0.000 \\ 0.000 & 1.000 \end{pmatrix} \begin{pmatrix} -0.707 & 0.707 \\ -0.707 & -0.707 \end{pmatrix}$$

2.5.4.4 Document and Query as Vectors

To find the ranked list of documents we need to apply a similarity measurement between a set of documents against a query. In this thesis we used cosine similarity [section 2.5.4.5] measure. To do this we need to derive individual document and query vectors. From SVD calculation we have the following equation:

$$\text{Equation-4(a): } M=USV^T$$

$$\text{Equation-5: } M^T=(USV^T)^T$$

$$\text{Equation-6: } M^T=U^T S^T V$$

$$\text{Equation-7: } M^T US^{-1}=U^T SVUS^{-1}$$

$$\text{Equation-8: } V=M^T US^{-1}$$

Now the document and query vectors are represented by the following equations:

$$\text{Document Vector: } DR_j=DR_j^T US^{-1}$$

$$\text{Query Vector: } DR_q=DR_q^T US^{-1}$$

Thus the reduced K -dimensional LSI space can be presented as:

$$\text{Document Vector: } DR_j=DR_j^T U_k S_k^{-1} \quad 9(a)$$

$$\text{Query Vector: } DR_q=DR_q^T U_k S_k^{-1} \quad 9(b)$$

2.5.4.5 Similarity Measurement

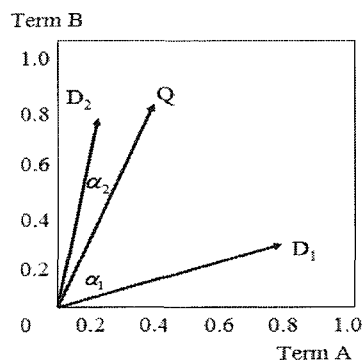
After deriving the term vector of documents and queries, a similarity measurement technique is applied to estimate the similarity between a pair of vectors. This similarity value is an indicator of how relevant the document is to the query.

A common similarity measure employed in the vector space model is called cosine similarity. In case of cosine similarity, the lower angles presents higher similarity and higher value of an angle represent dissimilarity between a query and a set of documents. Denote $|DR|$ to be the length of vector DR . The cosine similarity between vector DR_j and DR_q denoted by $\cos(DR_j, DR_q)$, can be evaluated by the following equation:

$$\circ \cos(DR_j, DR_q) = \frac{DR_j \cdot DR_q}{|DR_j| |DR_q|} \quad (10)$$

Two strategies are commonly used to utilize the similarity values for retrieving relevant documents. One is called range queries, which is to retrieve all documents up to a distance threshold. The other one is called nearest-neighbor queries, which is to retrieve the N best matches. Although we do not expect any similarity metric to be a perfect model that corresponds exactly with human judgment of relevance, the measurement should somehow be able to assign higher values to the documents with a higher proportion of the relevant text as well as assigning lower values to the documents with less relevant content. By integrating the ranking strategy into IR systems, the human user can restrict their attention to a set of documents of manageable size instead of having to go through every single record in the data set.

The following figure presents the idea of cosine similarity graphically:



Where:

D_1 represents document 1,

D_2 represents document 2 and

Q represents a query

α_1 is the angle between D_1 and Q

α_2 is the angle between D_2 and Q

Angle α (Degree)	Cosine(α)
0	1
30	0.86
45	0.70
60	0.5
90	0

Table 2.3: Angles and Their Cosine Values.

The Table 2.3 shows that when the angle between two vectors is very small, the cosine approaches 1. More precisely, when two documents are identical their cosine is 1; when they are orthogonal (no common terms/ totally dissimilar), their cosine is 0. As we are interested in rank-based retrieval and cosine similarity measure is a tool that helps us to rank the documents based on the similarity values against a query, we are using cosine similarity to rank the documents for retrieval purpose.

2.5.4.6 Retrieving Documents Using LSI

The retrieval process sorts the documents according to their similarity values, and returns a ranked list of documents to the user. The retrieval process represents documents and queries in the same manner and applies a certain function to estimate the similarity between them.

In the LSI method, each document/query is projected onto the LSI space that is obtained by using SVD. LSI then exploits the cosine measurement between the projections of a pair of term vectors in the LSI space to make comparison between the two documents. Thus, the similarity can be obtained by computing the cosine value of the angle between the document's term vector and the query's term vector. All the documents can be ranked according to their similarity values and an ordered set of documents retrieved.

2.6 TREC and Relevance Judgment

Relevance judgment is an obvious part of a standard IR dataset. Relevance judgments help us to measure recall-precision by indicating which documents are relevant to which topics or query. The following sections discuss the background information of TREC-8 relevance judgment.

2.6.1 Text REtrieval Conference-8 (TREC-8) Relevance Judgment

Relevance Judgments are of critical importance to a test collection. For each topic it is necessary to compile a list of relevant documents – as comprehensive a list as possible. All TRECs have used the pooling method to assemble the relevance assessments. In this method a pool of possible relevant documents is created by taking a sample of documents selected by the various systems of participating research groups. The pool is then shown to the human assessor, who makes a binary (yes/no) relevance judgment for each document in the pool. Un-judged documents are assumed to be not relevant. The particular sampling method used in TREC is to take the top 100 documents (pooling method with the depth 100) retrieved per judge run for a given topic and merge them into the pool for assessment. This is a valid sampling technique since all the systems used ranked retrieval methods, with those documents most likely to be relevant returned first. Each pool is sorted by document identifier so assessors cannot tell if a document was highly ranked by some systems or how many systems (or which systems) retrieved the document.

The method can be summarized as:

1. To have dozens of research groups from universities and companies participate.
2. To run all 50 queries through their system (TREC-8, Topic 401-450).
3. To submit raw retrieval results (Take the top 100 highest ranked documents from each topic e.g., TREC-8: 7100 documents).
4. To merge them into the candidate set e.g., TREC-8: 1736 documents.
5. To have human assessors judge relevance of each document.
6. To evaluate results and compile of performance statistics (e.g., TREC-8: 94 documents).

2.6.2 Relevance Judgement Problems

There are two potential problems to preparing the relevance judgments. Those are:

1. Human assessors make errors; and
2. There are often many more relevant documents in the corpus beyond the candidate set; TREC procedure will consider them all irrelevant.

A very recent research paper published in SIGIR 2008, Singapore tried to assess the accuracy of TREC relevance judgments with their own relevance judgments created by

their own human users and they found 63% similarity with TREC relevance judgments. [Al-Maskari et al 2008]

2.7 Evaluation of Information Retrieval Performance

In previous sections, we presented the basic architecture of an IR system and introduced some representative approaches that can be used to implement it. In this part of the chapter, we discuss the criteria that may be investigated while evaluating an IR system.

Frakes et al. [Frakes & Yates 1992] provided a summary of the evaluation process: an information retrieval system can be evaluated in terms of many criteria, including execution efficiency, storage efficiency, retrieval effectiveness and the features they offer a user. The system designers look into the relative importance of these factors based on the particular environment and expectations. Accordingly, they select appropriate data structures and algorithms for implementation.

Execution efficiency is measured by the time it takes a system, or part of a system, to perform a computation. This can be measured in C based systems by using profiling tools such as *prof* on UNIX. This factor has always been a major concern for the interactive IR systems because a long retrieval time will interfere with the usefulness of the system. The requirements of such IR systems usually specify maximum acceptable times for searching, and for database maintenance operations such as adding and deleting documents.

Storage efficiency is measured by the number of bytes needed to store the data. Space overhead, a common measure of storage efficiency, is the ratio of the sizes of the index files plus the size of the document files over the size of the document files.

Most IR experimentation has focused on retrieval effectiveness, which is based on relevance judgment. Relevance is an inherently subjective concept in that the ultimate goal is to satisfy the human users' needs. Due to the variety of the user's personal background, it is impossible to design a perfect system that meets all the expectations of all human users. Therefore, it is necessary to introduce a method to evaluate the performance of a retrieving process by estimating the degree of relevance at which the

retrieved information matches the query. In this thesis we used recall-precision and paired sample t-test to evaluate our retrieval systems.

2.7.1 Recall and Precision

Two widely used parameters to measure IR success, which are based on the concept of relevance, are precision and recall. Precision is the ratio of relevant items retrieved to all items retrieved. Recall is the ratio of relevant items retrieved to all the relevant items. To facilitate the understanding of the definitions, the following equations represent the recall and precision:

$$recall = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of relevant documents}} \quad (11)$$

$$precision = \frac{\text{Number of relevant documents retrieved}}{\text{Total number of documents retrieved}} \quad (12)$$

The expectation of the users may vary from one person to another. Some users attach more importance to precision, i.e., they want to see relevant information without going through a lot of junk. Others take recall as a preference, i.e., they want to see all the documents that are considered to be highly relevant. Hence, the evaluation that involves only one of the two parameters may be biased. Due to this reason, some methods that evaluate the IR performance in terms of precision and recall simultaneously, have been developed.

2.7.2 Recall-Precision Calculations and Plotting

The following section explains how to calculate and plot recall-precision using a numerical example. For more information [Manning et al 2008], [Croft et al 2009].

According to the theory of information retrieval evaluation, precision is calculated at 11 standard levels of recall 0, 10, 20 100 percent (or 0, 0.1, 0.2 ... 1.0). To measure the performance of an IR system, multiple queries or topics (e.g. 50 queries) are used. The precision averages at 11 standard recall levels are used to compare the performance of different systems and as the input for plotting the recall-precision graph (see Fig-2.2 and 2.3). Each recall-precision average is computed by summing the interpolated precisions at the specified recall level by using the following standard rule:

- For any standard recall level i , take maximum precision at any actual recall level $\geq i$
- This defines a precision at the standard recall of 0 even though precision at actual recall 0 is undefined

Let us consider the list of relevant documents:

Relevant Documents (9)/DocID								
0123	0132	0241	0256	0299	0311	0324	0357	0399

Let us consider an IR system that retrieved 12 documents as shown in Table-2.4. The relevant documents are shaded. The values of recall and precision of a single query were calculated by using the equations 11 and 12. Table-2.5 represents the interpolated precision of the same query.

Rank	DocID	Recall	Precision
1	0234	0	
2	0132	0.111	0.5
3	0115	0.111	
4	0193	0.111	
5	0123	0.222	0.4
6	0345	0.222	
7	0387	0.222	
8	0256	0.333	0.375
9	0078	0.333	
10	0311	0.444	0.4
11	0231	0.444	
12	0177	0.444	

Table 2.4: Recall-precision of a single query

Recall level	Interpolated Precision
0	1.0
10	0.5
20	0.4
30	0.4
40	0.4
50	0
60	0
70	0
80	0
90	0
100	0

Table 2.5: Interpolated precision of a query

Fig 2.2 represents the recall-precision graph using the Table 2.5.

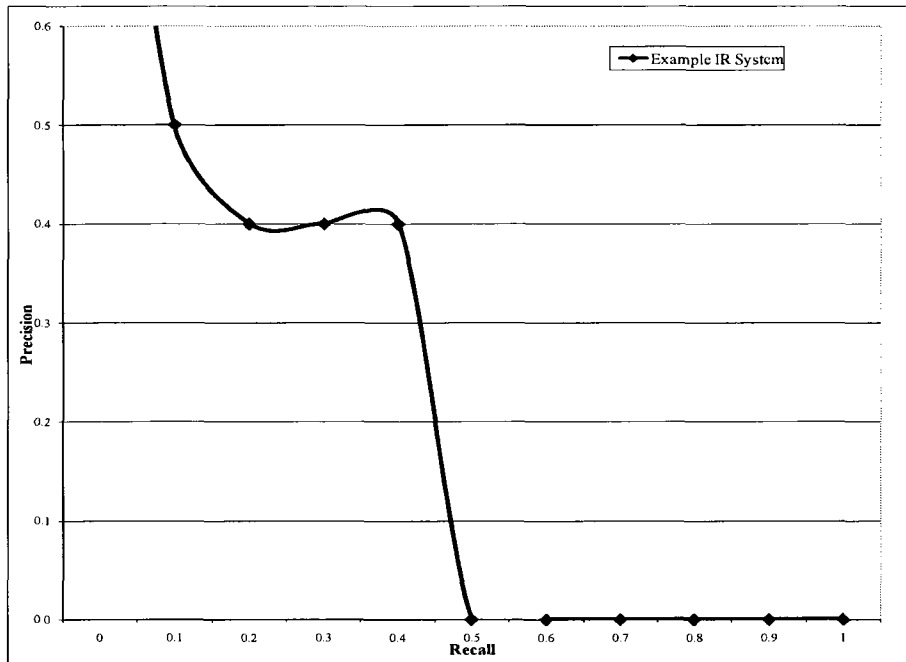


Figure 2.2: Recall-Precision Graph of a Query

We already mentioned that the performance of a single query does not necessarily represent the performance of a retrieval system. So, we need to compute the precision for multiple queries and average the interpolated precision values at each recall level. Table-2.6 represents the average interpolated precision of two different retrieval systems (System-1 and System-2).

Recall level	Average Interpolated Precision	
	System 1	System 2
0	1.0	1.0
10	0.5	0.8
20	0.4	0.75
30	0.37	0.6
40	0.33	0.6
50	0.33	0.5
60	0.33	0.4
70	0.2	0.37
80	0.2	0.33
90	0.12	0.2
100	0.0	0.0

Table 2.6: Interpolated Average Precision of Two Different Systems

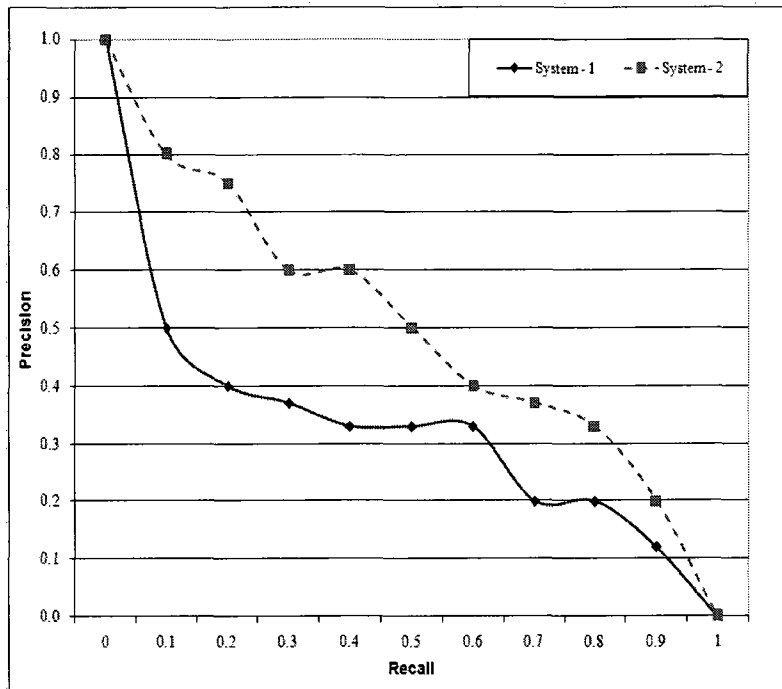


Figure 2.3: Recall-precision Graph of Two Retrieval Systems

Fig 2.3 represents the recall-precision graph of the Table 2.6

2.8 Paired Sample T-test

The paired samples t-test compares the means of two variables to determine whether there is a statistically significant difference between two populations. We could do this by calculating confidence intervals for the two populations and seeing if the mean (\bar{x}) of one population lies outside the 95% confidence interval of the other population. A formalized and slightly less cumbersome approach to answering this question is the *t-test*. A confidence interval (CI) is a range of values within which lies the true population mean. There are many possible confidence intervals. Each interval specifies a probability of “capturing” the true population mean. [Triola 2006] To perform the t-test we need to follow the following steps:

1. Choose the hypothesis:
 - Null: There is no significant difference between the means of the two variables.
 - Alternate: There is a significant difference between the means of the two variables.
2. Select the level of significance:

- In paired sample t-test, after making the hypothesis, we choose the level of significance. In most of the cases in the paired sample t-test, significance level is 5% or 0.05.

After doing the calculations of the t-test if we get the significance values greater than 0.05 it means that there is a significant difference between two systems otherwise the null hypothesis holds (no significant difference). Table-C in the appendix D represents the different values for significance level 5% or 0.05.

We calculated precision for different retrieval systems (by varying parameters e.g. weighting scheme, stop word). Every retrieval system is independent; however precision was calculated on the same dataset. Paired sample t-test is used to compare two means that are repeated measures for the same dataset. So, we used a paired sample t-test to find the significance of the difference among the precision levels of retrieval systems to compare the performance of the retrieval systems.

Chapter-3

Issues in Latent Semantic Indexing (LSI) Research

3.1 Introduction

Information retrieval (IR) deals with the representation, storage, organization of, and access to information. The representation and organization of the information should provide users with easy access to the information in which they are interested. Unfortunately characterization of the user's *information need* is not a simple problem. [Yates & Neto 1999]

For hundreds of years, people have understood the importance of archiving and finding information. With the advancement of computer technology, it is now possible to store large amounts of information and finding useful information from such collections became a necessity. On the basis of this necessity, the field of Information Retrieval (IR) was born in the 1950s. Over the last few decades, the field of IR has matured considerably. A number of automatic Information Retrieval systems are used on a daily basis by a variety of users. [Singhal 2008]

3.2 Related Research Work on Latent Semantic Indexing (LSI)

This chapter examines the short comings and problems in existing published works on Latent Semantic Indexing (LSI), one of the most popular techniques in the area of IR. LSI is more fully discussed in the section 2.5.4, if the reader requires more background. The published works revealed that there are open questions about very large datasets, stop words, and term weighting schemes in LSI. This thesis attempts to answer these questions, particularly for very large datasets used in LSI. The following sections discuss the related published research that used the LSI algorithm.

3.2.1 Using Latent Semantic Analysis to Improve Access to Textual Information

The first research work published on LSI was entitled "Using latent semantic analysis to improve access to textual information". [Dumais et al 1988] The authors used two different datasets. The descriptions of the datasets are given below:

- **MED:** The first database consisted of 1033 medical reference abstracts and titles. A 5823*1033 term-document matrix was obtained and retrieval effectiveness evaluated against 30 queries available with the dataset.

- **CISI:** The second standard dataset consisted of 1460 information science abstracts. A 5135*1460 term-by-document matrix was obtained and retrieval performance evaluated using 35 queries available with the dataset.

They did not use any stemming algorithm or remove stop words. Points to be concluded about this paper are:

The dataset is small for use in IR. For the first dataset (1033 medical reference abstracts and titles), the researchers claim that LSI shows 13% average improvement over raw term matching and show that LSI captured some structure in the data that was missed by raw term matching. For the second dataset (1460 information science abstracts), the authors do not claim any improvement of LSI over raw term matching technique. Homogeneity of documents and poorly stated queries in the second dataset caused very poor performance in precision level, so, the authors said that CISI was not a reliable dataset.

3.2.2 Indexing by Latent Semantic Analysis

Deerwester's [Deerwester et al 1990] paper "Indexing by latent semantic analysis" describes automatic indexing and retrieval. This paper also described the use of singular value decomposition. They used datasets with standard queries from the Medlars collection (same dataset mentioned in section 3.2.1).

They removed stop words (SMART's stop word list was used) but they did not use stemming on the input dataset. By reviewing their results we find the following weakness in their work:

LSI was evaluated with a very small dataset (input data and query set). It was a poor choice for a test collection because the selected dataset contains homogeneous documents. Some queries are vague and poorly stated.

In the first experiment, they claimed 13% average improvement in LSI results over the raw term matching technique. In another experiment, the authors claimed 50% improvement in retrieval performance in case of LSI (k=100/100-factor) against factor analytic techniques for information retrieval using small numbers of factors e.g., 7 dimensions, 13 dimensions and, 21 dimensions.

The authors only tested the above mentioned dataset with LSI and compared their results against a straightforward term matching method, a version of SMART, and the vector method reported by Voorhees (1985) for the same standard datasets. The SMART and Voorhees systems are information retrieval systems, and have different indexing, term weighting, and query processing procedure than LSI method.

A set of 439 common (stop) words was applied to reduce the number of terms. However, they did not use stemming on the input dataset. They summarize their findings, as “These results are modestly encouraging. The latent semantic indexing method is superior to simple term matching in one standard case and equal in another” when considering two different datasets. [Deerwester et al 1990]

3.2.3 Large-Scale Information Retrieval Using Latent Semantic Indexing

Todd A. Letsche [Letsche 1996] used the datasets shown in Table 3.1 in his master's thesis at the University of Tennessee, Knoxville, USA.

Document Collection	Abbrev.	Number of Terms	Number of Documents
PVM: parallel Virtual machine [GBD ⁺ 94]	PVM	2547	170
All of the following Combined: PVM: parallel Virtual machine [GBD ⁺ 94] LAPACK User's Guide Release 2.0 [ABB ⁺ 95] MPI: The Complete Reference [SOHL ⁺ 96] Templates for the Solution of Linear Systems [BBC ⁺ 94] Parallel Computing works [FWM94]	PLMTP	9842	1154
The Concise Columbia Encyclopedia	CCE	31110	15486
Usenet News Archives	USENET	534493	100000

Table 3.1: A set of datasets [Letsche 1996]

By reviewing his results we find the following issues:

Letsche removed stop words and applied Porter's stemming algorithm to reduce the terms. However, the length of the stop word list is not mentioned. LSI was

used to search the World Wide Web. He used a large dataset of 100,000 documents from USENET. However, Letsche claimed due to memory constraint (low hardware resources), the system was unable to search the USENET dataset. [Letsche 1996] In his thesis, the results are only based on small dataset. LSI achieved up to 30% better retrieval performance than lexical searching technique (Lexical analysis based). Lexical analysis searching technique compare tokens/terms to find the matching. This technique does not use singular value decomposition (SVD) or cosine similarity to measure the relevance.

3.2.4 Automatic Cross-Language Information Retrieval using Latent Semantic Indexing

Michael [Dumais et al 1997] used 2,482 documents for experimentation in the area of automatic cross-language information retrieval using latent semantic indexing. They used French and English documents. In this work, queries in one language can retrieve documents in the other language (as well as the original language).

Reviewing the findings of this paper, the following is noted:

They did not use any standard dataset (like TREC). They did not remove stop words or apply stemming. The authors compared LSI result with human generated retrieval. With the human generated text, the overall performance of this LSI system was about 10% worse for retrieving the corresponding category, but 15% better when looking at the top 10 retrievals. [Dumais et al 1997]

3.2.5 Latent Semantic Indexing-Based Intelligent Information Retrieval System for Digital Libraries

Aswani [Kumar et al 2006] used titles of 116 journals and periodicals available in the Vellore Institute of Technology, India as their own input dataset. Reviewing the findings of this paper, the following is noted:

The authors used a very small dataset and the data set and queries were not standard. There is no numerical indication of the improvements in retrieval performance. They applied the Porter's stemming algorithm on the input data set; however, they did not remove stop words. Authors mentioned in their paper that LSI has superiority over the standard vector space model. [Kumar et al 2006]

3.2.6 Singular Value Decomposition and Rank K Approximation

Bast and Majumder's [Bast & Majumder 2005] paper "Why Spectral Retrieval Works" studied singular value decomposition method by varying the value of K (rank K approximation) on three different (in size and content) datasets. They tried to find a feasible value of K by measuring the relatedness scores between two terms. Using this method they choose some values of K and measured the retrieval performance of LSI and LSI-based methods. They used the following dataset and corresponding queries for their experiments:

- Time Collection contains 425 documents and 3882 unique terms.
- Reuters Collection contains 21,578 documents and 5701 unique terms.
- Ohsumed Collection contains 233,445 documents and 99,117 unique terms

Reviewing the findings of this paper, the following is noted:

The Ohsumed dataset is large. For all three datasets, by varying the value of K it is found that the value of K does not play a significant positive role in the retrieval performance. For all three datasets, a bigger value of K reduces the relatedness of word pairs, so, reduces the retrieval performance in terms of recall-precision measures.

3.2.7 Singular Value Decomposition for Text Retrieval

Husbands' [Husbands et al 2001] paper entitled "On the Use of the Singular Value Decomposition for Text Retrieval" explored LSI retrieval performance for large datasets. The authors used the following datasets for their experiments:

- **MEDLINE/ MED collection** (please see section 3.2.1 for details): This dataset is an example of a small dataset.
- **TREC-6 Dataset:** A collection with 115,000 terms and 528,155 documents. This is an example of a large dataset.
- **NPL Dataset:** NPL dataset contains 11,429 documents and it is an example of a small dataset.

In their results they compare the performance of LSI with term matching retrieval. For MED data set, LSI ($K=200$) showed better retrieval performance than the term matching technique. However, the large TREC-6 dataset LSI ($K=200$) showed poor retrieval (lower

recall-precision values) performance. The following figure presents the performance in terms of recall-precision graph:

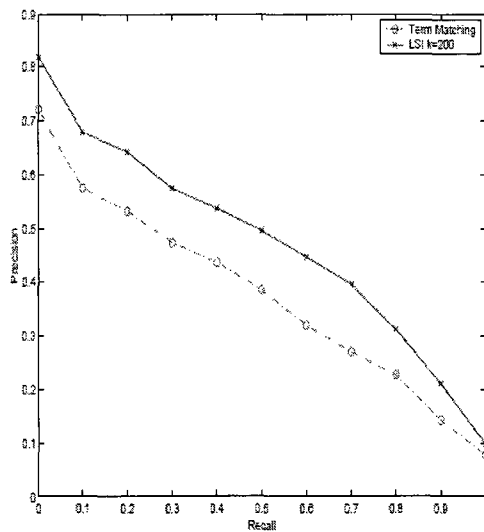


Figure 3.1: LSI vs. Term Matching on MEDLINE [Husbands et al 2001]

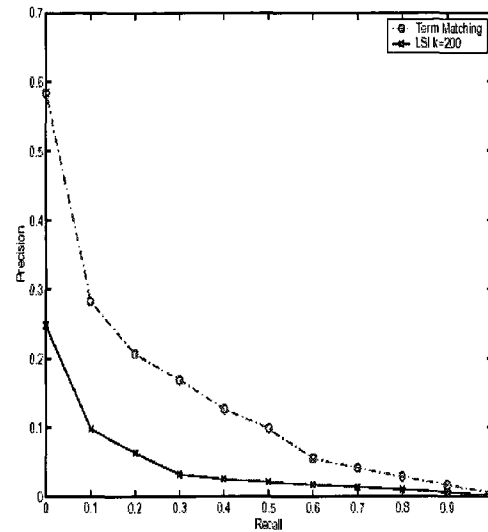


Figure 3.2: LSI vs. Term Matching on TREC6 [Husbands et al 2001]

Reviewing the findings of this paper, the following is noted:

Most LSI research shows LSI is superior than term matching technique and other methods [Dumais et al 1988][Deerwester et al 1990][Kumar et al 2006][Letsche 1996], however, this paper shows poor performance of LSI over the term matching technique in case of a specific large dataset. Stemming improves retrieval performance [Hull 1995], but the authors did not use stemming on their input dataset. Term weighting schemes have positive effects on retrieval performance, and tf-idf and log-entropy weighting schemes are proven weighting schemes in IR. However, the authors only used idf weighting schemes for their experiments. The authors proposed a new method called “Term norms”, and mentioned that this method has great influence and variability however; they showed LSI ($K=300$) had poor performance.

Husbands and co-authors reported poorer performance of LSI ($K= 200$ and $K=300$), and they did not use stop word removal or stemming on the TREC-6 input dataset. They only used the idf term weighting scheme in their experiment. From the IR literature, we know that pre-processing (stemming and stop word removal) and term weighting schemes improve retrieval performance significantly, so, we are encouraged to test the

performance of LSI on a large dataset with the use of pre-processing of the input dataset and the most accepted weighting schemes (tf-idf and log-entropy).

3.3 Conclusion

From the above discussion it is found that performance of LSI has been tested for several smaller datasets (e.g., MED, CISI abstracts etc) however, LSI has not been tested for a large standard dataset. In this research, we concentrated on the performance of LSI on a standard large dataset by varying different parameters e.g. stop word lists and term weighting schemes. The specification of the chosen standard large dataset or test collection is given below:

- Dataset: LA Times dataset TREC-8 text collection. It includes articles published in the years 1989 and 1990 (all articles from 1 Jan 1989 to 31 Dec 1990). The number of the documents is 131,321 and size is 508 MB.
- Topics (query): TREC-8 query set includes 50 standard queries (401-450).
- TREC-8 test collection also provides the relevant judgment information associated with each topic (query) for the above-mentioned dataset.

The significance of performance of retrieval systems is mentioned in terms of recall-precision values/graphs, and t-test.

Chapter-4

Test Collection and Pre-processing

4.1 Introduction

To have the answers to our research questions from LSI based information retrieval system, we need to perform the following steps:

1. Standard dataset/test collection selection and pre-processing;
2. Processing of data and query using LSI based IR system to retrieve information;
and
3. Evaluating the LSI based IR system for retrieval performance.

This chapter thoroughly discusses the above mentioned first step that focuses on the standard test collection and pre-processing of input data. Pre-processing includes extracting documents from large SGML files, stop word removal, and stemming. We developed our own software for pre-processing of the selected dataset. The following sections discuss the preprocessing steps in mere detail.

4.2 Standard Test Collection

Ad hoc information retrieval is the task where the document collection is fixed, and users submit queries to the information retrieval system, and the system returns a set of ranked retrieval results (i.e., documents). To experiment with a LSI based ad hoc information retrieval (IR) system, it is important to have a standard test collection consisting of three things: A document collection, a test suite of information needs expressible as queries, and a set of relevance judgments (a binary assessment of either relevant or non-relevant for each query-document pair). [Manning et al 2008] For LSI based ad hoc IR, the most well known and recognized test collection is provided by the Text Retrieval Conference (TREC).

4.2.1 TREC and Standard Test Collection

The U.S. National Institute of Standards and Technology (NIST) has run a large IR test bed evaluation series since 1992. Within this framework, there have been many tasks/tracks (e.g., Blog Track, Chemical IR Track, Legal Track, Filtering Track and Genomics Track, etc.) over a range of different test collections, but the best known test collections are the ones used for the TREC Ad Hoc task or track during the first eight TREC evaluations between 1992 and 1999. In total, these test collections contain 1.89

million documents (mainly, but not exclusively, newswire articles) and relevance judgments for 450 information needs, which are called topics and specified in detailed text passages. Individual test collections are defined over different subsets of this data. The early TRECs each consisted of fifty information needs, evaluated over different but overlapping sets of documents. TRECs 6 through 8 provide 150 information needs over about 528,000 newswire and Foreign Broadcast Information Service articles. This is probably the best sub-collection to use in future work, because it is the largest and the topics are consistent. Because the test document collections are so large, there are no exhaustive relevance judgments. Rather, NIST assessors' relevance judgments are available only for the documents that were among the top K documents (e.g., top $K=100$ documents) returned for some system that was entered in the TREC evaluation for which the information need was developed. For this thesis, Los Angeles Times (LA Times) test collection from TREC-8 has been chosen for conducting this research.

4.2.2 Los Angeles Times (LA Times) Text Collection (TREC-8)

The data/text collection includes the articles published by the Los Angeles Times in the two year period from Jan 1, 1989 - December 31, 1990. Each file contains the articles from one day (e.g., a file with the name "LA123190" contains articles published on 31 Dec 1990). Every such file contains a number of documents (e.g., the LA123190 contains 134 different documents). The following figure represents part of the content of file LA123190.

```

<DOC>
<DOCNO> LA123190-0001 </DOCNO>
<DOCID> 329361 </DOCID>
<DATE>
<P>
December 31, 1990, Monday, Home Edition
</P>
</DATE>
<SECTION>
<P>
Metro; Part B; Page 4; Column 3; Letters Desk
</P>
</SECTION>
<LENGTH>
<P>
33 words
</P>
</LENGTH>
<HEADLINE>
<P>
'TAGGER ARREST'
</P>
</HEADLINE>
<TEXT>
<P>
The only way we are ever going to end the nasty, filthy graffiti
problem is to come down hard on the idiots doing it. I would be happy
to contribute to a reward fund.
</P>
<P>
IRV BUSH, Marina del Rey
</P>
</TEXT>
<TYPE>
<P>
Letter to the Editor
</P>
</TYPE>
</DOC>
<DOC>
<DOCNO> LA123190-0002 </DOCNO>
<DOCID> 329362 </DOCID>

```

Figure 4.1: An extract from the LA Times Collection, file name LA123190, and illustrates one document

Some notable characteristics of LA Times files:

1. LA Times files are encoded as SGML (Standard Generalized Markup Language) file format (SGML is a superset of XML file format).
2. The uppercase words enclosed by <> and </> are tag names.
3. Every document is separated by <DOC> and </DOC> tag names.
4. <DOCNO> and </DOCON> tag names enclose the unique name of each document

The following table summarizes the characteristics of the input data set:

Characteristics of the LA Times Data Set (1989, 1990)	
Number of documents	131,321
Size of the Input Data Set	476MB
Average Vocabulary size (approximately)	500
Average document size (approximately)	40 KB
Largest file size	828 KB (LA052089_0101)
Smallest Size	352 Bytes (LA070189_0001)
Number of words in the smallest file	91
Number of words in the largest file	167,045
Number of relevant files (Out of 131,312 files) with respect to TREC-8 query set	1,151

Table 4.1: Input Data Characteristics (LA Times Data Set (1989, 1990))

4.3 Pre-processing of Data

Pre-processing of data is a very important step in doing research in the area of information retrieval. The pre-processing of data encompasses the following steps:

1. Extract documents from the large SGML data file (e.g., LA123190) into individual files for each document.
2. Remove stop words from the input dataset.
3. Apply stemming to the input dataset.

4.3.1 Method to Extract Documents

In the selected corpora every file contains approximately 100 to 130 documents. Each document is separated by a document number that is presented using a special tag (e.g., <DOCNO> LA010189-0001 </DOCNO>). So, in the very first step of the processing of raw data, smaller files are extracted from large files and each smaller file is named using the document tag number e.g., LA010189_0001. After extracting all the documents from TREC-8, 131,321 files are found. The following section presents the automatic TREC file extractor program in better detail:

TREC file extractor is a software system written in PHP. The flow diagram of the software system is depicted in figure 4.2. TREC file extractor takes the large TREC-8 SGML files as input one after another and produces the small individual documents. The

name of the extracted document is the same as the unique name found between the <DOCNO> and </DOCNO> tags. TREC file extractor was tested in Windows Vista and Windows Xp environments. The code developed to extract documents from TREC-8 SGML files is available in appendix B. The output of the TREC file extractor is used as the input of the pre-processing software, which is presented in the figure 4.2.



Figure 4.2: TREC File Extractor

Algorithmic steps of TREC File Extractor software:

1. Read each SGML files (LA123190 is a name of a file) one after another from the input directory.
2. Find each <DOC> and </DOC> tag pair (top-down methodology) as these tag pairs identify individual documents.
3. Extract (read) the content between a <DOC> and </DOC> tag pair including the tags and write the extracted content in a new SGML file as lower case (e.g., change all upper case letters to lower case).
4. Specify a unique name for the new file as “input file name_document number” (e.g., LA123190_0001).
5. Write the new files as text files to the output directory.

4.3.2 Stop Word Removal and Stemming

The IR group, University of Glasgow provides a list of stop words that contains 319 stop words and the IR group, University of Tennessee also provides a list of 439 stop words. In this research, University of Tennessee, University of Glasgow and an extended version of stop word lists were used in our experiments. We call the extended stop word list that we have prepared the UNBC_LAT stop word list. UNBC_LAT stop word list contains 1911 stop words. Algorithmic steps to create this stop word list are given below:

1. Consider the terms to create a stop word list those that have at least frequency 2 (a term must be present in a document at least twice).

2. Create an initial stop word list by combining the stop word list of both IR groups, University of Tennessee and University of Glasgow (no duplication in the list).
3. Remove all the punctuation from the input TREC-8 LA Times dataset.
4. Create a list of terms from the input dataset, in descending order of term frequencies (i.e. the terms with the highest term frequency will be at the top of the list).
5. Manually extract the special items to be added to the initial list (those terms are not already in the initial list) to create an extended stop word list:
 - Add all file names to the initial list as every file contains file names e.g., LA123190.
 - Add all tags names (e.g., <doc>, and <docno>) to the initial stop word list.
 - Add roman numbers to the initial list e.g., xvii.
 - Add scale units e.g., ft, mm, etc.
 - Add all adjectives, adverbs (e.g., ago, bye).
 - Add prefixes from words (e.g., non comes from non-governmental)
 - Add special words (e.g., haven (comes from haven't), doesn (comes from doesn't))
 - Add dates e.g., feb-92, 11-apr
 - Add foreign words as dataset is newspaper articles. Add suspicious words e.g., aaftink, aachen, ora etc
 - Add other words e.g., ext (telephone extension), 19th, z90, v6 (engine)

To create this stop word list, we manually searched the high and low frequency terms out of 132,785 terms in the frequency table at the initial stage. We repeated this in a number of cycles by removing different stop word lists (stop word list of the other IR groups) from the TREC-8 LA Times dataset, and manually searched for the remaining stop words from the frequency table. To search stop words is very time consuming as the dataset as well as the number of terms is also large. The most difficult thing is to choose a word as a stop word.

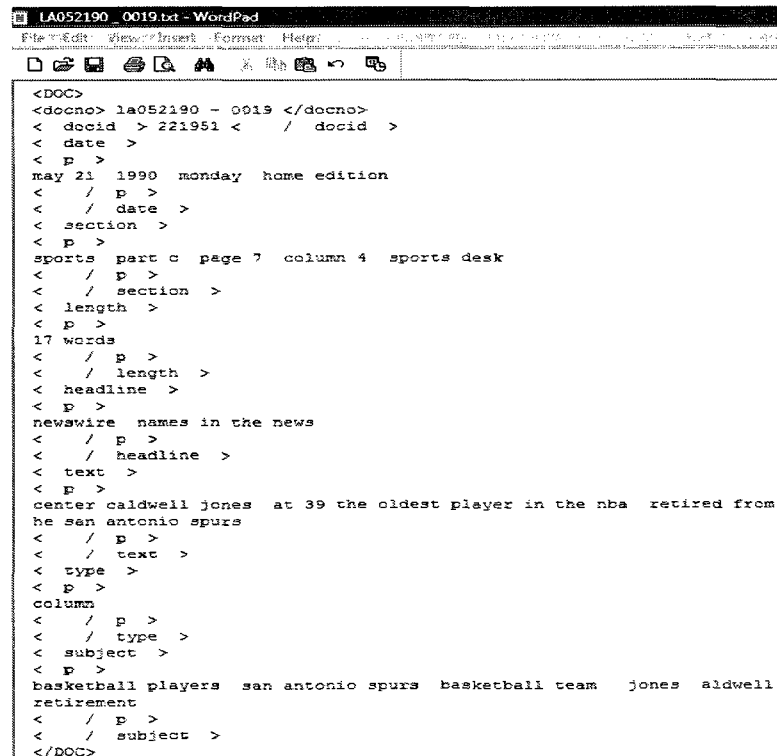
TREC-8 LA Times dataset contains newspaper articles, so, there are variations in the contents. As we started to work with the stop word list of the other IR groups and found that there are many high and low frequency stop words in our terms list (to be used to create a term-by-document matrix), we decided to build our own stop word list for

efficient processing of the input dataset because the existing stop word lists do not include many of the high and low frequency terms in our dataset.

All the stop word lists used in this thesis work are given in Appendix A. After creating the extended stop words list we remove these stop words from the input dataset.

The idea of stemming is introduced in section 1.2.4. In this thesis, we used Porter's stemming on the input dataset to produce stem words. Software developed to remove stop words from the input dataset and to stem the input dataset is discussed in Section 4.2.2.

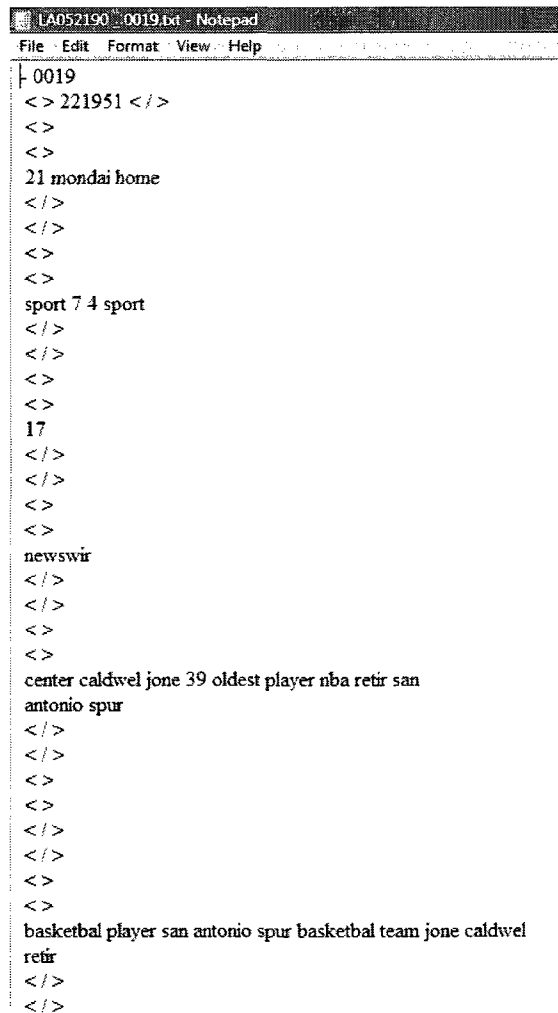
Fig. 4.3 and Fig. 4.4 present a single input file before and after pre-processing respectively. All the stop words (all tags between < >, etc.) are removed from the input file (LA052190_0019) after pre-processing. After removing stop words an input file still contains a few symbols (<, >, /) and numerical values. These special symbols and numerical values will be removed by the LSI system while parsing the input files during creation of the term-by-document matrix.



```
LA052190_0019.txt - WordPad
File Edit View Insert Format Help
<DOC>
<docno> la052190 - 0919 </docno>
< docid > 221951 < / docid >
< date >
< p >
may 21 1990 monday home edition
< / p >
< / date >
< section >
< p >
sports part c page 7 column 4 sports desk
< / p >
< / section >
< length >
< p >
17 words
< / p >
< / length >
< headline >
< p >
newswire names in the news
< / p >
< / headline >
< text >
< p >
center caldwell jones at 39 the oldest player in the nba retired from
he san antonio spurs
< / p >
< / text >
< type >
< p >
column
< / p >
< / type >
< subject >
< p >
basketball players san antonio spurs basketball team jones aldwel
retirement
< / p >
< / subject >
</DOC>
```

Figure 4.3: An Input File before Pre-processing

The following figure represents an input file after pre-processing.



```
LA052190_0019.txt - Notepad
File Edit Format View Help
| 0019
<> 221951 </>
<>
<>
21 mondai home
</>
</>
<>
<>
sport 7 4 sport
</>
</>
<>
<>
17
</>
</>
<>
<>
newswir
</>
</>
<>
<>
center caldwel jone 39 oldest player nba retir san
antonio spur
</>
</>
<>
<>
</>
</>
<>
<>
basketbal player san antonio spur basketbal team jone caldwel
retir
</>
</>
```

Figure 4.4: An Input File after Pre-processing

4.4 Stop Word Removal and Stemming Program

Pre-processing of input data includes *stop word removal* and *stemming*. Concepts of stop words and stemming have been introduced in the sections 2.4 and 2.5 respectively. The following section explains the working procedure of the data pre-processing phase.

The pre-processing software takes input from the output of the TREC file extractor program. Pre-processing software reads each word from each file (from input folder) one after another. It reads every term/word of a file from the beginning, and checks the word to determine whether it is in the stop word list or not. If the word is in the stop word list

then it simply deletes that word and looks for the next word from the same file. If the word is absent from the stop word list then the word is stemmed using Porter's Stemming algorithm [Porter 1997] and the stemmed word is saved in a file (in the output folder) with the same name as the file it has read. Pre-processing software repeats the same process for all input files to discard all the stop words and to produce stemmed output of all the input files. The code developed for the pre-processing software is available in appendix B.

The flow diagram of the pre-processing software system is depicted in the figure 4.5. Pre-processing software has been tested in Windows Vista and Windows Xp environments. The output of the pre-processing software system is the input of the Latent Semantic Indexing (LSI) retrieval system.

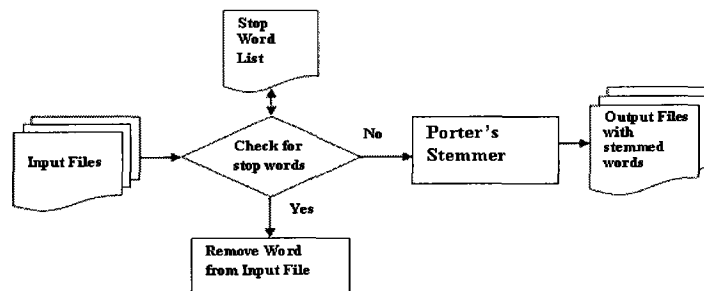


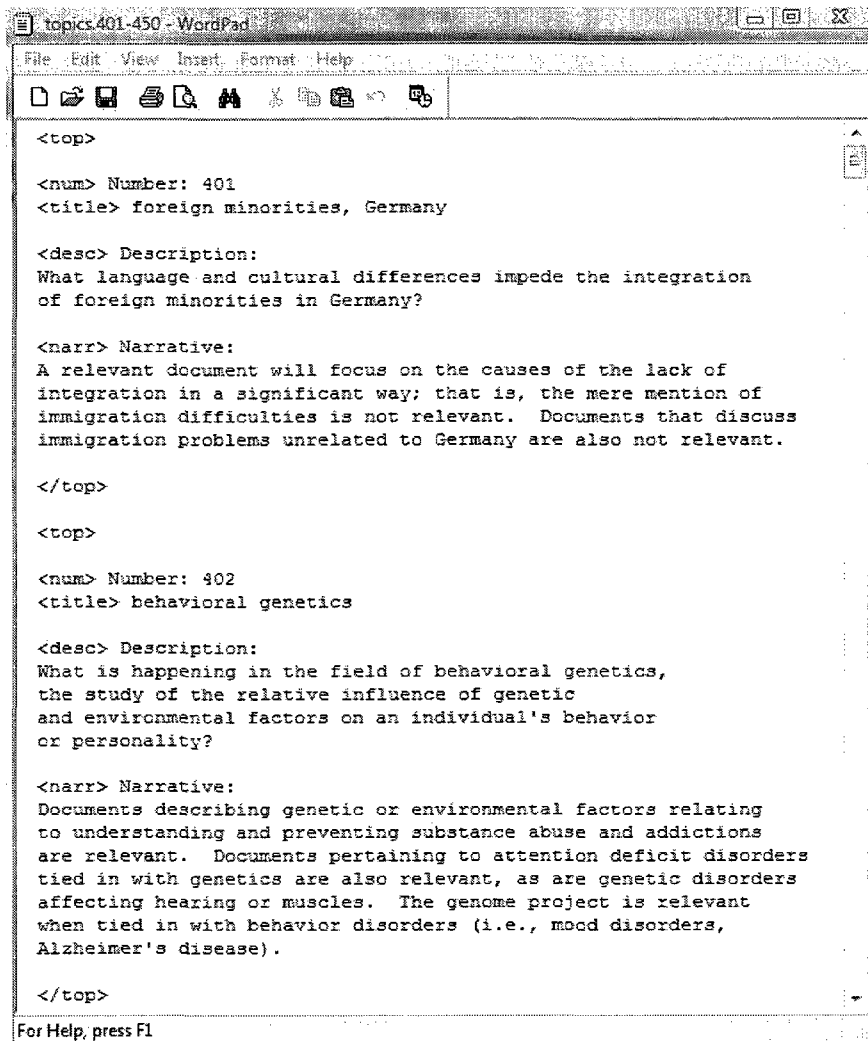
Figure 4.5: Pre-processing Software

Algorithmic steps of stop word removal and stemming software are:

1. Read every input file (LA123231 is a name of a file) one after another from the input directory.
2. Read a term/word from an input file.
3. Search for the word in the stop word list.
4. If the term is not in the stop word list, go to step-5 or if the term is in the stop word list, delete the term and put a NULL (in the position of the deleted word) then go to step-2 to read next word/term.
5. Apply Porter's stemming to create a stemmed word and write the stemmed word in a file with the same name of the file from which the word was read and save the file to the output directory.
6. Repeat steps-2 to step-5 to remove and stem all words from input files.

4.5 Query Selection

In this research work TREC-8 query sets (401 to 450) are used to evaluate the efficiency of the Latent Semantic Indexing (LSI) technique in terms of information retrieval for the mentioned dataset (section 4.2.2). TREC-8 query sets are taken from Text Retrieval Conference (TREC, http://trec.nist.gov/data/topics_eng/index.html) that is maintained by the National Institute of Standards and Technology (NIST). The relevance judgment information also taken from TREC. Fig. 4.6 represents the first two queries from the TREC-8 query set. The characteristics of a TREC-8 query: Every TREC-8 query starts with the tag <top> and ends with </top>. <num>: num tag presents the query number, in Fig 4.6 query numbers are 401 and 402. <desc>: description tag describes the actual query. The text for query number 401 is “*What language and cultural differences impede the integration of foreign minorities in Germany*”? <narr>: narrative tag contains the information for the human assessors to judge the relevance of documents retrieved by this query.



```
<top>

<num> Number: 401
<title> foreign minorities, Germany

<desc> Description:
What language and cultural differences impede the integration
of foreign minorities in Germany?

<narr> Narrative:
A relevant document will focus on the causes of the lack of
integration in a significant way; that is, the mere mention of
immigration difficulties is not relevant. Documents that discuss
immigration problems unrelated to Germany are also not relevant.

</top>

<top>

<num> Number: 402
<title> behavioral genetics

<desc> Description:
What is happening in the field of behavioral genetics,
the study of the relative influence of genetic
and environmental factors on an individual's behavior
or personality?

<narr> Narrative:
Documents describing genetic or environmental factors relating
to understanding and preventing substance abuse and addictions
are relevant. Documents pertaining to attention deficit disorders
tied in with genetics are also relevant, as are genetic disorders
affecting hearing or muscles. The genome project is relevant
when tied in with behavior disorders (i.e., mood disorders,
Alzheimer's disease).

</top>

For Help, press F1
```

Figure 4.6: First Two Queries (401 and 402) from the TREC-8 Query Set

4.7 Conclusion

This chapter discussed the idea and characteristics of test collection. The advantage of the standard test collection is that it contains human evaluation of relevance judgment which can be used in the evaluation of the LSI based IR system to be tested. Pre-processing of the input dataset and the program developed for pre-processing are also covered. The next chapter describes the experiments carried out after the pre-processing of the input dataset.

Chapter-5

Experimental Studies on Stop Word Lists and Term Weighting Schemes

5.1 Introduction

This chapter describes the experiments that have been conducted using the Latent Semantic Indexing (LSI) based information retrieval system. The basic research procedures are demonstrated in this chapter. The experimental results as well as their evaluations are also covered in the following sections.

5.2 GTP (General Text Parser)

The LSI based retrieval system used for the experiments is called the GTP (General Text Parser) [Giles et al 2001]. GTP is a general purpose text parser with a matrix decomposition option which can be used for generating vector space IR models. GTP is an open source tool developed by a group in the Department of Computer Science, University of Tennessee. The specific vector-space model exploited by GTP is Latent Semantic Indexing (LSI). The architecture of the GTP software is depicted in the Figure 5.1. The pre-processed (with stemming and stop word removal) data is submitted to the GTP processor for further processing.

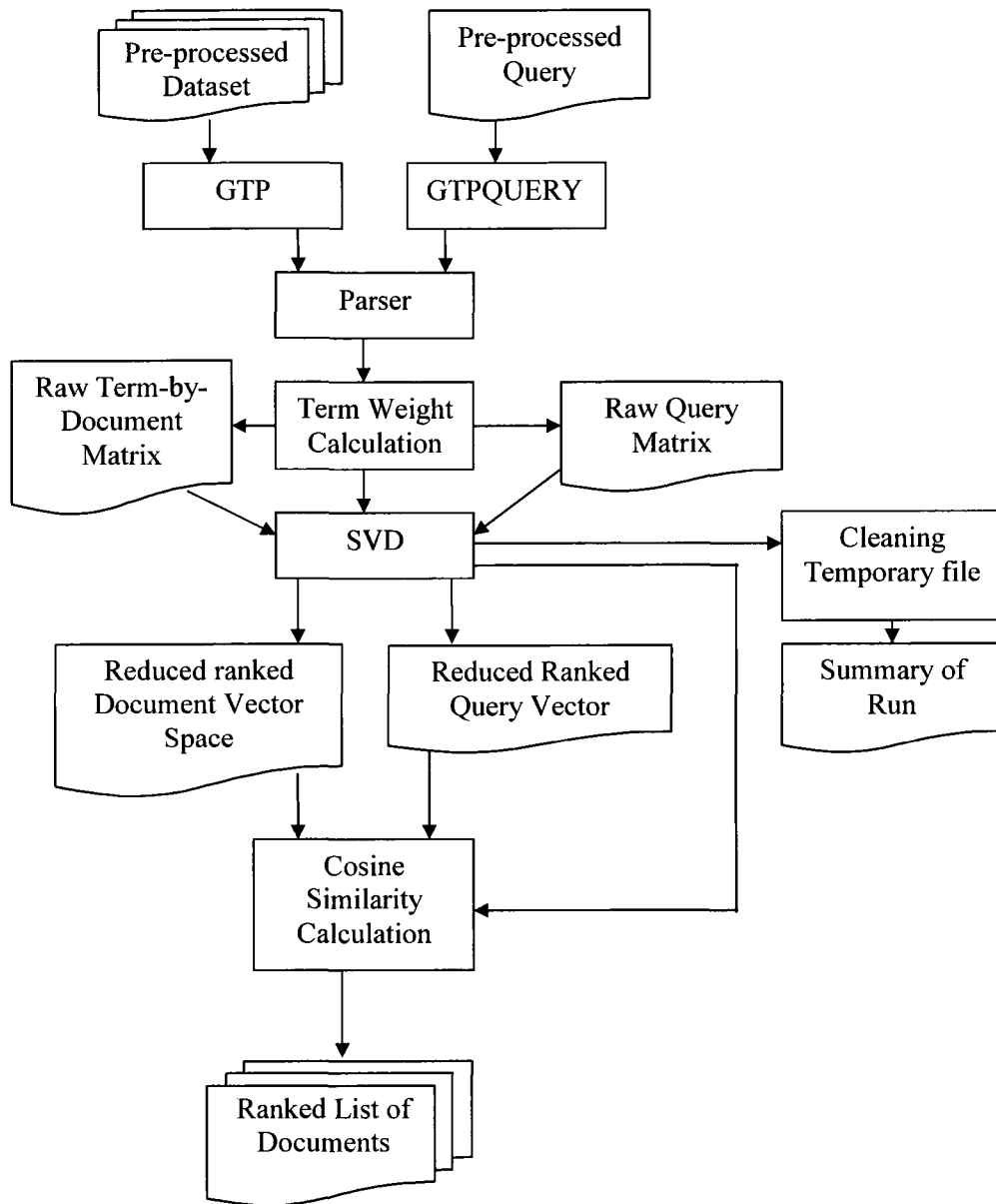


Figure 5.1: Latent Semantic Indexing (LSI) Retrieval Systems [GILES ET AL 2001]

GTP takes pre-processed text files as input and parses the files to create a database of terms/keys/words and their corresponding weights to create a raw term-by-document matrix, and then LSI performs Singular Value decomposition (SVD) on that raw matrix to produce the low-rank vector space in a binary file. Finally, LSI cleans up temporary working files and writes the summary of the run to the RUN_SUMMARY file.

GTPQUERY takes a query file as input and extracts the terms/keywords from the database and the corresponding weights of each term to create a raw query vector. Then, SVD creates a reduced ranked query vector.

Once the query vector is generated, a cosine similarity calculation is performed between the document vectors and the query vector to create a ranked list of documents based on the similarity values. The results (ranked list of documents) are written to a file and the name of the file takes the form q_result.#, where # is the query number. The results written in these files are sorted by the most relevant to least relevant.

5.3 Experimental Design

The following sections discuss the experiments done for this thesis. The experiments are performed to determine the answers to the research questions (Chapter-1, Section 1.3) associated with stop word lists and term weighting issues in LSI based information retrieval (IR) for the TREC-8 LA Times dataset (1989-1990).

5.3.1 Experiment-1: Study of Stop Words

This study tries to find out the effect of “stop words/Common words” on a text based LSI information retrieval (IR) process. Evidence will be developed to indicate the most effective stop word lists for LSI based ad hoc IR processes.

By surveying the literature, it was found that different research groups use their own stop word lists and these lists vary in the number of stop words they contain. For example, Information Retrieval Group, University of Glasgow (319 stop words) and University of Tennessee (439 stop words) have their own stop word lists. TREC encourages text retrieval research; however, TREC does not recommend any standard stop word list for IR based research. Christopher D. Manning [Manning et al 2008], in his book mentioned a brief way to prepare the list of stop words. The process is "*the general strategy for determining a stop list is to sort the terms by collection frequency (the total number of times each term appears in the document collection), and then to take the most frequent terms, often hand-filtered for their semantic content relative to the domain of the documents being indexed, as a stop list, the members of which are then discarded during indexing.*" By following the above idea in this thesis, an extended stop word (also called UNBC_LAT stop word list) list (please see Section 4.3.2) is used that includes University of Glasgow and University of Tennessee stop word lists, 730 TREC file names (input

data set), 22 tag names (e.g., doc, docno etc) and other words (e.g., alpha numeric words, roman numbers etc.). The total length of the UNBC_LAT stop word list is 1911.

The following table lists the various inputs and parameters used in this study.

Data Set	Stop Word List	Stemming	Weighting Scheme	Number of Queries
TREC-8 Los Angeles Times (LA Times)	University of Glasgow	Porter's Stemming	tf-idf	50 Queries for each experiment
	University of Tennessee			
	UNBC_LAT stop word List			
	Without removing stop words			

Table 5.1: Different Parameters for the Study of Stop Words

5.3.2 Experiment-2: Study of Weighting Schemes

By surveying the literature of text based information retrieval, it was found that research suggested the use of log-entropy [Dumais 1991] or tf-idf [Manning et al 2008] (term frequency- inverse document frequency) weighting schemes because those result in better retrieval performance over idf and raw term frequency. We also studied raw term frequency weighting scheme for our research interest.

The objective of the following three experiments is to study the two most popular weighting schemes, tf-idf and log-entropy, to find which weighting scheme is effective on a large data set on LSI based ad hoc text based IR system.

The following table lists the various inputs and parameters used in this study.

Data Set	Stop Word List	Stemming	Weighting Scheme	Number of Queries
TREC-8 Los Angeles Times (LA Times)	The Best Stop Word List from Experiment -1	Porter's Stemming	tf-idf	50 Queries for each Experiment
			log-entropy	
			raw term frequency	

Table 5.2: Different Parameters for the Study of Weighting Schemes

5.4 Results and Evaluations

The following sections present the findings and the evaluations of the experiments.

5.4.1 Result: Experiment-1

Table 5.3 shows the 10-point interpolated precision of four different systems. The recall precision graph based on the Table 5.3 data is shown in Figure 5.2.

10-Points Recall	PR_mylist_tfidf (System-1)	PR_Stem_tfidf (System-2)	PR_Ten_tfidf (System-3)	PR_Glasgow_tfidf (System-4)
0.1	0.1757	0.1385	0.1513	0.1221
0.2	0.1108	0.1058	0.0994	0.0864
0.3	0.0888	0.0841	0.0735	0.0799
0.4	0.0724	0.0743	0.0693	0.0722
0.5	0.0678	0.0710	0.0672	0.0694
0.6	0.0659	0.0662	0.0647	0.0671
0.7	0.0646	0.0632	0.0609	0.0660
0.8	0.0621	0.0592	0.0588	0.0628
0.9	0.0554	0.0553	0.0489	0.0578
1.0	0.0436	0.0247	0.0316	0.0374

Table 5.3: 10–point Interpolated Precision of Four Different Systems

In the Table 5.3, the recall value of 0.1 represents the top 10% of the documents (in the collection) which are relevant to a query set. As an example using PR_mylist_tfidf the precision associated with top 10% of the documents is 0.1757 or 17.57%. This value is calculated by interpolating the precision values of all 50 queries used for this thesis at the standard recall value 0.1. Details of the interpolation technique can be found in [Manning et al 2008].

We can compare retrieval systems with different parameters (e.g., stop word list) in terms of precision in different standard recall points e.g., 0.1, 0.2 etc. For example, at recall point 0.3 means, (top ranked 30% documents) the precision values for system-1(PR_mylist_tfidf) is 8.88% and for system-3(PR_ten_tfidf) is 7.35%. So, if we subtract $(8.88-7.35) \%$ =1.53%, means that system-1 shows 1.53% better retrieval performance than sytem-2 retrieval system for the top 30% retrieved documents. If we look at the Figure 5.2 at the point of recall 0.3, we can see the difference.

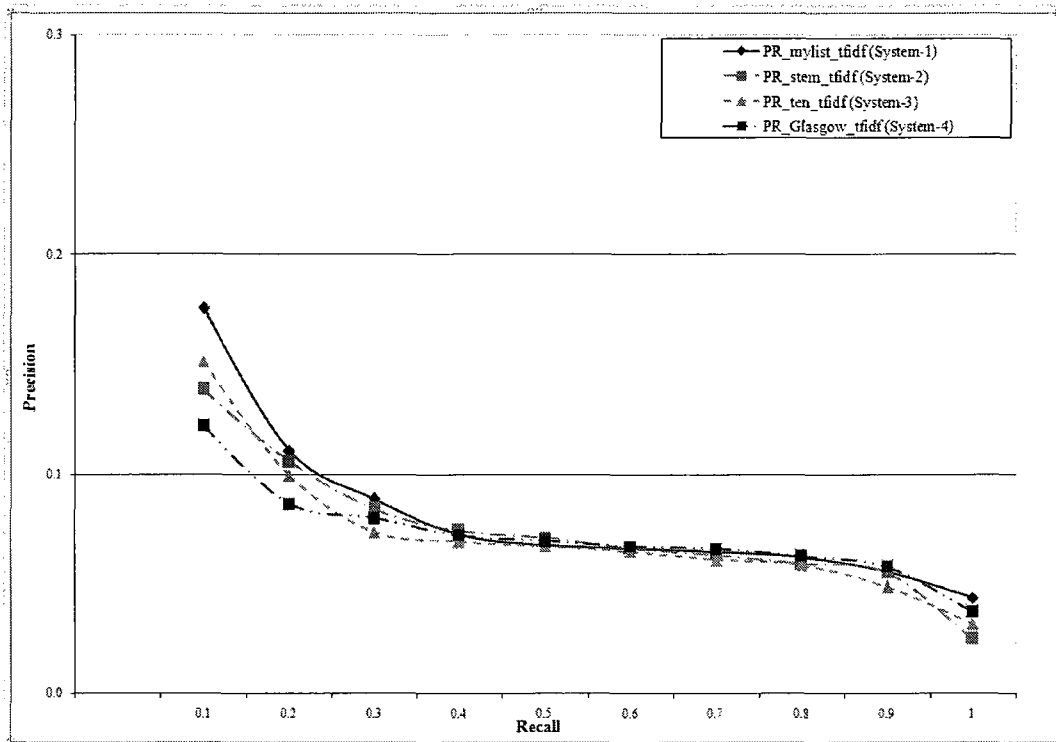


Figure 5.2: Recall-Precision Graph for Experiment-1

The system-1 (mylist_tfidf) with extended stop word list (mylist means UNBC_LAT stop word list) provides the best result when compared to the other 3 systems. For the top 10% retrieval, system-1 shows 5.37% better retrieval performance than system-4 (Glasgow means University of Glasgow stop word list), 3.68% better retrieval performance than system-2 (stem_tfidf, stem means without removing stop word list), and 2.44% better retrieval performance than system-3. However, after top 40% retrieval all the systems showed almost the same retrieval performance.

In the system-2 (stem_tfidf), we just applied Porter's stemming without removing stop words. It is pretty interesting that the retrieval performance of system-2 compared to system-4 (glasgow_tfidf) is 1.64% better.

From the above results, it is found that, in case of LSI based ah hoc information retrieval, the set of stop words is dependent on the set of input data. However, if someone has a powerful computer then one can expect significant retrieval performance without removing stop words. From the above result is it clear that the use of an arbitrary set of

stop words reduces retrieval performance in case of LSI-based ah hoc information retrieval with large dataset.

5.4.1.1 Paired Sample T-test

The idea of a t-test is introduced in section 2.8. We calculated the t-test using software SPSS 17.0. Table-5.4 represents recall-precision of four different systems and means and standard deviations of precision. Table 5.5 shows output of the t-test (detail step by step calculations are available in Appendix D). We compared system-1 with three other systems in terms of the level of significance. We found 3 different level of significance for every pair of systems. By analyzing the significance level (see Table-C in Appendix D) it is found that the t-test produced a result that is below the threshold level. This is an interesting finding in that if we apply paired sample t-test to compare different retrieval systems in terms of level of significance, paired sample t-test results indicate insignificant differences for all cases.

10-Points Recall	PR_mylist_tfidf (System-1)	PR_Stem_tfidf (System-2)	PR_Ten_tfidf (System-3)	PR_Glasgow_tfidf (System-4)
0.1	0.1757	0.1385	0.1513	0.1221
0.2	0.1108	0.1058	0.0994	0.0864
0.3	0.0888	0.0841	0.0735	0.0799
0.4	0.0724	0.0743	0.0693	0.0722
0.5	0.0678	0.0710	0.0672	0.0694
0.6	0.0659	0.0662	0.0647	0.0671
0.7	0.0646	0.0632	0.0609	0.0660
0.8	0.0621	0.0592	0.0588	0.0628
0.9	0.0554	0.0553	0.0489	0.0578
1.0	0.0436	0.0247	0.0316	0.0374
Mean	0.0807	0.0742	0.0726	0.0721
Standard Deviation	0.0381	0.0307	0.0326	0.0219

Table 5.4: 10-point Interpolated Precision, Mean and Standard Deviation of Four Systems

Paired Samples Test						
		Paired Differences		t	Degree of freedom (N-1)	Level of Significance
		95% Confidence Interval of the Difference				
		Lower	Upper			
Pair 1	System1 - System2	-.0024183	.0153783	1.647	9	.134
Pair 2	System1 - System3	.0027259	.0135741	3.399	9	.008
Pair 3	System1 - System4	-.0041491	.0213491	1.526	9	.161

Table 5.5: Comparison among Systems in Terms of T-test

5.4.2 Result: Experiment-2

Table 5.6 shows the 10–point interpolated precision of three different systems. The recall precision graph based on the Table 5.6 data is shown in Figure 5.3.

10-Points Recall	PR_mylist_tfidf (System – 1)	PR_mylist_loentropy (System – 5)	PR_mylist_raw_freq (System – 6)
0.1	0.1757	0.1464	0.0910
0.2	0.1108	0.1061	0.0874
0.3	0.0888	0.0803	0.0789
0.4	0.0724	0.0784	0.0720
0.5	0.0678	0.0743	0.0690
0.6	0.0659	0.0714	0.0636
0.7	0.0646	0.0688	0.0626
0.8	0.0621	0.0666	0.0596
0.9	0.0554	0.0515	0.0569
1.0	0.0436	0.0388	0.0339

Table 5.6: 10–point Interpolated Precision of Three Different Systems

The objective of this study was to find the right term weighting scheme for LSI based retrieval system for a large dataset. From the results presented in Table 5.6, it is found that in case of top 10% retrieved documents, system-1 with tf-idf term weighting scheme showed 17.57% and system-6 with raw term frequency showed 9.1% retrieval performance. So if you subtract $17.57 - 9.1 = 8.47 \cong 9\%$, means system-1 showed 9% better retrieval performance than system-6. Similarly, log-entropy (system-5) showed 5.54% better retrieval performance than raw term frequency (system-6). On the other hand, tf-idf (system-1) showed $2.93 \cong 3\%$ better retrieval performance than system-5 with the log-entropy term weighting scheme.

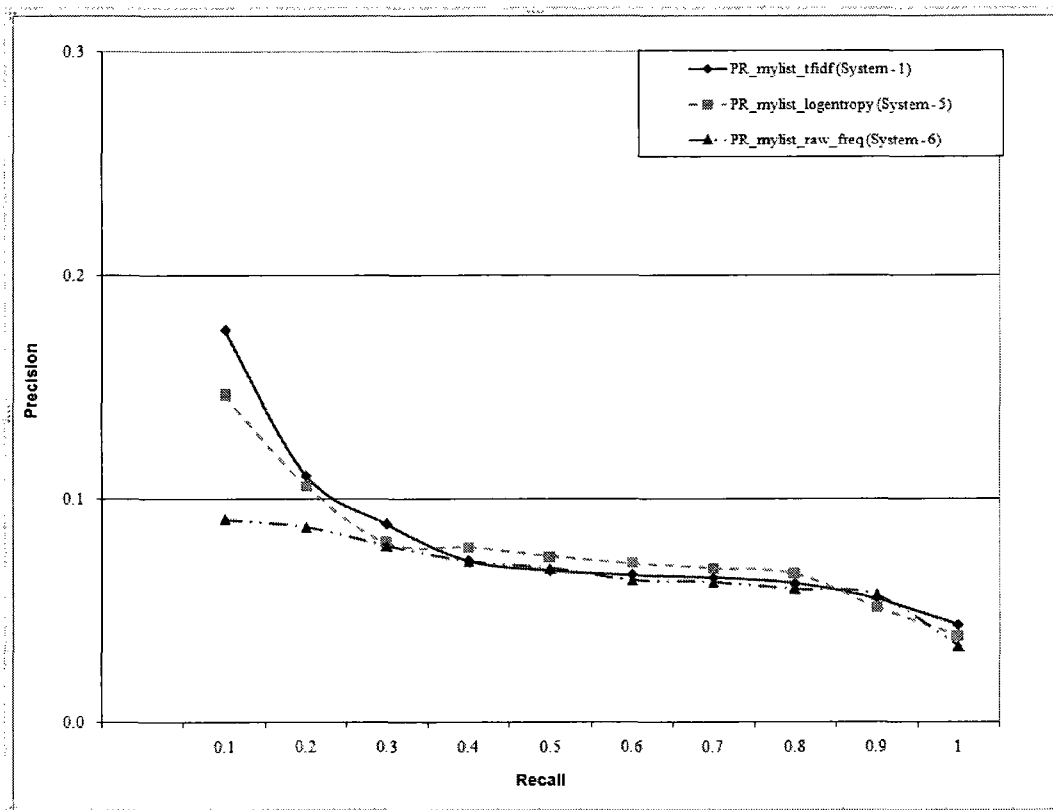


Figure 5.3: Recall-Precision Graph for Experiment-2

If we look at Figure 5.3 in the recall-precision graph at the recall level 0.1, we can see the above results in a graphical form. So, in case of a large dataset tf-idf performs better than log-entropy and raw term frequency weighting schemes. However, after top 40% retrieval (in Figure 5.3 please look at the recall values 0.4 to 1.0) all the retrieval systems showed almost the same retrieval performance. As 1151 documents are relevant out of 131,321 documents (only 0.88% documents are relevant in this test collection) in LA Times TREC-8 standard test collection, we suspect that almost all relevant documents are retrieved in the top ranked 30% retrieved documents, that's why after 40% retrieved documents every retrieval system showed equal performance as shown in the Figures 5.2 and 5.3.

5.5 Conclusion

In the above two studies (section 5.4.1 and 5.4.2), we performed seven sets of experiments, and used 50 queries (TREC-8) to evaluate the results of the experiments. 10-point recall-precision graphs, and a t-test have been used to measure the retrieval

performance of LSI for large test collection. By performing these experiments we found the following research results:

- Different input datasets contain different contents (e.g., news paper articles, medical journals etc), so, the term list of an input dataset also is different for the other datasets. It is clear that for any unique document collection a tailored stop word list must be assembled.
- In the existing published results on the LSI based retrieval system there was no concrete evidence about which term weighting scheme works better in case of large datasets. In our research, we conclude that the result (in numerical forms) shows tf-idf term weighting performs better than log-entropy and raw term frequency weighting schemes when the test collection becomes large.
- Our findings on LSI based retrieval system with large dataset extend the earlier published results discussed in chapter-3 by providing concrete evidence of relevance performance.
- The t-test failed to distinguish among the retrieval performance of different retrieval systems used in this research work.

LSI showed promising results in case of TREC-8 LA Times dataset. TREC-8 LA Times dataset is an example of a large unstructured digital textual dataset. So, we can benefit from using LSI to find information from large digital unstructured textual archives.

Chapter-6

Conclusion and Future Directions

6.1 Introduction

In this chapter, we present a summary of this thesis as well as mention possible directions of our future research.

6.2 Summary

The field of Information Retrieval (IR) was born in the 1950s and over the last forty years, the field has matured considerably. Several IR systems are used on an everyday basis by a wide variety of users. [Singhal 2008] Retrieving information from digital archives is a challenge to the users, and an automatic IR system is an obvious solution to this challenge.

In Chapter-1, we introduced the fundamental terms of LSI research and presented the problem statement.

In Chapter-2, we discussed the theories of LSI based IR system and related matters. This chapter covers the term weighting schemes, ranked based similarity (between a query and documents) measurement technique, singular value decomposition, TREC-8 relevance judgment, and the idea of recall-precision and t-test. We discussed TREC-8 relevance judgment and the recall-precision graph because those help us to evaluate the retrieval results.

In Chapter-3, we surveyed the published results on LSI for last 20 years that helped us to identify the potential research questions on LSI based IR system.

In Chapter-4, we described the characteristics of standard large TREC-8 test collection, and the pre-processing of the selected input dataset. To address the research questions on a large dataset, we created and used our own stop words lists, Porter's stemming, and LSI based vector space model (singular value decomposition), for a structured set of experiments.

In Chapter-5, we presented the experimental design, experimental results, and the list of findings from the experimented results. All the results are based on the top 10% retrieval of documents. We performed two different experiments to judge the performance of LSI in case of large dataset. The experiments are given below briefly:

- Study the effect of different stop word lists.
- Study the effect of different term-weighting schemes.

In the first experiment, we tested the retrieval performance of LSI using 3 different stop word lists and without removing stop words from the input dataset. The stop word list (UNBC_LAT stop word list) we created and used in LSI showed 5.37% better performance than University of Glasgow stop word list, 2.44% better than the University of Tennessee stop word list, and 4% better than without removing stop word list.

In the second experiment, we tested three different term-weighting schemes (raw term frequency, tf-idf, and log-entropy) on a large dataset. tf-idf weighting schemes showed 9% better retrieval performance, and log-entropy showed 5.54% better performance than raw term frequency. On the other hand, tf-idf showed 3% better retrieval performance than log-entropy. In summary, tf-idf showed better performance than log-entropy weighting and raw term frequency schemes in case of large dataset.

In general, LSI has a number of advantages to process large dataset as follows:

1. Both terms and documents are explicitly represented in the same space.
2. Queries and new documents can easily be added.
3. LSI uses SVD to reduce dimension and to remove noise.
4. LSI is able to handle polysemy and synonymy.

Thus, LSI is able to represent and manipulate large data sets, making it viable for real-world applications. [Deerwester et al 1990] Our results on LSI with large dataset also proved that LSI is able to handle large dataset.

6.3 Possible Extensions of the Experiments

Latent Semantic Indexing (LSI) is being used in a variety of information retrieval and text processing applications, although its primary application has been for conceptual text

retrieval and automated document categorization. [Dumais 2004] In this thesis we studied the accuracy of LSI in case of large data set for document retrieval.

Some additional research works related to this study might include the following:

1. By analyzing the results [section 4.3.1 and 4.3.2] of this research, it is found that every experiment shows almost equal performance at the recall points 0.4 to 1.0. As only 0.88% documents are relevant, we suspect that almost all relevant documents are retrieved in 30% top ranked documents; meaning that the graphs show almost equal retrieval performance after recall point 0.4. This suspicion might be an interesting study to find the reason(s) behind this equal performance in this dataset. This suspicion could also be explored for other large dataset.
2. Term-weighting schemes are very influential in LSI based IR systems. One possible study that might be interesting is to apply a position based term weighting scheme (sometimes called positional indexing). This weighting scheme is very costly in terms of computing time, but may produce better retrieval performance. We plan to work on position based term-weighting schemes in the future.
3. The General Text Parser [GTP] [Giles et al 2001] used in this research has a command line interface. Adding Graphical User Interface (GUI) could enhance user interaction with the GTP. A java version of GTP is available with GUI; however, it has memory limitation that prevents it from processing a large dataset.

Bibliography

- [Al-Maskari et al 2008] Azzah Al-Maskari, Mark Sanderson, and Paul Clough. *Relevance Judgments between TREC and Non-TREC Assessors*. SIGIR' 08, Singapore, July 20-24, 2008.
- [Anderson et al 2009] Anderson, David Ray, Sweeney, Dennis J., and Williams, Thomas Arthur (2009). *Statistics for business and economics*. 11th ed. Cengage Learning Inc. lorence, KY
- [Baeza-Yates & Ribeiro-Neto 1999] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley Longman Publishing Co. Inc., first edition, 1999.
- [Berry & Dumais 2008] Berry Michael W. & Dumais Susan. *Latent Semantic Indexing Web Site*. <http://www.cs.utk.edu/~lsi/>, access time: Jan 2008.
- [Berry et al 1993] M. Berry, T. Do, G. O'Brien, V. Krishna, and S. Varadhan. *Svdpackc (version 1.0) user's guide*. Technical Report No. CS-93-194, University of Tennessee, May 1993.
- [Chakravarthy & Netserf 1995] A. Chakravarthy and K. Haase. Netserf. *Using semantic knowledge to find internet archives*. In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, 1995. Pp. 4–11.
- [Chen et al 2001] L. Chen, N. Tokuda, and A. Nagai. *Probabilistic information retrieval method based on differential latent semantic index space*. IEICE Trans. on Information and Systems, E84-D (7):910–914, 2001.
- [Chen et al 2003] L. Chen, N. Tokuda, and A. Nagai. *A new differential lsi space-based probabilistic document classifier*. Information Processing Letters, 88:203– 212, 2003.
- [Chen et al 2004] L. Chen, J. Zeng, and J. Pei. *Classifying noisy and incomplete medical data by a differential latent semantic indexing approach*. In Data Mining in Biomedicine. Springer Press, 2004.
- [Chen et al 2005] L. Chen, J. Zeng, and N. Tokuda. *A "stereo" document representation for textual information retrieval*. Journal of the American Society for Information Science and Technology, 2005.
- [Chen et al 2006] Liang Chen, Jia Zeng and Naoyuki Tokuda. *A "Stereo" Document Representation for Textual Information Retrieval*. Journal of the American Society for Information

Science and Technology (JASIST). 57:6, pp. 768--774. 2006.

- [Chen et al 1999] Liang Chen, Naoyuki Tokuda and Akira Nagai, *Probabilistic Information Retrieval Method Based on Differential Latent Semantic Index Space*. IEICE Trans. Fundamentals, Vol.E82, No.1 January 1999.
- [Coefficient of Variation] http://www.graphpad.com/help/prism5/prism5help.html?coefficient_of_variation_%28cv%29.htm. (Access time Dec 2009)
- [Cohen & Hirsh 1998] W. Cohen and H. Hirsh. *Text categorization using whirl*. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, pages 169–173, New York, 1988.
- [Cooper & Dabney 1992] W. Cooper, F. Gey, and D. Dabney. *Probabilistic retrieval based on staged logistic regression*. In Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 198–210, Copenhagen, Denmark, 1992.
- [Croft et al 2009] W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Information Retrieval in Practice*. Addison Wesley, 2009.
- [Deerwester et al 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. A. *Indexing by latent semantic analysis*. Journal of the American Society for Information Science, 1990, 41(6), pp. 391-407.
- [Downing & Clark 2003] Downing, Douglas and Clark, Jeffrey (2003). Business Statistics. 4th ed. Barron's Educational Series, Inc. Hauppauge, NY
- [Dumais et al 1988] Dumais S. T., Furnas, G. W., Landauer, T. K. and Deerwester, S. *Using latent semantic analysis to improve access to textual information*. In Proceedings of CHI'88: Conference on Human Factors in Computing, New York: ACM, 1988. pp. 281-285.
- [Dumais et al 1997] Susan T. Dumais, Todd A. Letsche, Michael L. Littman, Thomas K. Landauer. *Automatic Cross-Language Retrieval Using Latent Semantic Indexing*. AAAI-97 Spring Symposium Series: Cross- Language Text and Speech Retrieval, 1997. Stanford University, pp. 18-24.
- [Dumais et al 1996] Susan T Dumais, Thomas K Landauer, Michael L Littman. *Automatic Cross Linguistic Information Retrieval using Latent Semantic Indexing*. SIGIR 1996.

- [Dumais 1991] S. Dumais. *Improving the retrieval of information from external sources*. Behavior Research Methods, Instruments, & Computers, 23(2):229--236, 1991.
- [Dumais 2004] Dumais, S. *Latent Semantic Analysis*. ARIST Review of Information Science and Technology. vol. 38, 2004, Chapter 4.
- [Dumais 1992] Dumais, S. T. *Enhancing performance in LSI (Latent Semantic Indexing) Retrieval*. Technical memo, 1992.
- [Fisher & Yates 1995] R.A. Fisher and F. Yates. 6th ed. *Statistical tables for biological, agricultural and medical research*. London: Longman Group, 1995.
- [Foltz 1990] P. W. Foltz. *Using Latent Semantic Indexing for Information Filtering*. In R. B. Allen (Ed.) Proceedings of the Conference on Office Information Systems, Cambridge, MA, 1990. pp. 40-47.
- [Frakes & Yates 1992] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, 1992.
- [Gansterer et al 2007] Wilfried Gansterer, Andreas Janecek, and Robert Neumayer. *Spam filtering based on latent semantic indexing*. 5th SIAM Workshop on Text Mining, Minneapolis, MN, USA, 2007.
- [Greengrass 2001] E. Greengrass. *Information retrieval: A survey*. Technical Report DOD Technical Report TR-R52-008-001, 2001.
- [Grei et al 1997] W. Grei, W. Croft, and H. Turtle. *Computationally tractable probabilistic modeling of Boolean operators*. In Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1997. pp. 119–128.
- [Giles et al 2001] Justin T. Giles, Ling Wo, Michael W. Berry. *GTP (General Text Parser) Software for Text Mining*. CRC Press 2001.
- [Harman 1992] D. Harman. *User-friendly systems instead of user-friendly front-ends*. Journal of the American Society for Information Science, 43(2):164–174, 1992.
- [Harman 1995] D. Harman. *Overview of the third text retrieval conference*. Third Text REtrieval Conference (TREC-3), pp. 1–19. National Institute of Standards and Technology Special Publication 500-207, 1995.

- [Harter 1986] S. Harter. *Online Information Retrieval: Concepts, Principles, and Techniques*. Academic Press, Inc., 1986.
- [Hofmann 1999] T. Hofmann. *Probabilistic latent semantic indexing*. 22nd Annual ACM Conference on Research and Development in Information Retrieval, pp. 50–57, Berkeley, California, 1999.
- [Hull 1994] D. Hull. *Improving text retrieval for the routing problem using latent semantic indexing*. 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 282–291, 1994.
- [Husbands et al 2001] Husbands, P., Simon, H., & Ding, C. *On the use of singular value decomposition for text retrieval*. In *Computational Information Retrieval (SIAM)* (pp. 45–156), Philadelphia, PA, 2001.
- [Kumar et al 2006] Ch. Aswani Kumar, Ankush Gupta, Mahmooda Batool and Shagun Trehan. *Latent Semantic Indexing-Based Intelligent Information Retrieval System for Digital Libraries*. *Journal of Computing and Information Technology - CIT 14, India 2006*, pp. 191–196
- [Labský et al 2005-A] Labský M., Vacura M., and Praks P. *Web Image Classification for Information Extraction*. First International Workshop on Representation and Analysis of Web Space (RAWS-05), Prague, 2005; pp. 55-62;
- [Labský et al 2005-B] Martin Labský, Vojtěch Svátek and Ondřej Šváb. *Information Extraction from HTML Product Catalogues: from Source Code and Images to RDF*. The 2005 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 401-404, Washington, USA, 2005
- [Lee 1994] J. Lee. *Properties of extended Boolean models in information retrieval*. 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 182–190, 1994.
- [Letsche 1996] Todd A. Letsche. *Toward Large-Scale Information Retrieval Using Latent Semantic Indexing*. A masters thesis supervised by Dr. Michael Berry, The University of Tennessee, Knoxville, USA, 1996.
- [Letsche & Berry 2009] Todd A. Letsche and Michael W. Berry. *Large-Scale Information retrieval with Latent Semantic Indexing*. <http://www.cs.utk.edu/~berry/lsi++/>, access time: Feb 2009

- [Liddy et al 1994] E. Liddy, W. Paik, and E. Yu. *Text categorization for multiple users based on semantic features from a machine-readable dictionary*. ACM Transactions on Information Systems, 12(3):278–295, 1994.
- [Lynn 2002] Patrick A. Lynn. *Evolving the General Text Parser (GTP) Utility into a Usable Application Via Interface Design*. A Thesis Presented for the Master of Science Degree, The University of Tennessee, Knoxville By, December 2002
- [Manning et al 2008] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. *An Introduction to Information Retrieval*. Cambridge University Press Cambridge, 2008.
- [Mayfield & McNamee 1997] James Mayfield and Paul McNamee. N-grams vs. words as indexing terms. In TREC-6 Conference Notebook Papers, 1997.
- [Montague 2002] M. Montague. Metasearch. *Data fusion for document retrieval*. PhD thesis, Dartmouth College, Hanover, New Hampshire, 2002.
- [Machala et al 2006] Praks P., Machala L., Snášel V., Strížík M. *Intelligent Information retrieval using numerical linear algebra and applications*. International Symposium GIS Ostrava 2006 Proceedings. Ed. J. Růžicka VSB-TU Ostrava, January 23-25, 2006. pp. 1-13 (CD-ROM).
- [Ogilvie & Callan 2003] P. Ogilvie and J. Callan. *Combining document representations for known item search*. Twenty Sixth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 143–150, Toronto, Canada, 2003.
- [Ott & Larsen 1992] Ott, R Lyman and Larson, Richard (1992). *Statistics a tool for the social sciences*. 5th ed. Pws-Kent publishing company, Boston.
- [Paice 1984] C. Paice. *Soft evaluation of Boolean search queries in information retrieval systems*. Butterworth Publishers Stoneham, MA, USA, pp.33–42, 1984.
- [Pearce & Nicholas 1996] C. Pearce and C. Nicholas. *Telltale: Experiments in a dynamic hypertext environment for degraded and multilingual data*. Journal of the American Society for Information Science, 47(4):263–275, 1996.
- [Porter 1980] M. Porter. *An algorithm for suffix stripping*. Program, 14(3):130–137, 1980.

- [Porter 1997] M. Porter. *An algorithm for suffix stripping*. In K. S. Jones and Willett P., editors, *Readings in information retrieval*, pp. 313–316. Morgan Kaufmann Publishers Inc., 1997.
- [Praks et al 2004] Pavel Praks, Libor Machala and Václav Snášel. *On SVD-Free Latent Semantic Indexing for Iris Recognition of Large Databases*. Fifth International Workshop on Multimedia Data Mining (MDM/KDD'04) in conjunction with ACM SIGKDD Tenth International Conference on Knowledge Discovery and Data Mining, pp. 67-71, Seattle, USA, 2004.
- [Praks et al 2003] Praks P., Dvorský J., Snášel V. *Latent Semantic Indexing for Image Retrieval Systems*. SIAM Conference on Applied Linear Algebra, Williamsburg, VA – USA, 2003
- [Qin et al 2005] Bing Qin, Ting Liu, Zhang Yu, Sheng Li. *Research on Multi-Document Summarization Based on Latent Semantic Indexing*. Journal of Harbin Institute of Technology, China 2005, pp. 91-94
- [Rehder et al 1998] B. Rehder, M. Schreiner, M. Wolfe, D. Laham, T. Landauer, and W. Kintsch. *Using latent semantic analysis to assess knowledge: Some technical considerations*. *Discourse Processes*, 25:337–354, 1998.
- [Rijsbergen 1979] C. van Rijsbergen. *Information Retrieval*. Butterworths, London, 2nd edition, 1979.
- [Rilo 1995] E. Rilo. *Little words can make a big difference for text classification*. 18th Annual International ACM SIGIR conference on Research and Development in Information Retrieval, pages 130–136, 1995.
- [Rose et al 1990] K. Rose, E. Gurewitz, and G. Fox. *A deterministic annealing approach to clustering*. *Pattern Recognition Letters* 11, 11:589–594, 1990.
- [Salton 1971] G. Salton. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall, Englewood Clis, New Jersey, 1971.
- [Salton 1989] G. Salton. *Automatic text processing: The transformation, analysis, and retrieval of information by computer*. Addison-Wesley, Reading, MA, 1989.
- [Salton & Buckley 1987] G. Salton and C. Buckley. *Term-weighting approaches in automatic text retrieval*. Technical Report, Cornell University, USA, 1987.

- [Salton & McGill 1983] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill Publishing company, New York, 1983.
- [Saul & Pereira 1997] L. Saul and F. Pereira. *Aggregate and mixed-order Markov models for statistical language processing*. In Claire Cardie and Ralph Weischedel, editors, Second Conference on Empirical Methods in Natural Language Processing, pages 81–89. Association for Computational Linguistics, Somerset, New Jersey, 1997.
- [Schutze & Silverstein 1997] H. Schutze and C. Silverstein. *Projections for efficient document clustering*. In ACM/SIGIR, pages 74–81, 1997.
- [Singhal 2008] Amit Singhal. Modern Information Retrieval: A Brief Overview. Google, Inc. <http://pages.cs.wisc.edu/~anhai/>, access time Jan 2008
- [Smith 2003] Lindsay I. Smith. *A Tutorial on Principal Component Analysis*. Unpublish Document downloaded for the link kybele.psych.cornell.edu/~edelman/Psych-465-Spring-2003/PCA-tutorial.pdf, in the date, 9 Sep 2007
- [Spigel 1988] Spigel, Murray R (1988). *Theory and problem of Statistics*. 2nd ed. Schaum's outline series, McGraw-Hill.
- [Spoerri 2007] Anselm Spoerri. A Visual Tool for Information Retrieval. Ph.D. Research and Thesis at MIT, 1995. <http://www.scils.rutgers.edu/~aspoerri/InfoCrystal/InfoCrystal.htm>, access time: Sep 2007.
- [Strang 1998] G. Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 3rd edition, 1998.
- [SVDA 2008] Singular value decomposition, <http://www.mathworks.com/access/helpdesk/help/techdoc/ref/svd.html>, accessing time: Dec 2008.
- [STLR 2007] Standard Template Library Resources. <http://www.sgi.com/tech/stl/>, accessing time: Dec 2007.
- [Taha & Gosh 1996] I. Taha and J Gosh. Characterization of the Wisconsin breast cancer database using a hybrid symbolic-connectionist system. Technical Report UT-CVISS-TR-97-007, the Computer and Vision Research Center, University of Texas, 1996.
- [Triola 2006] Triola, Mario F (2006). *Elementary Statistics*. 3rd ed. Pearson, Addison-Wesley.

- [TREC] The Text REtrieval Conference (TREC), <http://trec.nist.gov/>
- [Triola 2006] Mario F. Triola. *Elementary Statistics*. 10th edition (2006), Addison Wesley.
- [Turtle & Croft 1991] H. Turtle and W. Croft. *Evaluation of an inference network-based retrieval model*. ACM Transactions on Information Systems, 9(3), 1991.
- [Wegner 2010] Wegner T (2010). Applied Business Statistics. 2nd ed. Juta Academic.
- [Wolfe & Goldman 2003] Wolfe MB, Goldman SR. *Use of latent semantic analysis for predicting psychological phenomena: two issues and proposed solutions*. Behavior Research Methods, Instruments and Computers, 2003, 35(1):22-31.
- [Wiemer-Hastings et al 1999] P. Wiemer-Hastings, K. Wiemer-Hastings, and A. Graesser. *How Latent is Latent Semantic Analysis?* In Proceedings of the 16th International Joint Congress on Artificial Intelligence, pp. 932–937, Morgan Kaufmann, San Francisco, 1999.
- [Wikipedia] Wikipedia, the free encyclopedia. <http://en.wikipedia.org/>
- [Waller & Kraft 1979] W. Waller and D. Kraft. *A mathematical model of a weighted Boolean Retrieval system*. Information Processing and Management, 15:235–245, 1979.
- [Wolberg & Mangasarian 1990] W. Wolberg and O. Mangasarian. *Multisurface method of pattern separation for medical diagnosis applied to breast cytology*. In Proceedings of the National Academy of Sciences, volume 87, 1990. pp. 9193–9196.
- [Wu & Zhou 2002] J. Wu and Z. Zhou. *Face recognition with one training image per person*. Pattern Recognition Letters, 23(14): 1711–1719, 2002.
- [Zeng 2005] Jia Zeng. *Information Retrieval by Multi-Perspective Representation*. Thesis (M.Sc.)--University of Northern British Columbia, 2005.
- [Zimmerman 1991] H. Zimmerman. *Fuzzy Set Theory and its Applications*. Kluwer Academic Publishers, 2nd edition, 1991.

Appendix A

1. Stop Words List: Generated Stop Words List (University of Northern British Columbia)

apr-31	a340s	acoc	amp	b10
aug-44	a35	across	an	b100
dec-66	a36	ad	ana	b12
feb-92	a4	ada	anc	b2
11-apr	a41	adl	and	b210
15-aug	a43	adm	ang	b3
6-dec	a5	adn	anh	b4
a	a6	ae	ani	b5
a1	a7	afc	ann	b52
a10	a8	afi	another	b6
a11	a9	afl	any	b7
a12	aa	after	anyawi	b747
a13	aaa	afterwards	anybody	b8
a14	aaaa	ag	anyhow	b9
a15	aaaah	again	anyone	ba
a16	aable	against	anything	ba2
a18	aachen	agn	anyway	ba3
a19	aad	ago	anywhere	ba4
a2	aaf	ah	ap	ba6
a20	aaftink	ahl	apart	baa
a21	aagla	ahm	appear	baa1
a22	aah	ak	appropriate	baa2
a23	aahed	al	ar	baa3
a24	aahing	ala	are	back
a25	aahts	alf	around	bbdo
a26	aai	all	as	be
a27	aar	allows	asa	became
a28	aas	alm	ash	because
a29	ab	almost	aside	become
a3	aba	alone	aso	becomes
a30	abb	along	associated	becoming
a300	abc	alp	ast	been
a301	abl	already	asu	before
a31	about	also	at	beforehand
a310	above	although	att	behind
a32	abt	always	av	being
a320	ac4	am	ava	below
a320s	aca	ama	available	ben
a321	acc	amc	aw	ber
a33	accordingly	ami	away	beside
a330	ace	amo	awfully	besides
a330s	ach	among	b	best
a340	acn	amongst	b1	better

between	cac	cra	doesn	everybody
beyond	came	crc	doing	everyone
blm	can	currently	don	everything
blmg	cannot	cvj	done	everywhere
bnd	cant	cwl	doo	ew
bo	cause	cy	dosen	ex
boa	causes	cya	down	example
bol	cb	d	downwards	except
boo	cbn	d1	dr	ext
both	cbo	d10	ds	f
bp	cc	d11	du	fl
brief	ccaa	d12	during	faa
bs	ccdc	d13	dy	far
bsd	cch	d14	e	fcc
bta	cct	d15	e1	fe
btu	ce	d16	e5	few
bu	cee	d2	each	fg
but	cellrule	d3	eb	fi
bv	certain	d4	ebb	fifth
by	cfa	d5	ec	fig
bye	ch	d6	ed	first
byline	cha	d7	edit	five
byu	changes	d8	ee	fl
c	chi	d9	eel	flo
c1	chj	date	eg	followed
c10	chp	dateline	eh	following
c11	chr	day	ei	foo
c12	chu	db	eight	for
c13	cio	de	eip	former
c14	ck	dea	either	formerly
c15	clo	ded	ek	forth
c16	cmc	def	el	four
c17	cmdr	described	elk	from
c18	cmv	dh	else	ft
c19	co	di	elsewhere	fu
c2	column	dib	em	further
c20	come	did	en	furthermore
c21	consequentl	didn	enough	g
c22	y	different	eo	g3
c23	contain	dl	ep	g4
c24	containing	dmc	epa	g5
c2h2	contains	do	epp	g6
c3	coq	doc	er	ga
c4	correction	doc	es	gc
c5	correspondi	doc	et	ge
c6	ng	docid	etc	get
c7	cot	docno	ev	gets
c759915	could	docno	even	given
c8	cpa	docno	ever	gives
c9	cpl	does	every	gn

go	him	isn	15th	la012089
gone	himself	it	16	la012090
goo	his	its	17	la012189
good	hither	itself	185	la012190
got	hj	iv	1983	la012289
gq	hmmm	ivo	1985	la012290
graphic	hmmmmm	ix	1986	la012389
graphics	ho	j	1987	la012390
great	hoo	j1	1989	la012489
grp	how	j10	1990	la012490
gte	howbeit	j2	19th	la012589
gto	however	j3	la	la012590
gtp	hoy	j4	la010189	la012689
gu	ht	j5	la010190	la012690
h	hu	j6	la010289	la012789
h1	hy	j8	la010290	la012790
h11	i	ja	la010389	la012889
h13	ibf	jc	la010390	la012890
h18	iby	ji	la010489	la012989
h2	ic	jo	la010490	la012990
h20	ida	jr	la010589	la013089
h2a	ie	just	la010590	la013090
h2o	if	k	la010689	la013189
h3	ignored	k1	la010690	la013190
h4	ii	k1n	la010789	la020189
h5	iii	k2	la010790	la020190
h6	ik	k2r	la010889	la020289
h7	il	k9	la010890	la020290
ha	im	ka	la010989	la020389
haa	ima	ka7	la010990	la020390
had	immediate	kan	la011089	la020489
hadn	in	kao	la011090	la020490
hardly	ina	kee	la011189	la020589
has	inasmuch	keep	la011190	la020590
have	inc	kept	la011289	la020689
haven	indeed	kg2	la011290	la020690
having	indicate	kg7	la011389	la020789
he	ing	kg8	la011390	la020790
headline	inner	kh1	la011489	la020889
hem	insofar	kh2	la011490	la020890
hence	instead	kh7	la011589	la020989
her	into	kh8	la011590	la020990
here	inward	know	la011689	la021089
hereafter	ip	knx	la011690	la021090
hereby	iq	ko	la011789	la021189
herein	ir	ky	la011790	la021190
hereupon	iri	l	la011889	la021289
hers	irk	10	la011890	la021290
herself	is	12	la011989	la021389
hi	isl	14	la011990	la021390

la021489	la031189	la040589	la043089	la052589
la021490	la031190	la040590	la043090	la052590
la021589	la031289	la040689	la050189	la052689
la021590	la031290	la040690	la050190	la052690
la021689	la031389	la040789	la050289	la052789
la021690	la031390	la040790	la050290	la052790
la021789	la031489	la040889	la050389	la052889
la021790	la031490	la040890	la050390	la052890
la021889	la031589	la040989	la050489	la052989
la021890	la031590	la040990	la050490	la052990
la021989	la031689	la041089	la050589	la053089
la021990	la031690	la041090	la050590	la053090
la022089	la031789	la041189	la050689	la053189
la022090	la031790	la041190	la050690	la053190
la022189	la031889	la041289	la050789	la060189
la022190	la031890	la041290	la050790	la060190
la022289	la031989	la041389	la050889	la060289
la022290	la031990	la041390	la050890	la060290
la022389	la032089	la041489	la050989	la060389
la022390	la032090	la041490	la050990	la060390
la022489	la032189	la041589	la051089	la060489
la022490	la032190	la041590	la051090	la060490
la022589	la032289	la041689	la051189	la060589
la022590	la032290	la041690	la051190	la060590
la022689	la032389	la041789	la051289	la060689
la022690	la032390	la041790	la051290	la060690
la022789	la032489	la041889	la051389	la060789
la022790	la032490	la041890	la051390	la060790
la022889	la032589	la041989	la051489	la060889
la022890	la032590	la041990	la051490	la060890
la030189	la032689	la042089	la051589	la060989
la030190	la032690	la042090	la051590	la060990
la030289	la032789	la042189	la051689	la061089
la030290	la032790	la042190	la051690	la061090
la030389	la032889	la042289	la051789	la061189
la030390	la032890	la042290	la051790	la061190
la030489	la032989	la042389	la051889	la061289
la030490	la032990	la042390	la051890	la061290
la030589	la033089	la042489	la051989	la061389
la030590	la033090	la042490	la051990	la061390
la030689	la033189	la042589	la052089	la061489
la030690	la033190	la042590	la052090	la061490
la030789	la040189	la042689	la052189	la061589
la030790	la040190	la042690	la052190	la061590
la030889	la040289	la042789	la052289	la061689
la030890	la040290	la042790	la052290	la061690
la030989	la040389	la042889	la052389	la061789
la030990	la040390	la042890	la052390	la061790
la031089	la040489	la042989	la052489	la061889
la031090	la040490	la042990	la052490	la061890

la061989	la071489	la080889	la090289	la092789
la061990	la071490	la080890	la090290	la092790
la062089	la071589	la080989	la090389	la092889
la062090	la071590	la080990	la090390	la092890
la062189	la071689	la081089	la090489	la092989
la062190	la071690	la081090	la090490	la092990
la062289	la071789	la081189	la090589	la093089
la062290	la071790	la081190	la090590	la093090
la062389	la071889	la081289	la090689	la100189
la062390	la071890	la081290	la090690	la100190
la062489	la071989	la081389	la090789	la100289
la062490	la071990	la081390	la090790	la100290
la062589	la072089	la081489	la090889	la100389
la062590	la072090	la081490	la090890	la100390
la062689	la072189	la081589	la090989	la100489
la062690	la072190	la081590	la090990	la100490
la062789	la072289	la081689	la091089	la100589
la062790	la072290	la081690	la091090	la100590
la062889	la072389	la081789	la091189	la100689
la062890	la072390	la081790	la091190	la100690
la062989	la072489	la081889	la091289	la100789
la062990	la072490	la081890	la091290	la100790
la063089	la072589	la081989	la091389	la100889
la063090	la072590	la081990	la091390	la100890
la070189	la072689	la082089	la091489	la100989
la070190	la072690	la082090	la091490	la100990
la070289	la072789	la082189	la091589	la101089
la070290	la072790	la082190	la091590	la101090
la070389	la072889	la082289	la091689	la101189
la070390	la072890	la082290	la091690	la101190
la070489	la072989	la082389	la091789	la101289
la070490	la072990	la082390	la091790	la101290
la070589	la073089	la082489	la091889	la101389
la070590	la073090	la082490	la091890	la101390
la070689	la073189	la082589	la091989	la101489
la070690	la073190	la082590	la091990	la101490
la070789	la080189	la082689	la092089	la101589
la070790	la080190	la082690	la092090	la101590
la070889	la080289	la082789	la092189	la101689
la070890	la080290	la082790	la092190	la101690
la070989	la080389	la082889	la092289	la101789
la070990	la080390	la082890	la092290	la101790
la071089	la080489	la082989	la092389	la101889
la071090	la080490	la082990	la092390	la101890
la071189	la080589	la083089	la092489	la101989
la071190	la080590	la083090	la092490	la101990
la071289	la080689	la083189	la092589	la102089
la071290	la080690	la083190	la092590	la102090
la071389	la080789	la090189	la092689	la102189
la071390	la080790	la090190	la092690	la102190

la102289	la111689	la121189	least	men
la102290	la111690	la121190	length	mfg
la102389	la111789	la121289	less	mg
la102390	la111790	la121290	lest	mi
la102489	la111889	la121389	let	might
la102490	la111890	la121390	li	mk
la102589	la111989	la121489	life	mm
la102590	la111990	la121490	like	mme
la102689	la112089	la121589	little	mo
la102690	la112090	la121590	ll	moc
la102789	la112189	la121689	lo	moo
la102790	la112190	la121690	long	more
la102889	la112289	la121789	loo	moreover
la102890	la112290	la121790	lot	most
la102989	la112389	la121889	lou	mostly
la102990	la112390	la121890	lp	mott
la103089	la112489	la121989	ls	mr
la103090	la112490	la121990	lt	ms
la103189	la112589	la122089	ltd	msg
la103190	la112590	la122090	lx	mt
la110189	la112689	la122189	ly	mu
la110190	la112690	la122190	m	much
la110289	la112789	la122289	m1	must
la110290	la112790	la122290	m16	mvp
la110389	la112889	la122389	m2	mx
la110390	la112890	la122390	m3	my
la110489	la112989	la122489	m4	myself
la110490	la112990	la122490	m5	n
la110589	la113089	la122589	m6	n1
la110590	la113090	la122590	m62	n2
la110689	la120189	la122689	m71	n5
la110690	la120190	la122690	m78	n6
la110789	la120289	la122789	ma	na
la110790	la120290	la122790	maa	na3
la110889	la120389	la122889	made	na6
la110890	la120390	la122890	mae	name
la110989	la120489	la122989	make	namely
la110990	la120490	la122990	mal	ncols
la111089	la120589	la123089	man	ncr
la111090	la120590	la123090	many	ne
la111189	la120689	la123189	may	near
la111190	la120690	la123190	mc	necessary
la111289	la120789	la2	mca	neither
la111290	la120790	la23	mcb	nev
la111389	la120889	la89	md	never
la111390	la120890	la90	mdc	nevertheles
la111489	la120989	last	me	s
la111490	la120990	latter	mea	new
la111589	la121089	latterly	meanwhile	next
la111590	la121090	le	mee	nfc

nfl	op	pp	really	should
ng	opt	pr	relatively	si
ni	or	prc	respectivel	since
nine	ora	pre	y	six
nl	ord	pro	right	smc
nlr	ot	probably	rk	smu
nmb	other	provides	ro	so
no	others	pt	roo	some
nobody	otherwise	pta	rowrule	somebody
nom	ou	ptl	rv	somehow
non	ought	q	s	someone
none	our	q106	s1	something
noone	ours	qb	s1630	sometime
nor	ourselves	qd2	s1w	sometimes
normally	out	qd3	s2	somewhat
not	outside	qd4	s358	somewhere
nothing	over	qd7	s4	soo
nov	overall	qd8	s605	sou
novel	ow	qe2	said	specified
now	own	qe4	same	specify
nowhere	p	qe5	sb	specifying
nr	p1	qe6	sba	spn
nu	p2	qe7	sbk	sq
o	pa	qe8	sc	sr
oat	page	qed	sca	ss
oc	part	qf2	scc	ssi
occ	particular	qf5	scca	ssy
och	particularly	qf7	sce	st
od	pb	qg4	scr	state
oda	pba	qg5	sdg	still
odd	pc	qg6	sdy	stu
of	pcb	qh4	se	su
off	pcc	qi	second	sub
oft	pcl	qtr	secondly	subject
often	pcp	que	section	such
oh	pe	quite	see	sup
oj	people	r	seem	sw
ol	per	r1	seemed	sy
old	perhaps	r2	seeming	syd
on	pfc	r2d2	seems	t
once	pg	r3	self	t4
one	ph	ra	selves	ta
ones	pic	ra3	sensible	table
only	pj	ra6	sent	tablecel
onto	placed	rather	serious	tablecell
ooh	plc	rb	seven	tablerow
ooo	please	rbi	several	take
oooo	plo	rc	shall	taken
ooooo	plus	rda	she	tcu
ooz	possible	re	shoo	td

te	towards	v8v	whence	x6
text	ts	v9l	whenever	xi
than	tu	v9w	where	xiii
that	twa	va	whereafter	xiv
the	twice	value	whereas	xix
their	two	various	whereby	xr
theirs	type	ve	wherein	xt
them	u	very	whereupon	xtra
themselves	u2	vh	wherever	xv
then	u60	vi	whether	xvi
thence	ua	via	which	xxi
there	uc	vii	while	xxiii
thereafter	ul	viii	whither	xxiv
thereby	ulf	vill	who	xxv
therefore	un	vin	whoever	xxvii
therein	unc	viz	whole	xxx
thereupon	und	vs	whom	xyz
these	under	vt	whose	y
they	unless	vu	why	y95
thi	uno	vw	wig	yap
third	until	vy	will	ye
this	unto	w	with	year
thorough	up	w6	within	years
thoroughly	upa	w8	without	yet
those	upon	wa	wk	yo
though	us	was	wok	yoo
three	use	wasn	word	you
through	used	way	words	your
throughout	useful	wbc	work	yours
thru	uses	wcb	world	yourself
thus	usf	wdm	would	yourselves
thy	using	we	wouldn	yve
time	usually	wee	wow	z
tmc	ut	well	wr	z28
tnt	uw	went	wt	z90
to	v	were	wu	zac
together	v20	what	wuz	zero
too	v2500	whatever	wwii	zx
tot	v6	whatnot	x	zz
toward	v8	when	x2	zzzz

2. Stop Words List: University of Tennessee

a	across	against	alone	although
about	after	all	along	always
above	afterwards	allows	already	am
accordingly	again	almost	also	among

amongst	cant	few	ignored	mr
an	cause	fifth	immediate	much
and	causes	first	in	must
another	certain	five	inasmuch	my
any	changes	followed	inc	myself
anybody	co	following	indeed	n
anyhow	come	for	indicate	name
anyone	consequentl	former	indicated	namely
anything	y	formerly	indicates	near
anywhere	contain	forth	inner	necessary
apart	containing	four	insofar	neither
appear	contains	from	instead	never
appropriate	correspondi	further	into	nevertheless
are	ng	furthermore	inward	new
around	could	g	is	next
as	currently	get	it	nine
aside	d	gets	its	no
associated	day	given	itself	nobody
at	described	gives	j	none
available	did	go	just	noone
away	different	gone	k	nor
awfully	do	good	keep	normally
b	does	got	kept	not
back	doing	great	know	nothing
be	done	h	l	novel
became	down	had	last	now
because	downwards	hardly	latter	nowhere
become	during	has	latterly	o
becomes	e	have	least	of
becoming	each	having	less	off
been	eg	he	lest	often
before	eight	hence	life	oh
beforehand	either	her	like	old
behind	else	here	little	on
being	elsewhere	hereafter	long	once
below	enough	hereby	ltd	one
beside	et	herein	m	ones
besides	etc	hereupon	made	only
best	even	hers	make	onto
better	ever	herself	man	or
between	every	him	many	other
beyond	everybody	himself	may	others
both	everyone	his	me	otherwise
brief	everything	hither	meanwhile	ought
but	everywhere	how	men	our
by	ex	howbeit	might	ours
c	example	however	more	ourselves
came	except	i	moreover	out
can	f	ie	most	outside
cannot	far	if	mostly	over

overall	sensible	theirs	until	whereupon
own	sent	them	unto	wherever
p	serious	themselves	up	whether
particular	seven	then	upon	which
particularly	several	thence	us	while
people	shall	there	use	whither
per	she	thereafter	used	who
perhaps	should	thereby	useful	whoever
placed	since	therefore	uses	whole
please	six	therein	using	whom
plus	so	thereupon	usually	whose
possible	some	these	v	why
probably	somebody	they	value	will
provides	somehow	third	various	with
q	someone	this	very	within
que	something	thorough	via	without
quite	sometime	thoroughly	viz	work
r	sometimes	those	vs	world
rather	somewhat	though	w	would
really	somewhere	three	was	x
relatively	specified	through	way	y
respectively	specify	throughout	we	year
right	specifying	thru	well	years
s	state	thus	went	yet
said	still	time	were	you
same	sub	to	what	your
second	such	together	whatever	yours
secondly	sup	too	when	yourself
see	t	toward	whence	yourselves
seem	take	towards	whenever	z
seemed	taken	twice	where	zero
seeming	than	two	whereafter	
seems	that	u	whereas	
self	the	under	whereby	
selves	their	unless	wherein	

3. Stop Words List: University of Glasgow

a	beyond	former	ltd	out
about	bill	formerly	made	over
above	both	forty	many	own
across	bottom	found	may	part
after	but	four	me	per
afterwards	by	from	meanwhile	perhaps
again	call	front	might	please
against	can	full	mill	put
all	cannot	further	mine	rather
almost	cant	get	more	re
alone	co	give	moreover	same
along	computer	go	most	see
already	con	had	mostly	seem
also	could	has	move	seemed
although	couldnt	hasnt	much	seeming
always	cry	have	must	seems
am	de	he	my	serious
among	describe	hence	myself	several
amongst	detail	her	name	she
amongst	do	here	namely	should
amount	done	hereafter	neither	show
an	down	hereby	never	side
and	due	herein	nevertheles	since
another	during	hereupon	s	sincere
any	each	hers	next	six
anyhow	eg	herself	nine	sixty
anyone	eight	him	no	so
anything	either	himself	nobody	some
anyway	eleven	his	none	somehow
anywhere	else	how	noone	someone
are	elsewhere	however	nor	something
around	empty	hundred	not	sometime
as	enough	i	nothing	sometimes
at	etc	ie	now	somewhere
back	even	if	nowhere	still
be	ever	in	of	such
became	every	inc	off	system
because	everyone	indeed	often	take
become	everything	interest	on	ten
becomes	everywhere	into	once	than
becoming	except	is	one	that
been	few	it	only	the
before	fifteen	its	onto	their
beforehand	fify	itself	or	them
behind	fill	keep	other	themselves
being	find	last	others	then
below	fire	latter	otherwise	thence
beside	first	latterly	our	there
besides	five	least	ours	thereafter
between	for	less	ourselves	thereby

therefore	which
therein	while
thereupon	whither
these	who
they	whoever
thick	whole
thin	whom
third	whose
this	why
those	will
though	with
three	within
through	without
throughout	would
thru	yet
thus	you
to	your
together	yours
too	yourself
top	yourself
toward	
towards	
twelve	
twenty	
two	
un	
under	
until	
up	
upon	
us	
very	
via	
was	
we	
well	
were	
what	
whatever	
when	
whence	
whenever	
where	
whereafter	
whereas	
whereby	
wherein	
whereupon	
wherever	
whether	

Appendix B

Source Code: TREC File Extractor

Download and Install following XAMPP software

<http://www.apachefriends.org/download.php?xampp-win32-1.6.6a-installer.exe>

Click on the EXE and follow the easy steps (Please check "Install Apache as Service", "Install MySQL as Service"). It will create PHP/MySQL & Apache environment for you in the local PC. Please choose a small named directory as the installation directory (for example "c:\php" or "c:\apache" names are very small enough to easily access from code).

How to install:

Extract "TREC file spliter.zip". Put all the files under a webroot directory. As you are using XAMPP (apache 2 friends), the webroot directory is "htdocs".

Please create a directory (e.g., dp) under directory "c:\php\htdocs" and put all files there (e.g., c:\php\htdocs\dp). Then open the ".htaccess" file using wordpad.

I consider your webroot is "C:/php/htdocs/" and "dp" is your project dir where you kept all files.

Now type this address in your browser:

http://localhost/dp/
user:ankzbd
password:unbcca

TIPS:

Goto the "dp" directory there are two other directories "new_input" Copy all large files here and goto <http://localhost/dp/> (to access the program) to process the files and now check the "new_output" directory for output files. It will help you to avoid upload and download the files.

Version 1.0

Main.php

```
/* 1. Creates the initial GUI
   2. Calls show_input_files.php
   3. Calls show_output_files.php
   4. Calls upload_file.php
*/
```

```
<table align="left" width="90%">
```

```
<tr>
```



```

        <td><td>
        <td>&nbsp;</td>
</tr>

<tr bgcolor="#fcfcfc">
  <td colspan="2" align="center" class="header"><h2>Data Pre- processing</h2></td>
</tr>

<?php
if(!empty($msg))
{
?>

<tr bgcolor="#C1CDCE">
  <td>&nbsp;</td><td colspan="1" align="center"><?=$msg?></td>
</tr>

<?php
}
?>

<tr valign="top">

  <td width="15%">
    <table width="90%" style="border: 1px solid #9c9c9c">
      <tr>
        <td><a href="?action=show_input_files"
          class="<?=$class1?>">Input Files</a></td>
      </tr>

      <tr>
        <td><a href="?action=show_output_files"
          class="<?=$class2?>">Output Files</a></td>
      </tr>

      <tr>
        <td><a href="?action=upload_file" class="<?=$class3?>">Upload
          Input</a></td>
      </tr>

    </table>
  </td>

  <td id="content">

    <?php
      include($file_to_include);
    ?>

  </td>
</tr>

```

```

<tr bgcolor="#fcfcfc">
    <td colspan="2" align="center" class="footer">
        &copy;ANK Zaman, zaman@unbc.ca, University of Northern British
        Columbia.
    </td>
</tr>

```

```
</table>
```

```
index.php
```

```

/*
 1. Calling common.lib.php and XmlToArray.class.php
 2. Switches between following four operations
    i. create_output_files ii. show_output_files iii. show_uploader (to
    upload files to be processed iv. show_input_files
 3. Handles some error messages and file deletion operation (if needed)
*/
<?php
ob_start(); // Turn on output buffering

require_once("common.lib.php");
require_once("XmlToArray.class.php");

$files = array();

$action = 'show_input';

if(!empty($_GET['created']))
{
    $msg = "Output files created successfully";
}

if(!empty($_GET['action']))
{
    $action = trim($_GET['action']);
}

if(!empty($_GET['action2']))
{
    $action2 = trim($_GET['action2']);
    $file2del = $_GET['file'];
    $filetype = $_GET['type'];

    $file2del = 'new_' . $filetype . "/" . $file2del;

    if(unlink($file2del))
    {
        $msg = "File deleted successfully";
    }
}

if(!empty($_POST['delete_selected']))

```

```

{
$filetype = $_POST['file_type'];

$del_cnt = 0;

foreach($_POST['files'] as $file2del=>$val)
{
    if($val == 1)
    {
        $file2del = 'new_' . $filetype . "/"$file2del";
        unlink($file2del);
        $del_cnt++;
    }
}

if($del_cnt)
{
    $msg = "$del_cnt files deleted succesfully";
}
}

if(!empty($_FILES['input_file']['name']))
{
    $dest = "new_input/" . $_FILES['input_file']['name'];

    if(move_uploaded_file($_FILES['input_file']['tmp_name'], $dest))
    {
        $msg = 'File <' . $_FILES['input_file']['name'] . '> uploaded    successfully';
    }
    else
    {
        $msg = 'Error: file <' . $_FILES['input_file']['name'] . '> was not    uploaded';
    }
}

}
$class1 = 'unselected';
$class2 = 'unselected';
$class3 = 'unselected';
$class4 = 'unselected';

switch($action)
{
    case 'create':
        $file_to_include = "tpls/create_output_files.php";
        break;

    case 'show_output_files':
        $files = findDirFiles("new_output");
        $file_to_include = "tpls/show_output_files.php";
        $class2 = 'selected';
}

```

```

break;

case 'upload_file':
    $file_to_include = "tpls/show_uploader.php";
    $class3 = 'selected';
break;

case 'show_input_files':
default:
    $files = findDirFiles("new_input");
    $file_to_include = "tpls/show_input_files.php";
    $class1 = 'selected';
}

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
  <head>
    <title>Data Pre-processing</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
    <script type="text/javascript" src="common.js"></script>
  </head>

  <body bgcolor="#cccccc">

    <?php
      include("tpls/main.php");
    ?>

  </body>
</html>

```

Show uploader.php

```

/* It creates file upload form to upload files to be processed and displays files after
uploading by calling show_input_files.php
*/
<table width="100%">
  <form name="inputForm" action="?action=show_input_files" enctype="multipart/form-
data" method="POST">
    <input type="hidden" name="upload" value="yes">

    <tr bgcolor="#D4D0C2" height="20">
      <th colspan="2" align="left">Please Upload Your Input File</th>
    </tr>

    <tr>
      <td><input type="file" name="input_file"></td>
    </tr>

```

```

        <tr>
        <td><input type="submit" value="Upload"></td>
        </tr>

        <tr bgcolor="#D4D0C2">
        <td colspan="2">&nbsp;   </td>
        </tr>

</form>

</table>

```

Common.lib.php (traverse the input directory)

```

/* Travers the input folder reads every input file one after another and extract
every document as a separate file with unique name
*/
<?php
function getbetweenTwo($textBefore,$textAfter,$allText,$offset=0)
{
    $pattern='#'.$textBefore.'(.*'.$textAfter.'#isU';

    preg_match_all($pattern, $allText,$matches);

    /* Searches subject for all matches to the regular expression given in pattern
and puts them in matches in the order specified by flags . After the first match is
found, the subsequent searches are continued on from end of the last match. */

    return @$matches[1][$offset];
} //EO Method

function getbetween2($textBefore,$textAfter,$allText,$offset=0)
{
    $pattern='#'.$textBefore.'(.*'.$textAfter.'#isU';
    preg_match_all($pattern, $allText,$matches);
    return @$matches[1];
} //End of function getbetween2()

/**
 * Traverses & Find Files in Directory
 * @param $path to traverse
 * @return $files
 */
function findDirFiles($path)
{
    $dir = opendir ($path);
    $files = array();

    while ($file = readdir ($dir))
    {

```

```

    if (($file == ".") or ($file == ".."))
    {
        continue;
    }

    if (filetype("$path/$file") == "dir")
    {
        ;
    }
    else
    {
        $files[] = $file;
    }
} //End of while

closedir($dir);

return $files;

} //End of function findDirFiles()

function createOutputFile($inputFile)//extracting documents form large files.
{
    $xml_data = file_get_contents("new_input/$inputFile");
    $arrayData = array();
    $arrayData = getbetween2('<DOC>', '</DOC>', $xml_data);

    //Displaying the Array

    echo "<pre>";
    print_r($arrayData);
    echo "</pre>";

    if($num_docs = sizeof($arrayData))
    {
        foreach($arrayData as $v)
        {
            $doc_no = getbetweenTwo('<DOCNO>', '</DOCNO>', $v);
            $doc_no = str_replace('-', '_', trim($doc_no));

            //file_data = "<DOC>\r\n" . $v . "\r\n</DOC>";
            $file_data = "<DOC>" . strtolower($v) . "</DOC>";

            $fp = fopen("new_output/$doc_no", "w");
            fwrite($fp, $file_data);
            fclose($fp);
        }
    }
} //End of createOutputFile() Function

```

?>

Show_input_File.php

```
/* 1. Displays the list of input files (file name and size)
   2. Controls the deletion of input files
   3. Controls the download option of input files by calling download.php
   4. Calls showFile()
*/
```

```
<table width="100%">
<form name="inputForm" action="?action=show_input_files" method="POST">
  <input type="hidden" name="delete_selected" value="yes">
  <input type="hidden" name="file_type" value="input">

  <tr bgcolor="#D4D0C2" >
    <td width="10"><input type="checkbox" name="checkall"
      onclick="checkUncheckAll(this);"/></td>
    <td>File Name</td>
    <td>File Size (Bytes)</td>

  </tr>

  <?php
  if(sizeof($files))
  {
    foreach($files as $fname)
    {
      $fsize = filesize("new_input/$fname");
    }
  }
  <tr>
    <td><input type="checkbox" name="files[<?=$fname?>]"
      value="1"/></td>
    <td><?=$fname?></td>
    <td><?=$fsize?></td>
    <td><a href="javascript: showFile('input',
      '<?=$fname?>');">View Content</a> | <a
      href="?action=create&file=<?=$fname?>">Create Output Files</a>
      | <a
      href="?action=<?=$_GET['action']?>&action2=delete&type=input&file=<
      =$fname?>" onClick="javascript: return confirm('Are you sure to
      delete?')">Delete</a> | <a
      href="download.php?type=input&file=<?=$fname?>">Download</a></td>
  </tr>

  <?php
  }
}
?>

  <tr bgcolor="#D4D0C2">
```

```

        <td colspan="4"><input type="submit" name="delete_selected" value="
Delete Selected "></td>
    </tr>

```

```

</form>
</table>

```

Creat_output_files.php

```

/*    calls createOutputFile() function */
<?php

$file = $_GET['file'];
createOutputFile($file);
header("location: ?action=show_output_files&created=1");
exit;

?>

```

Show_output_files.php

```

/*
1. Displays the list of output files (file name and size)
2. Controls the deletion of output files
3. Controls the download option of output files by calling      download.php
4. Calls showFile()
*/

```

```

<table width="100%">
    <form name="inputForm" action="?action=show_output_files" method="POST">
        <input type="hidden" name="delete_selected" value="yes">
        <input type="hidden" name="file_type" value="output">

        <tr bgcolor="#D4D0C2" >
            <td width="10"><input type="checkbox" name="checkall"
onclick="checkUncheckAll(this);"/></td>
            <td>File Name</td>
            <td>File Size (Bytes)</td>
            <td></td>
        </tr>

        <?php
        if(sizeof($files))
        {
            foreach($files as $fname)
            {
                $fsize = filesize("new_output/$fname");
            }
        }
        ?>
        <tr>
            <td><input type="checkbox" name="files[<?=$fname?>]"
value="1"/></td>
            <td><?=$fname?></td>
            <td><?=$fsize?></td>

```



```

        <td><a href="javascript: showFile('output', '<?=$fname?>');">View
Content</a>
        |
        <a href="?action=<?=$_GET['action']?>&action2=delete&type=output&file=
<?=$fname?>" onClick="javascript: return confirm('Are you sure want to
delete?')">Delete</a>
        |
        <a href="download.php?type=output&file=<?=$fname?>">Download</a></t
d>
    </tr>

    <?php
    }
    }
    ?>

    <tr bgcolor="#D4D0C2">
        <td colspan="4"><input type="submit" name="delete_selected" value="
Delete Selected "></td>
    </tr>

</form>
</table>

```

Show_file.php

```

/*    creates a button "close window" at the top (while displaying a file content
*/
<?php

$dir = "new_" . $_GET['ftype'];
$file = $_GET['fname'];

$filename = "$dir/$file";

?>

<a href="javascript:window.close();">CLOSE WINDOW</a><br />
<p style="width: 100%; background-color: #9c9c9c;">
<?php

highlight_file($filename); // //highlights the selected file.

?>
</p>

```

Download.php (

```

<?php

```

```

/* PHP scripts often generate dynamic content that must not be cached by the client
browser or any proxy caches between the server and the client browser. Many proxies and
clients can be forced to disable caching with following header():*/
## avoid all caching

header('Cache-Control: no-cache, must-revalidate');
header('Pragma: no-cache');
header('Expires: Mon,26 Jul 1980 05:00:00 GMT'); // date in the past

$type = $_GET['type'];
$filename = "new_{$type}/".$_GET['file'];

if(file_exists($filename))//to download extracted files
{
    header('Content-Type: application/octet-stream');
    header("Content-Length: " . filesize($filename));
    header('Content-Transfer-Encoding: base64');
    header('Content-Disposition:                               attachment;
filename="'.basename($filename).'"');
    readfile($filename);
}
else
{
    die("<font color=RED><h2>This resource does not exist</h2></font>");
    // Equivalent to exit()
}
?

```

INTRODUCTION: Stemming and Stop Word Removal Program

Version 1.0

pstem folder contains the following two folders

inputs : this folder contains the input text files

outputs: generated output text files will be saved here

and the following files

index.php : this file start to access the input folder by calling the process.php file

output.php : this file helps to display the generated output files

process.php: this file creates temporary array of terms and compare then with stop word list for (to keep or to delete it) then call the stemming.php for stemming.

stemming.php: this file contains the code of actual Porter's stemming.

stopword.txt: it contains the list of stop words that will be discarded form the input files

How to Install

Download and Install following XAMPP software

<http://www.apachefriends.org/download.php?xampp-win32-1.6.6a-installer.exe>

Click on the EXE and follow the easy steps (Please check "Install Apache as Service", "Install MySQL as Service"). It will create PHP/MySQL & Apache environment for you in the local PC.

Please choose a small named directory as the installation directory (for example "c:\php" or "c:\apache" names are very smalls enough to easily access from code).

How to install:

Extract "pstem.zip". Copy "pstem" folder and put it into webroot directory. As you are using XAMPP (apache 2 friends), the webroot directory is "htdocs".

I consider your webroot is "C:/php/htdocs/" and "pstem" is your program dir where you kept all files.

INDEX.PHP

```
/*
This program is free to use (and modification) for education and research.
This program has developed as part of masters thesis in the area of Information
Retrieval(IR).
If you have any question Contact with

Zaman email zaman@unbc.ca
or
Dr. Charles Brown email brownc@unbc.ca
Computer Science Program
University of Northern British Columbia
*/
```

```
/* This program creates a button called "CONTIONUE" and call the process.php program
segment.*/
```

```
<h1> Stop Word Removal and Porter's Stemming :: Step 1</h1>
Please place all input files in dir "inputs" and then press "Continue" button.
<br />
<br />
<input type="button" value="Continue" &raquo; "
onClick="window.location.href='process.php'">
```

PROCESS.PHP

```

/*
This program is free to use (and modification) for education and research.
This program has developed as part of masters thesis in the area of Information
Retrieval(IR).
If you have any question Contact with

Zaman email zaman@unbc.ca
or
Dr. Charles Brown email brownc@unbc.ca
Computer Science Program
University of Northern British Columbia
*/

/*   This program segment calls the stemming.php
      reads the input files
      removes stop words
      performs stemming
      creates new files with the same name as input files
*/
<?php

ob_start();// Turn on output buffering

set_time_limit(0);
// Limits the maximum execution time, If set to zero, no time limit is imposed

echo "<h1>Step 2 :: Processing Files</h1>Please wait while processing ...<BR />";

include('stemming.php');

$obj = new PorterStemmer();

$stop_words = array();

$dir_files = array();
$dir_files = findDirFiles("inputs");

if(!sizeof($dir_files))
{
    echo "<br />No input files found. <a href=index.php>Go back</a> and try
again.";
    exit;
}

$file_index = 1;

if(!empty($_GET['file_index']))
{
    $file_index = $_GET['file_index'];
}

```

```

        if(!empty($dir_files) && $file_index <= sizeof($dir_files))
        {
            $inp = $dir_files[$file_index - 1];

            $input_file = "inputs/$inp";
            $file_data = file_get_contents($input_file);

            $stop_words = file('stopword.txt');
            array_walk($stop_words, 'trim_value');

            foreach($stop_words as $word)
            {
                $word = trim($word);
                //cleans every word with extra space, tab, new           //line
            }
etc
            $words = array(" $word ", "\n$word ", "\r\n$word           ", " $word\n",
" $word\r\n", "\n$word\n",           "\r\n$word\r\n");
            $replacements = array(' ', ' ', ' ', ' ', ' ',           ", ");

            $file_data = str_ireplace($words, $replacements,           $file_data);
                //replaces stop words with NULL character
        }

        // Implementing Porter's Stemmer
        $file_data = preg_replace_callback('/(\w+)/i',           'replace_fn',
$file_data);

        /* preg_replace_callback() - returns an array if the subject parameter is
        an array, or a string otherwise.If matches are found, the new
        subject will be returned, otherwise subject will be returned unchanged.
        */

        $output_file = "outputs/$inp";

        $fp = fopen($output_file, "w");
        fwrite($fp, $file_data);
        fclose($fp);

        echo "<b>$file_index input files processed so           far.</b>";

        $file_index++;
        echo "<meta http-equiv=\"refresh\" content=\"2;
        url=process.php?file_index=$file_index\">";

    }
else
{

```

```

        echo "<br /><b>Processing Finished.</b> <a href=output.php>See output
files</a>.";
    }

function trim_value(&$v)
//Referencing the word to be trimmed
{
    $v = trim($v);
}

function replace_fn($matches)
//Calling stem() function from stemming.php
{
    global $obj;
    $matches[1] = $obj->Stem($matches[1]);
    return $matches[1];
}

function findDirFiles($path)
//to access and organize inputs directory
{
    $dir = opendir ($path);
    $files = array();

    while ($file = readdir ($dir))
    {
        if (($file == ".") or ($file == ".."))
        {
            continue;
        }

        if (filetype("$path/$file") == "dir")
        {
            ;
        }
        else
        {
            $files[] = $file;
        }
    } //End of while

        closedir($dir);

        sort($files);

        return $files;

} //End of function findDirFiles()

```

?>

OUTPUT.PHP

```
/*
 *This program is free to use (and modification) for education and research.
 *This program has developed as part of masters thesis in the area of Information
 Retrieval(IR).
 *If you have any question Contact with

 *Zaman email zaman@unbc.ca
 *or
 *Dr. Charles Brown email brownc@unbc.ca
 *Computer Science Program
 *University of Northern British Columbia
 */

/* This program segment organizes (by sorting) and displays the output files*/

<?php

echo "<h1>Step 3:: Output</h1>";

$outputs = array();
$outputs = findDirFiles('outputs');

if(sizeof($outputs))
{
    sort($outputs);

    foreach($outputs as $filename)
    {
        echo '<br /><b><a href="outputs/'.$filename.'">'.$filename.'</a></b>';
    }
}
else
{
    echo "No output files found.";
}

function findDirFiles($path)
//to access and organize outputs directory
{
    $dir = opendir ($path);
    $files = array();

    while ($file = readdir ($dir))
    {
        if (($file == ".") or ($file == ".."))
```

```
{
  continue;
}

if (filetype("$path/$file") == "dir")
{
  ;
}
else
{
  $files[] = $file;
}
} //End of while

closedir($dir);

sort($files);

return $files;

} //End of function findDirFiles()
?>
```


Appendix C

General Text Parser [University of Tennessee][Giles et al 2001]

GTP has a command-line interface of the form -

gtp filename -c common_word_file -t temp_dir [options]

where, filename must be the first argument. The -c and -t specifiers must exist, but they may be in any order. Options are enlisted in the following table-

Options	Explanation
-i	Index of input files
-c	<i>common words file (stop word file)</i>
-w	Specify a custom weighting scheme. Local and global refer to local and global weighting formulas. Local can be tf (term frequency), log, or binary. Global can be normal, idf, or entropy.
-t	Temp Directory (where temporary working files are stored.)
-z	Perform singular value decomposition.
-R	Specify a name for the current run of GTP.
-h	Create the Harwell-Boeing compressed matrix.
-u	Keep the Harwell-Boeing compressed matrix in an uncompressed file (on output) if the matrix is created.
-O	Specify that the <i>output</i> file is to be in one binary file for SVD. (keys.gdbm)
-o	Specify that the key, id# global frequency, (e.g., freqtable.rtf)

Table Table B. 1: GTP Options [Giles et al 2001]

The first argument, filename, is the name of the file or directory to be parsed. If filename is actually a directory, gtp traverses this directory and all subdirectories in a recursive fashion and parses each regular file it encounters. If filename is a single file, gtp simply parses it only. Gtp moves sequentially through each file, extracting keys/terms comprised of relevant characters, and ignoring keys contained in the common word list specified by the arguments -c common_word_file (set of stop words).

As gtp tokenizes the terms, it associates them with a number. The number is incremented sequentially in the order they are encountered. After tokenizing the keys and associating

each one with the document it was extracted from, gtp begins calculating term weights. The (global) weights of the terms are computed over the collection of documents if no global weighting scheme is specified each term carries a global weight of 1. By default, only a local weight is assigned and this is simply the frequency with which the term appears in a document. Two thresholds exist for term frequencies: Global and local. By default, the global and local thresholds are both 1. A term must appear more than 1 time in the entire collection AND in more than one document in the collection before it will be weighted.

With the use of different options, GTP produces several other files in varying formats. The output file, created by specifying the -O option, is a binary file that contains all the vector (term and document) and singular value information produced by the SVD. The layout of the output file is as follows: header information consisting of the number of terms parsed, number of documents encountered, and the number of factors used; term vectors for all terms parsed; document vectors for all documents parsed; singular values computed. The following table presents the generated files/folder by GTP:

File name	Type Description (Section)
<i>RUN SUMMARY</i>	Summary of options used (Listed in table 3)
<i>LAST RUN</i>	Summary of options used on most recent GTP run (Listed in table 3)
<i>Output</i>	Binary Vector information generated by SVD
<i>keys.gdbm</i>	DBM Database of keys generated (http://gnuwin32.sourceforge.net/packages/gdbm.htm)
<i>Rawmatrix.rtf</i>	Raw term-by-document matrix
<i>matrix.hb.rtf</i>	Contains the Harwell-Boeing matrix.
<i>lao2</i>	Summary of SVD calculation.
<i>nonz</i>	Total terms in the document collection, including those are not parsed.
<i>index.i.rtf</i>	Index of input files
<i>temp</i>	Directory where temporary working files are stored.
<i>Frequency.rtf</i>	Term, serial, raw_global_frequency, #files_present_the_term, global_weight (idf2) [idf2 formula is found in the files wgt.h and weight.cc in the folder src\wingtp\

Table B. 2 GTP Generated Files/Folder [Giles et al 2001]

Query Processing with GTPQUERY

GTPQUERY relies completely on files produced by a standard GTP run. Those files are output, keys (keys.gdbm), and LAST RUN. Query processing will fail if these files are not in the working directory.

Gtpquery has a command-line interface of the form –

gtpquery filename -c common_word_file [options]

- where filename must be the first argument. The -c specifier must exist, but may be in any order.

The first argument, filename, is the name of the file or directory containing queries to be parsed. If filename is actually a directory, gtpquery traverses this directory and all subdirectories in a recursive fashion and parses each regular file it encounters. If filename is a single file, gtpquery simply parses it only. Gtpquery moves sequentially through each file, extracting keys comprised of relevant characters, and ignoring keys contained in the common word list specified by the arguments -c common_word_file. By default, only keys that begin with characters A-z will be parsed. Keys beginning with a digit (0-9), with the exception of numbers that could be interpreted as dates in the 1700's 1800's and 1900's, will be ignored to the next whitespace character. By default, only characters A-z and 0-9 are considered to be characters to be included as part of the remainder of a key. Other options are enlisted below-

-help

Summarize options.

-S

Scale the query vector by the singular values before calculating cosine similarity.

-n

Set the number of factors to use (default is found in the LAST_RUN file; the value of nfact. The LAST_RUN file is a shortened version of the RUN_SUMMARY file. It contains all the parameters used in your last gtp run.).

-k #

Set the number of results returned (default is all).

-p

Do not print query completion messages to standard output.

Query processing is performed by finding a cosine similarity measure between a query vector and document vectors. The query vector can be considered a pseudo-document.

The files that GTPQUERY produces are strictly results files. Each file has a prefix of q result.#, where # is a number starting with 1. The number represents the corresponding number id of the query that was performed. The file contains a list of document ids and cosine similarity measures organized by most relevant to least relevant. GTP is a software package that provides text parsing of small to large document collections and matrix decomposition for use in information retrieval applications. The brief summary of GTP is presented below-

- (i) Parse text (single files or directories),
- (ii) Construct sparse matrix data structures (with choices of different term weighting strategies),
- (iii) Perform selected matrix decompositions for the representation of terms, documents, and queries in a reduced-rank vector space, and
- (iv) Convert user-supplied natural language queries into appropriate query vectors for cosine-based matching against term and/or document vectors in that vector space.

GTP is a public domain software, which (includes documentations) is freely available for anyone to download form the following link-

<http://www.cs.utk.edu/~lsi>

Appendix D

T-Test Calculations

Notes		
Output Created		20-Aug-2009 16:17:02
Comments		
Input	Data	\\pg-uni-fs-01\zaman\TSProfile\Desktop\data.sav
	Active Dataset	DataSet1
	Filter	<none>
	Weight	<none>
	Split File	<none>
	N of Rows in Working Data File	10
Missing Value Handling	Definition of Missing	User defined missing values are treated as missing.
	Cases Used	Statistics for each analysis are based on the cases with no missing or out-of-range data for any variable in the analysis.
Syntax		T-TEST PAIRS=System1 System1 System1 WITH system2 system3 system4 (PAIRED) /CRITERIA=C(.9500) /MISSING=ANALYSIS.
Resources	Processor Time	0:00:00.000
	Elapsed Time	0:00:00.000

[DataSet1] \\pg-uni-fs-01\zaman\TSProfile\Desktop\data.sav

Paired Samples Statistics					
		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	System1	.080710	10	.0381119	.0120520
	system2	.074230	10	.0306572	.0096946
Pair 2	System1	.080710	10	.0381119	.0120520
	system3	.072560	10	.0326347	.0103200
Pair 3	System1	.080710	10	.0381119	.0120520
	system4	.072110	10	.0219340	.0069361

Paired Samples Correlations				
		N	Correlation	Sig.
Pair 1	System1 & system2	10	.958	.000
Pair 2	System1 & system3	10	.989	.000
Pair 3	System1 & system4	10	.967	.000

Paired Samples Test				
		Paired Differences		
		Mean	Std. Deviation	Std. Error Mean
Pair 1	System1 - system2	.0064800	.0124390	.0039336
Pair 2	System1 - system3	.0081500	.0075823	.0023977
Pair 3	System1 - system4	.0086000	.0178220	.0056358

Paired Samples Test						
		Paired Differences		t	df	Sig. (2-tailed)
		95% Confidence Interval of the Difference				
		Lower	Upper			
Pair 1	System1 - system2	-.0024183	.0153783	1.647	9	.134
Pair 2	System1 - system3	.0027259	.0135741	3.399	9	.008
Pair 3	System1 - system4	-.0041491	.0213491	1.526	9	.161

Level of significance for two-tailed t-test

Degrees of Freedom (df)	Level of significance for two-tailed test			
	0.1	0.05	0.01	0.001
1	6.31	12.71	63.66	636.62
2	2.92	4.30	9.93	31.60
3	2.35	3.18	5.84	12.92
4	2.13	2.78	4.60	8.61
5	2.02	2.57	4.03	6.87
6	1.94	2.45	3.71	5.96
7	1.89	2.37	3.50	5.41
8	1.86	2.31	3.36	5.04
9	1.83	2.26	3.25	4.78
10	1.81	2.23	3.17	4.59
11	1.80	2.20	3.11	4.44
12	1.78	2.18	3.06	4.32
13	1.77	2.16	3.01	4.22
14	1.76	2.14	2.98	4.14
15	1.75	2.13	2.95	4.07
16	1.75	2.12	2.92	4.02
17	1.74	2.11	2.90	3.97
18	1.73	2.10	2.88	3.92
19	1.73	2.09	2.86	3.88
20	1.72	2.09	2.85	3.85
21	1.72	2.08	2.83	3.82
22	1.72	2.07	2.82	3.79
23	1.71	2.07	2.82	3.77
24	1.71	2.06	2.80	3.75
25	1.71	2.06	2.79	3.73
26	1.71	2.06	2.78	3.71
27	1.70	2.05	2.77	3.69
28	1.70	2.05	2.76	3.67
29	1.70	2.05	2.76	3.66
30	1.70	2.04	2.75	3.65
40	1.68	2.02	2.70	3.55
60	1.67	2.00	2.66	3.46
120	1.66	1.98	2.62	3.37
infinity	1.65	1.96	2.58	3.29

Table C: Level of significance for two-tailed t-test [Fisher & Yates 1995]

Publications

1. **A N K Zaman**, and Charles Grant Brown, – "*Latent Semantic Indexing and Large Dataset: Study of Term-Weighting Schemes*", accepted for publication in the IEEE Fifth International Conference on Digital Information Management (ICDIM 2010), Lakehead University, Thunder Bay, Canada from July 5-8, 2010.
2. **A N K Zaman**, and Charles Grant Brown, – "*Information Retrieval (IR) from a Large Data Set*" published in the proceedings of International Conference on Electronics, Computer and Communication (ICECC 2008), June 27-29, 2008, University of Rajshahi, Bangladesh. ISBN 984-300-002131-3.