Applying Local Latent Semantic Indexing for Information Retrieval Visualization

by

Michael Hugh Miller
B.A., University of Victoria, 1992

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Health Information Science

We accept this thesis as conforming
to the required standard

[redacted]

Dr. P. Fisher, Supervisor (School of Health Information Science)

[redacted]

Dr. J. McDaniel, Departmental Member (School of Health Information Science)

[redacted]

Dr. M. Masson, Outside Member (Department of Psychology)

[redacted]

Dr. F. Ruskey, External Examiner (Department of Computer Science)

University of Victoria

Supervisor: Dr. Paul Fisher

ABSTRACT

Health professionals and consumers are not keeping pace with advancements in knowledge reported in the scientific literature. One of the reasons for this state of affairs is the lack of effective tools to search and present relevant information from MEDLINE and other literature sources. Current techniques are hindered by the ambiguities of natural language and the difficulty of presenting results from a search to the user effectively. This thesis presents a number of core methodologies for indexing and searching text databases such as MEDLINE and discusses the benefits and drawbacks of each method. Four techniques for visualizing search results are also presented and discussed including a novel method proposed by the author called Local Latent Semantic Indexing / Cluster (LLSI/Cluster). Experiments with LLSI/Cluster on three test collections of MEDLINE articles indicate that similar articles tend to cluster together. These findings suggest that LLSI/Cluster has potential as a visualization method for displaying a large set of documents to the user in a graphical manner.

Examiners:

[redacted]

Dr. P. Fisher, Supervisor (School of Health Information Science)

[redacted]

Dr. J. McDaniel, Departmental Member (School of Health Information Science)

[redacted]

Dr. M. Masson, Outside Member (Department of Psychology)

[redacted]

Dr. F. Ruskey, External Examiner (Department of Computer Science)

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my advisor Paul Fisher for giving me the freedom to choose my own thesis topic.

Thanks to the other members of my Supervisory Committee, Dr. Michael Masson, and Dr. Jim McDaniel for their helpful comments and suggestions. Special thanks to Jim McDaniel who kept me focused on the completion of this thesis, and was always willing to lend an ear, even at short notice. Jim also offered several useful suggestions related to the coding of the LLSI/Cluster system.

Thanks to Jochen Moehr for his enthusiasm and encouragement, and for providing me with part-time employment on the HEAL*Net* project.

Thanks to Dr. Yuri Kagolofsky, and Dr. David Maxwell who I consulted on periodically for their medical expertise. Yuri and David were also very cooperative in acting as testers for various iterations of the interface.

Special thanks to Yuri for our many discussions on information retrieval. HEAL*Net* programmer, David Freese, also participated in many of these discussions. David patiently fielded my questions about C++, and often had plausible explanations for the strange behaviour of certain C++ compilers.

Also thanks to Robert Davies for providing the comprehensive Newmat matrix library to the C++ community, free of charge. Similar thanks to Dr. Michael Berry and his graduate students at the University of Tennessee who developed the SVDPACKC program, and made it publicly available.

Thanks to Alice for her love, patience, and support.

Dedication

To my father, Don Miller (1933 - 1985) who would have shared my enthusiasm for this topic.

Chapter 1

1. Introduction

The rate of growth of scientific information over the last few centuries staggers the imagination. Going back to the first known scientific journals in the 1600's, Price noted that the amount of new scientific literature published doubles every 15 years.[1] The Library of Congress now contains over 100 million items.[2] Online catalogs such as CARL Uncover provide access to over 17,000 periodicals. The widespread use of electronic publishing media, such as CD-ROMs, and the World Wide Web promise to further accelerate the publication of scientific and other literature.

Advances in data storage, communications and software technology have dramatically increased our ability to store, index, and move information. Yet, in spite of these advances, the avalanche of journal articles, books, and other works has buried many of us in information. Information overload has serious implications for health professionals and consumers. Physicians and other health professionals are not keeping abreast of advancements in knowledge reported in the literature.[3] This lack of knowledge can potentially compromise patient care. A resulting obstacle to optimal health care is that attempts to locate pertinent literature are often hindered by the vast number of articles, and a lack of effective tools to locate relevant information in document collections.

As the amount of scientific information has grown, so have efforts to manage and search bibliographic databases. These databases contain title, author, abstract and other information about the literature being referenced, but do not contain the complete text. Unfortunately, many people find it extremely difficult to find relevant information with current information retrieval (IR) systems.[4] The process of articulating an effective query is often hindered by the complexity of the query language or the ambiguity of the topic.

To add to the problem, the volume of information available to many IR system users is overwhelming. Many searches return a set of hundreds or thousands of candidate documents, which makes the task of sorting desired output from a query tedious and time consuming.[1,5] The critical problem is no longer how to acquire information, but how to store and retrieve it in such a way that one can find useful material.

1.1 Characteristics of Language

Were it not for the inconsistencies of natural language, IR would be a straightforward endeavor. All written and spoken languages are ambiguous by definition; this ambiguity poses serious problems for document indexing and searching. Three major characteristics of natural language complicate information retrieval: synonymy, polysemy, and term dependence. Synonymy refers to the characteristic that different words often have the same meaning. For example, the terms *high* and *elevated* may refer to the same concept.[6] One empirical investigation of the pervasiveness of synonymy in various domains found that two different people will use the same term when referring to an entity less than 20 percent of the time.[7] A parallel problem to synonymy is polysemy; the same word may have more than one meaning: *bank* of a river, versus a savings and loan *bank*. Terms exhibit dependence if the occurrence of one term depends on the presence of another. The phrase *diabetes mellitus* is an example of term dependence: it is difficult to think of a case where *mellitus* would occur in an article without *diabetes*. Examples of all three of these characteristics can be found in any document collection. Many efforts directed at building better information retrieval systems focus on one or more of these difficulties.

1.2 Variations on the Information Retrieval Problem

Conceptually, the larger problem of searching for information can be divided into three major sub-problems: ad-hoc search, relevance feedback, and routing. In an ad-hoc search the system executes a search based on a query and returns a set of candidate documents. In this paradigm, the only predictive information available to the system is

the user's query. In the relevance feedback problem there is more information available than in the ad-hoc search. Relevance feedback occurs in later stages of an ad-hoc search when a user has found one or more relevant documents and wants to find more. The IR system incorporates information into a new query from documents that the user has already judged relevant or not relevant in past queries. The routing problem refers to a situation where the user has already judged a number of documents to be relevant or not, and wants to find more relevant ones. In routing, the system has access to a growing profile of the user's interests which is built up over time. In this case the system attempts to classify new documents as relevant or not, using the information in the user's profile to predict whether the user will be interested in a new document. This thesis focuses exclusively on the ad-hoc search problem, although references will be made to relevance feedback and routing.

Marchionini[8] defines the ad-hoc search process in the following way: the user defines an information requirement, selects the source to be searched, and articulates the information need by formulating a query. After running the search, the user examines the results and either ends the search, or decides to reformulate the problem. This shift occurs because the results obtained in the last search iteration may provide some new insight that changes the information need. The key element in this paradigm, is that the information need is dynamic rather than static. Regrettably, many of the search tools that exist today are not very good at helping users define and articulate their information needs. Tools for accessing bibliographic databases of health literature are no exception.

1.3 MEDLINE: The Problem Setting

Many different health related citation indexes exist; the most commonly used one is MEDLINE. This database, produced by the U.S. National Library of Medicine (NLM) contains citations from approximately 3,700 international biomedical journals.[9] Citations are comprised of an article title, authors, abstract, journal, and other publication information. The entire MEDLINE collection contains over eight million citations, divided chronologically into several more manageable pieces. Citations relating to clinical medicine make up the core of MEDLINE but the collection also covers many

related fields such as pre-clinical sciences and administration. MEDLINE articles are currently indexed manually using a hierarchical controlled vocabulary called Medical Subject Headings (MeSH). The MEDLINE database can be searched with the MEDLARS search tool provided by the NLM, or a third party tool.

MEDLINE articles are indexed by manually tagging each article with a canonical set of terms from the vocabulary of MeSH codes. These codes can then be incorporated as part of a search to find a particular article or set of articles. The MeSH vocabulary restricts the character and number of terms that can be used to index a document. Like other controlled vocabularies, MeSH is very large and volatile. The 1994 version of MeSH contains over 17,000 terms and underwent over 1,000 changes in a single year.[10]

Indexing consistency, incomplete coverage and high costs are three major problems that plague the process of indexing articles in MEDLINE. Funk and Reid found that the inter-indexer reliability of using MeSH terms to index articles for MEDLINE documents is only 30 to 60% depending on the depth of indexing detail examined.[11] The MeSH vocabulary can not be modified fast enough to accomodate rapid changes in health care concepts. For example, in the mid 1980's, as the terminology for AIDS grew, MeSH lagged several years behind.[6] The lack of indexing reliability with MEDLINE and limitations to the comprehensiveness of MeSH presents an enormous obstacle. The performance of any IR system depends on the consistency of indexing. In addition to inconsistency problems, the cost of manual indexing is substantial: the National Library of Medicine alone spends over two million dollars per year to manually index MEDLINE articles.[12] For these reasons, MEDLINE users as well as the NLM are interested in alternatives to manual indexing.

The MEDLINE database is available on-line and via CD-ROM from more than a dozen different vendors. The NLM reported in 1994 that over 6.9 million searches were conducted on its own system by over 100,000 users.[6] As access to MEDLINE has become more widespread, the user base has changed. Traditionally, medical librarians were the primary users of MEDLINE. Physicians sent a written information request to a librarian, who translated the query into an appropriate set of MeSH codes and performed a search using MEDLARS. With the widespread availability of computing resources, an

increasing number of physicians, researchers, students and other non-librarians now perform their own MEDLINE searches. This shift in clientele has an important implication on the queries that are posed, and the quality of search results. Unlike medical librarians, today's MEDLINE users are not fluent in MeSH, the key to articulating effective queries with MEDLARS.[13] Experimental results suggest that searches made by novice MEDLINE users produce significantly worse results than those made by medical librarians.[14]

The types of searches performed on MEDLINE vary depending on who is doing the searching and the purpose of the search. Clinicians, nurses and other professionals in clinical settings are often looking for specific information that will help to satisfy immediate patient care needs. Hersh and Hickham found that searches for articles addressing therapeutics and diagnosis along with review articles were the most common types of IR requests in a clinical setting.[15] Researchers in academic settings are more likely to use MEDLINE to find information relevant to ongoing projects or publications in progress.

Two approaches have been taken in the past to deal with the difficulties that physicians and other non-librarians have with using MEDLARS; (1) teaching physicians and other end users to effectively incorporate MeSH into their searches, and (2) developing better retrieval engines and interfaces. Attempts at the first approach have shown that physicians and other health professionals are unwilling to spend much time learning MeSH.[16] Consequently, this thesis will focus on alternative search strategies and interfaces for searching MEDLINE and related collections.

1.4 State of the Art Information Retrieval

Although commercial systems for searching MEDLINE have changed little in the last few decades, research in the broader field of IR has flourished. Some of the fruits of this research offer effective alternatives to manual indexing, controlled vocabularies, and unmanageable output. Newer models based on statistical theory offer many improvements over Boolean methods currently in use by most commercial systems. Very recently, IR research has become increasingly focused on enhancements that harness the

user's own knowledge and capacity for classification and pattern recognition. Relevance feedback and visualization methods are two examples of such enhancements. Relevance feedback can be thought of as a query-by-example, where the user essentially says, "find more articles like this one". Visualization tools display results of a search, and other information, graphically instead of in a ranked list. Visualization allows a user to view a large amount of information at once in a more meaningful way than traditional ranked lists. In time, relevance feedback and visualization tools may become part of commercial systems for searching MEDLINE and related collections.

1.5 Goals and Problem Statement

This thesis has two major goals. The first is to identify and evaluate alternative methods of indexing and searching MEDLINE databases. Ideally, such methods should address ambiguities of language without employing external information such as controlled vocabularies. Although the primary problem of interest here is the ad-hoc search, an ideal method should at least support relevance feedback.

Another problem is that sifting through the results of an information search is very time consuming and difficult. Thus, the second goal is to devise and evaluate a method for presenting search results in an alternative fashion to a ranked list. Such a method should free the user from the constraints of list-based output.

This thesis addresses two research questions. The first question is whether a method for IR visualization developed by the author, called Local Latent Semantic Indexing / Cluster (LLSI/Cluster), consistently clusters relevant articles together in three dimensional space. The second question is to what degree does LLSI/Cluster improve the accuracy with which relevant articles can be found compared to two list-based retrieval methods.

1.6 Outline of Chapters

The current chapter presents the information retrieval problem, and explains why it is a critical problem for the health field. Some of the problems inherent in current

methods of dealing with this issue, such as controlled vocabularies and manual indexing, are presented. This thesis focuses on the ad-hoc search problem, an iterative process in which the user defines an information need in the form of a query.

Chapter two presents basic IR terminology and methods. Fundamental definitions and the essential elements of the indexing process are presented. Test collections and performance measures are also introduced in this chapter. In chapter three, various search methodologies are presented and critiqued with respect to the goals outlined in the introduction. The third chapter covers common search methods used in IR such as the Boolean model, the Vector Space Model (VSM) as well as linguistic and knowledge-based approaches, and Latent Semantic Indexing (LSI). In chapter four, existing methods for visualization are compared and critiqued. A new methodology for information visualization is presented called Local Latent Semantic Indexing / Cluster (LLSI/Cluster). Chapter five describes a retrieval system developed by the author based on the LLSI/Cluster model. Chapter six formally states the research problem under investigation, and presents specific hypotheses. Chapter seven outlines the methodology for three experiments that compare LLSI/Cluster, LSI and the VSM. Chapter eight describes the results of these experiments. In Chapter nine, major findings are discussed, and interpreted within the framework of this study as well as with respect to existing research. Implications for current theory and practice are also discussed, along with limitations that affect the generalizability of findings. Chapter 10 concludes the thesis with a summary of key points.

Chapter 2

2. Background

The first goal of this thesis is to assess the merits of different methods for automatically indexing document collections. In order to appreciate the theoretical and practical issues involved, one must have an understanding of basic techniques and terminology used in information retrieval. This chapter provides some fundamental IR definitions, and describes a typical indexing process of converting raw text into tokens. Commonly used methods of assessing the performance of IR systems are also presented.

2.1 Information Retrieval Definitions

Information retrieval systems use a wide range of methods to index documents, receive input from users, and estimate the similarity between queries and documents. However, all IR systems have the following processes in common:

1. Documents are translated into simplified representations that retain the essence of their content. This translation is traditionally done by a person using a controlled vocabulary as a reference, although it may be done automatically by an indexing program.

2. To retrieve documents, users describe information needs in a query. Queries are typically in free text (natural language form), although users may include terms from the controlled indexing vocabulary to enhance retrieval.

3. There is a matching process where the system compares the query to the documents in the collection, and the system returns a set of candidate documents to the user.

A document is a unit of information in the collection, e.g., a journal article abstract. Document collections may be static or dynamic. Examples of static collections are archives, encyclopedias and other collections that do not grow or change often over time. In dynamic collections, such as news groups, documents are constantly appearing and retiring. Collections are often comprised of thousands or even millions of documents.

A query is the user's information request articulated as a text expression. Queries may be in the form of simple phrases, complete sentences, or even entire documents. The retrieval system uses the terms in the query to determine those documents in the collection to return to the user. The user may reformulate their query based on feedback from a previous search.

2.2 The Indexing Process

Indexing a document collection typically involves the following components, lexical analysis, an inverted index, a stop list, and stemming. Lexical processing includes -- but is not limited to -- converting raw text into a set of tokens,[i] converting text to a single case, and removing punctuation. Once the lexical analyzer has finished filtering text into tokens, each useful term can be stored in an index, a separate database table linking documents to different terms. This inverted index allows for efficient searching of the document base. The inverted index may also contain information about each term's importance and frequency of occurrence within a particular document.

In any collection there are a large number of words that carry little or no semantic content on their own. For example, words such as "a", "the", "and" are not worth indexing. A list of such words, called a stop list, can be used to filter out non-content bearing terms and reduce the overall size of the index. Stop lists may be either domain

---

[i] Terms, acronyms, numbers, and punctuation are all examples of tokens.

independent or tuned to a particular sphere of knowledge such as medicine. The size of the stop list may vary from less than 40 terms to over 500. Evidence suggests that there is no retrieval performance penalty for using large stop lists.[17] The advantage of using a larger stop list is that it can greatly reduce the overall size of the index. Shorter stop lists ensure that more terms are included in the index but do not necessarily lead to better retrieval performance.

Many terms used in composition are syntactic variations on a root form; e.g., cancer, cancers, cancerous. A commonly used procedure in IR called stemming attempts to reduce terms to this root form. This strategy is based on the assumption that most common morphological variants of a word have similar meanings and are used in similar contexts. Stemming can help improve the coverage of a search while reducing the size of the index at the same time.

Experimental evidence suggests that while stemming may not definitively improve performance, it does not appear to degrade performance either.[18] A small number of stemming algorithms are in wide use, but there does not appear to be significant differences in their impact on retrieval performance using the resulting indices.[18,19] A disadvantage of stemming is that it is error prone; root words may be stemmed, or words may be stemmed incorrectly. In some cases the user may want to make a distinction between the a term and its root form; stemming blurs these distinctions.

2.2.1 Test Collections

IR researchers have constructed several test collections for objective comparisons between different IR methodologies and systems. Test collections consist of a set of documents and queries. The document set contains some items that human experts have judged relevant to each query and some that are not. Early test collections were relatively small, containing on average 1,000 documents, and a handful of queries. Recently developed collections such as OHSUMED[20] and TREC[21] are more representative of a typical document collection. Both collections contain hundreds of thousands of

documents. MEDTEST (2,344 items) and OHSUMED (ca. 350,000 items) are two test collections created from MEDLINE abstracts that are available on the Internet.[i]

2.2.2 Performance Measures: Precision and Recall

Precision and recall are measures used to objectively evaluate IR system performance. The ratio of relevant documents retrieved divided by the number of documents retrieved is called precision (see Equation 2-1). Recall is the ratio of relevant documents retrieved divided by the number of relevant documents in the entire collection (see Equation 2-2). Therefore, precision is a measure of search accuracy while recall assesses the thoroughness of a search. In many cases, there is a trade-off between recall and precision: adjustments to a retrieval system that improve recall tend to negatively affect precision, and *vice versa*.

$$\text{precision} = \frac{\text{Number of relevant documents retrieved}}{\text{Number of documents retrieved}}$$

Equation 2-1

$$\text{recall} = \frac{\text{Number of relevant documents retrieved}}{\text{Number of relevant documents in the collection}}$$

Equation 2-2

The most common display of retrieval performance is known as the precision-recall curve; it is a scatter-plot of precision by recall, evaluated at each point in the ranking where a new relevant document is discovered. Alternatively, precision and recall can be measured at a fixed document cut-off value (DCV). Each query generates a different precision-recall curve. A system's performance can be judged by averaging these curves. If a DCV is used, the results can be averaged over all queries. Averaging curves

[i] These collections are available from: ftp://medir.ohsu.edu/pub/ohsumed/

that plot precision by recall is more complicated, because the defined recall levels are different for each query.

Macro-evaluation is a common strategy for averaging precision-recall curves.[22] In macro-evaluation, precision is measured at fixed levels of recall for each query. Recall is often set at 10 percent intervals. An averaged recall precision curve can be built by taking the average precision over all queries measured at each of these recall points. In order to ensure that the fixed levels of recall are defined for each query, it is necessary to perform some interpolation of the individual precision-recall curves. Using the average precision-recall curve, the performance of different systems can be compared.

Figure 2.1 illustrates a typical precision-recall graph using results from an experiment with 75 queries from the MEDTEST collection. For this example, precision is plotted at 0.1 intervals of recall. The Figure illustrates the tendency for precision to decrease as recall increases.

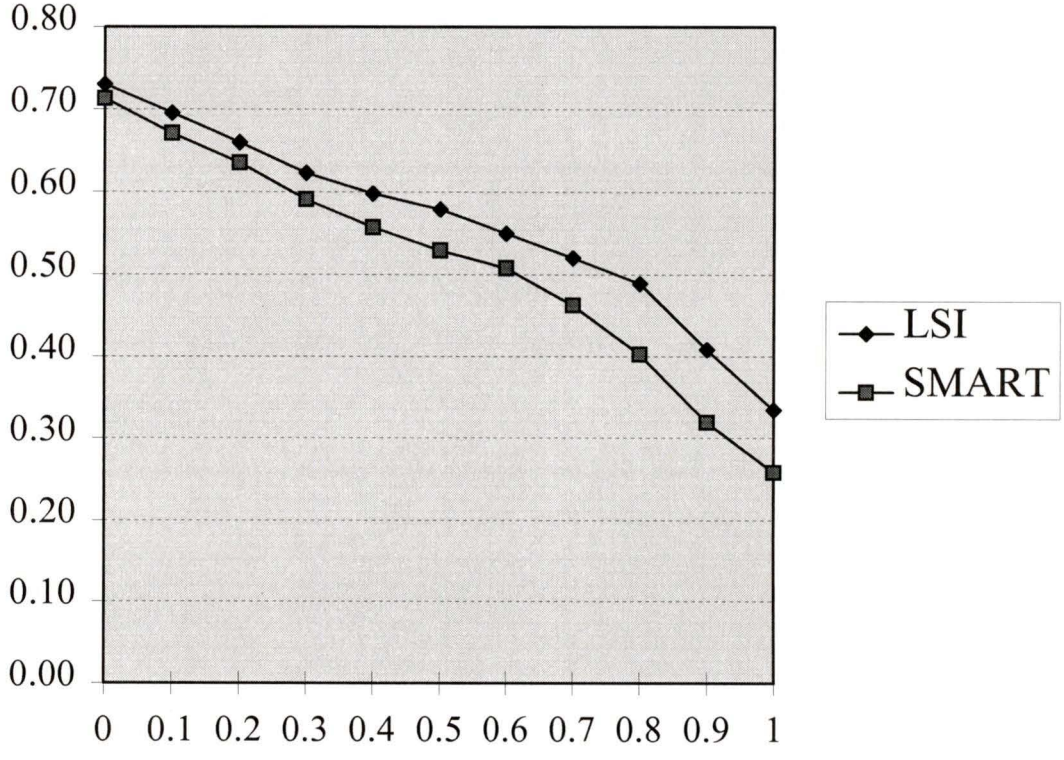


Figure 2.1: A typical recall-precision graph

Chapter 3

3. Search Strategies in IR

In the previous chapter, basic IR definitions, storage methods, and pre-processing procedures are described. The current chapter introduces four candidate search strategies, or retrieval engines that have been used or could be used for indexing MEDLINE articles. The Boolean model, Vector Space Model (VSM), linguistic and knowledge-based methods, and Latent Semantic Indexing (LSI) are each described along with their benefits and drawbacks.

3.1 Boolean Retrieval

The Boolean model uses Boolean logic to search document collections. Systems such as MEDLARS that handle Boolean searches are often indexed with the aid of a controlled vocabulary. In Boolean retrieval, queries are formulated with the logical operators AND, OR, and NOT. In set theory, these represent intersection, union, and complement respectively. An example query `hypertension OR high blood pressure AND risk` corresponds to the statement, "find all articles that contain the term `risk`, and either `hypertension`, or `high blood pressure` or both". The Boolean model is simple to implement and Boolean searches are fast. The Boolean method is often used to search large document collections such as library systems.

Despite its enduring popularity, the Boolean model suffers from significant limitations. There is no way to indicate term importance in Boolean queries or searches. This property makes it very difficult for the user to optimize a query because the addition of even a single term can drastically alter the set of articles returned to the user. Lack of term importance also means that the system can not rank results in order of likelihood of interest; the user must peruse the output to find meaningful articles among irrelevant

ones. If a relevant document is found, there is no way for the system to incorporate information from that document into subsequent queries.

3.2 The Vector Space Model

Research conducted by Luhn[23] has developed into the Vector Space Model (VSM). Luhn's research was motivated by a concern that controlled vocabularies and classification schemes would change over time. Luhn foresaw that the concepts used to classify a document at a particular time could be different from those used to classify the same document at another time. The relevance of previously indexed documents would therefore be lost in the newer conceptual framework. Luhn proposed the use of a "notion-space" to alleviate the problems of synonymy and polysemy. He suggested that each document should be represented in the notion-space by an array whose elements are comprised of the various ideas or concepts in the document collection. The process of searching the collection could be carried out by encoding notions in the query like those in the documents, and then comparing the encoded query to the encoded documents in the space.

In the 1960's Gerard Salton initiated the SMART project, a retrieval system based on the VSM.[24] SMART continued to evolve into the mid 1990's, and is now widely regarded as the gold-standard of document information retrieval systems.

The VSM represents documents and queries as unit length vectors of term weights. Each element of a document vector indicates the importance of a specific term contained in that document. An example term/document matrix is depicted in Figure 3.1.

	documents				
terms	h1	h2	h3	h4	...
prevent	0.25	0.25	0.25	0	...
hypertens	0	0.62	0	0.62	...
report	0	0	0	0	...
group	0.17	0.17	0	0	...
high	0.12	0	0	0	...
...	...	...	...	...	...

Figure 3.1: A partial term-document matrix

The value of these elements may be binary to reflect the presence or absence of a term in the document. Alternatively, the elements may represent the frequency of the term within a given document, or a more complex term weight. Typically a real number term weight, rather than a the simple occurrence or term frequency is used. Common term weighting algorithms are presented in the next section.

In the VSM, queries are also represented as a vector of terms, much like a document. The similarity of a query to a document can be measured by taking the dot product or cosine of the angle between them.

$$\cos(q_i, d_j) = \frac{\sum (q_{ik} \cdot d_{jk})}{\sqrt{\sum (q_{ik})^2 \cdot \sum (d_{jk})^2}}$$

Equation 3-1

where q_{ik} and d_{ik} give the value of the term k in query q_i and document d_j respectively.

A more precise term for the measure given in Equation 3-1 is the *dot product*. The dot product measures the angle between the two vectors; a smaller angle between two vectors indicates greater similarity. The dot product is similar to the correlation coefficient, but unlike with correlation coefficients, the values are not standardized. A VSM system presents documents to the user in the order of highest to lowest document to query similarity value. The user can control the size of the retrieved set by setting an arbitrary cutoff point.

3.2.1 Term Weighting

In any collection of documents, some terms are likely to be more important than others. A critical feature of the VSM is that is allows this information to be utilized through the assignment of representative weights to each term that is indexed. Term weights fall into two main categories; (1) term frequency weights, and (2) term importance weights.

Term frequency weighting (TF) is based on the premise that frequently occurring terms are highly representative of the content of a particular document, while incidental words are not representative. This principle is embodied in Zipf's law, which states that the frequency of a given word multiplied by its rank in a list of all terms in the collection is approximately constant.[25] A simpler way of phrasing Zipf's law is that in an average collection, there is a small percentage of terms that are used much more often than one would expect at random. For instance, in many health related collections, terms such as `patient` and `procedure` appear in many documents, while words such as `cholangiopancreatography` appear much less frequently. Table 3.1 illustrates the principle of Zipf's law. The top ten frequently occurring terms in this example make up almost one fourth of the 1,000,000 occurrences in the example. The number of times a term occurs multiplied by its rank in the list is a approximately a constant number.

Table 3.1: Illustration of Zipf's Rank-Frequency Law[i]

(Number of word occurrences = 1,000,000)

Rank (R)	Term	Frequency (F)	R * F
1	the	69,971	69971
2	of	36,411	72822
3	and	28,852	86556
4	to	26,149	104596
5	a	23,237	116185
6	in	21,341	128046
7	that	10,595	74165
8	is	10,099	80792
9	was	9,815	88335
10	he	9,543	95430

Because term importance does not increase linearly, term frequency weights are typically transformed logarithmically. A VSM retrieval system can capture term frequency information in queries as well as documents. Term frequency weights (TF_{ij}) are a measure of the frequency in which term *i* occurs in document *j*.

$$TF_{ij} = \log(\text{frequency of term } i \text{ in document } j) + 1$$

Equation 3-2

A term's importance not only depends on the frequency of its occurrence within a document, but within the collection as a whole. For instance, a frequently occurring term like `patient` makes a very a poor discriminator between documents. Rare terms, such as `cholangiopancreatography`, are more valuable for discriminating between documents.

---

[i] Adapted from Salton G, McGill MJ. *Introduction to modern information retrieval.* New York: McGraw-Hill. 1983.

One of the more commonly used methods of term importance weighting is called the Inverse Document Frequency (IDF) measure (see Equation 3-3).[26]

$$IDF_i = \log\left(\frac{\text{Number of documents in collection}}{\text{Number of documents that contain term } i}\right) + 1$$

Equation 3-3

Both TF and IDF term weighting schemes are usually applied in concert to produce a composite term weight.

$$WEIGHT_{ij} = TF_{ij} * IDF_i$$

Equation 3-4

The composite weight measure defined in Equation 3-4 assigns the highest weights to those words that occur most frequently in a document but infrequently throughout the rest of the collection. Consequently, in the VSM, these terms are considered the best discriminators of individual documents.

3.2.2 Enhancements to the Vector Space Model

Other extensions to the VSM include query expansion, word sense disambiguation, and phrase modeling. Query expansion is the identification of additional terms that are synonymous to terms in the query using an external thesaurus or other means. Word disambiguation is a method for deriving the intended meaning of homonyms in the query using dictionaries, morphological analysis or other methods. Experimental evidence suggests that word sense disambiguation rarely improves performance and sometimes degrades it.[17] Because the VSM assumes term independence, some researchers have experimented with syntactic and statistical methods for identifying phrases that are commonly used in a collection. This process is called phrase modeling. There is evidence that statistical phrase modeling improves precision moderately, but at a high storage cost.[22]

3.2.3 Benefits/Drawbacks

Unlike the Boolean Model, it is possible to rank the output of a VSM search, since each document can be sorted according to its degree of similarity to the query. Those documents most likely to be relevant, are often at the top of the list. Queries are less sensitive to the user's choice of a particular term than in the Boolean model because only highly weighted terms will significantly alter ranking. The VSM can also handle queries based on long text passages. Unlike Boolean searching, it is possible to incorporate relevance feedback.

A drawback of the VSM is that it is difficult for the user to understand how the system ranks results. In a case where the top-ranked articles are all relevant, this property may not be a drawback. However, when the system fails to rank highly relevant documents appropriately, or does not find them at all, the user may be at a loss as to the reason for such a failure.

3.3 Linguistic and Knowledge-based Information Retrieval Methods

Despite the proven success of the Vector Space Model for indexing and retrieval, another faction of IR researchers maintain that word-based statistical methods have merely, "picked some of the low hanging fruit off the tree,"[27] and that a deeper semantic interpretation of texts and queries will be required before information retrieval can reach its full potential.[28] These concept-based methods can be subcategorized as linguistic or knowledge-based. Both techniques deserve mention here because they have received considerable attention from researchers in medical informatics.

Linguistic analysis involves assigning one or more syntactic "tags" to each word in a document, and then constructing phrases that identify noun and adjective combinations useful for the identification of semantic content.[26] Linguistic approaches to IR have the potential to handle ambiguous phrases that may be misinterpreted by word-based approaches. For example, documents on topics other than *hypertension* may contain the three words that comprise its synonym: *high*, *blood*, and *pressure*, but in a

completely different context, e.g., Ocular *pressure* was *high*...despite adequate *blood* levels of medication. Unfortunately, syntax alone does not account for many of the ambiguities in natural language, and the accuracy of current syntactic algorithms is inadequate.[17] False constructs can be erroneously tagged as content bearing phrases and useful phrases that correctly reflect content may not be tagged because of the constraints imposed by the linguistic process. The theoretical benefit of syntactic parsing has not translated into significant improvements in retrieval in practice.[29,30]

Knowledge-based methods of IR have attracted much attention in health care disciplines, yet they too have failed to improve on simpler methods. A knowledge base provides a mapping between the concepts commonly used in a domain, and their representation in an index. Typically, a knowledge base includes a description of major concepts of interest, as well as information about how the concepts are related. Numerous representation schemes have been developed for knowledge bases, including conceptual graphs, frames, scripts, and cases. Often a network structure is used to represent the relationships between concepts. Knowledge bases are used in conjunction with rules that specify how the conceptual network is traversed. Many different approaches have been tried; the general idea is to apply a particular semantic structure designed for a specific field and combine it with some form of rules for traversing the structure. Examples of some of these techniques are semantic nets,[31] advanced linguistic processing techniques,[32, 33,34] and Bayesian inference.[35] In each approach, the goal is to estimate the similarity of a query to those documents in the collection based on the degree to which the query can be satisfied from the available documents.

Linguistic and knowledge-based approaches to IR sometimes perform successfully in very limited and well defined areas but they have not scaled up well to broader domains.[36] Linguistic parsing has proved far too complex a problem for current technology. The use of large knowledge bases for information retrieval is constrained by a lack of proven methods for determining what knowledge is actually needed to describe concepts in a particular field, how to best represent knowledge, how to separate domain knowledge from a seemingly unlimited context, and how to efficiently update existing knowledge bases.[37]

3.4 Latent Semantic Indexing

A major problem with Boolean and VSM models is that they do not directly address the problems of synonymy, polysemy, and term dependence. To compensate, the VSM can be supplemented with other methods such as query expansion or phrase modeling. Phrase modeling adds slight improvement at increased cost in complexity and storage. External thesauri not only add more complexity to an IR system, but also introduce their own limitations. Thesauri must be manually updated as new concepts in the field emerge and existing ones evolve.

Linguistic and knowledge-based methods of IR address synonymy and polysemy problems directly. Despite the appeal of linguistic parsing and knowledge-based approaches, systems based on these principles have failed to improve precision and recall over simpler word-based models such as the VSM.

In the 1960's Borko and Bernick[38] followed up on another of Luhn's ideas: to index documents by notion rather than individual terms alone. The researchers used a multivariate dimension reduction algorithm called Principle Components Analysis (PCA) to create a small number of orthogonal indexing vectors. The authors used existing document scores on each of the indexing vectors to classify new documents into a set of categories with moderate success.

In the late 1970's, Koll[39] expanded on the ideas of Borko and Bernick by using PCA for ad-hoc information retrieval. Koll placed documents at the centroid of the locations of their constituent terms, and terms at the centroid locations of their constituent documents. Queries were treated in the same way as documents; Koll used the Euclidean distance between the query and each document to create similarity rankings. Average precision over all recall points for the system on a small test collection was comparable to a VSM system.

Latent Semantic Indexing (LSI) builds on the research of Luhn, Borko and Bernick, and Koll. LSI is an extension of the Vector Space Model that derives orthogonal indexing constructs from the patterns of word use throughout an entire collection.[40] The reasoning behind LSI is that there is some "latent" structure in the patterns of word usage

that is partially obscured by the variability of word choice. LSI creates a reduced representation of the term-document space. With LSI, documents with similar term usage patterns are located in similar regions in multidimensional space. This allows for the possibility of a document being highly ranked in response to a query even if the query and the document do not share any terms.

LSI differs from earlier IR work with PCA in a number of ways. Unlike the work of Koll or Borko and Bernick, LSI uses many more constructs to represent the ideas or constructs latent in the collection. A construct is a non-technical name for one of the orthogonal dimensions or axes generated by the LSI process. Unlike PCA or Factor Analysis, LSI represents both terms and documents in the same space and no attempt is made to interpret the LSI constructs. Due to the computational constraints at the time, the efforts of Borko and Bernick, and Koll were restricted to working with very small collections. With enough processing power and memory, it is now possible to implement LSI on collections containing 100,000 documents.[41]

The dimension reduction algorithm at the heart of LSI is called the Singular Value Decomposition (SVD). The SVD is used in the computation of Principle Components Analysis (PCA), discriminant analysis, and other multivariate statistical procedures.[42] For the purpose of information retrieval, the SVD is a technique for deriving a reduced set of orthogonal indexing constructs from the original term-document matrix. A term is any token not filtered out at the indexing stage. The constructs may be thought of as artificially constructed concepts that reflect the common themes in documents throughout the collection. Unlike real concepts, LSI constructs are not directly interpretable. Each term or document is characterized by a vector of weights that indicates the strength of its association with each of the constructs.

Unlike many other IR models, LSI represents each document and term as a vector in a reduced dimensional space. To return a set of documents that are similar to the query, the query is transformed into the reduced space, and then compared against all documents. The following section outlines the mathematics of LSI. Appendices A and B contain a detailed illustration of LSI using a small collection of documents.

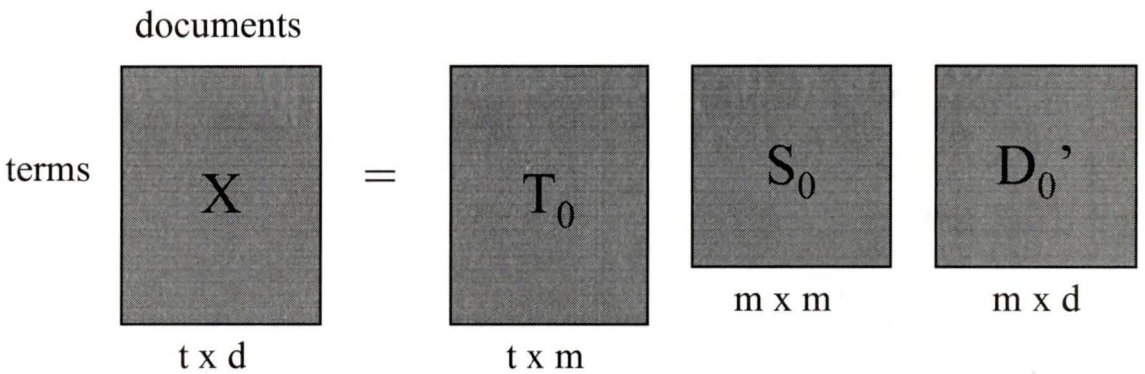
3.4.1 A Mathematical Description of Latent Semantic Indexing

This section illustrates the linear algebra on which LSI is based. The SVD model states that any rectangular matrix, X, a t x d matrix of terms by documents, X, can be described by the product of three matrices:[43]

$$X=T_0S_0D_0'$$

Equation 3-5

such that T_0 and D_0 have orthonormal columns[i] and S_0 is a diagonal matrix[ii]). SVD solutions are unique, and by convention the diagonal elements of S_0 are constructed so as to be positive and ordered from largest to smallest. Figure 3.2 illustrates the SVD.



Singular value decomposition of the term-document matrix, X. Where:

T_0 has orthogonal, unit length columns ($T_0' T_0=I$)
D_0 has orthogonal, unit length columns ($D_0'D_0=I$)
S_0 is the diagonal matrix of singular values.[iii]
t is the number of rows of X
d is the number of columns of X
m is the rank[iv] of X (min (t, d))

Figure 3.2: Full Rank Singular Value Decomposition

---

[i] The columns are uncorrelated with each other, and the square root of the sum of squared elements is 1.

[ii] The values on the diagonal of S_0 are called the singular values, and the off diagonal values are zeros.

[iii] The singular values correspond to the square roots of the non-negative eigenvalues of $X'X$.

[iv] The rank of a matrix corresponds to the number of linearly independent (uncorrelated) columns or rows in the matrix.

For $X=T_0S_0D_0'$ the matrices T_0, D_0, and S_0 must all be of full rank (they can not be truncated). However, because the diagonal elements of S_0 are ordered according to size, the SVD allows for an optimal approximate fit using smaller, truncated T_0 S_0 and D_0' matrices. The first k largest singular values in S_0 may be retained, and the remaining ones set to 0. The product of the resulting matrices is a matrix, $\hat{X}$ which approximates X and is of rank k. The matrix $\hat{X}$ is the matrix of rank k which retains as much of the variance accounted for by X as possible. Because zeros were introduced into S_0, the S_0 matrix can be simplified by deleting all but the non zero rows and columns to obtain S, and then deleting the corresponding columns of T_0 and D_0 to obtain T and D respectively. The result is the reduced model:

$$X \approx \hat{X} = TSD'$$

Equation 3-6

This model is the rank k least squares approximation to X. That is, although T, S and D' now have fewer columns, they are guaranteed to be the best possible approximation to X at k columns while minimizing the residual error.[i]

In LSI, as in Factor Analysis, PCA, and other models employing the SVD, there is an issue of the number of dimensions to retain. Setting the optimum value of k is important. There should be enough constructs to fit the real structure inherent in the data, yet the number should be limited to omit sampling error and unimportant relationships.

3.4.2 Projecting Queries into the LSI Space

Queries can be thought of as documents in the LSI model in a manner similar to the VSM. In LSI a query must be projected into the reduced dimensional space before it can be compared with other documents. This involves placing the query at the centroid of its constituent terms.

---

[i] For a formal proof, see endnote 43.

$$d_q = X_q' T S^{-1}$$

Equation 3-7

where:

d_q is the location of the query in LSI document space DS
X_q' is the transpose of a raw query vector containing term weights for the terms in the query and zeros otherwise
T are the right singular vectors (matrix of terms by constructs)
S^{-1} is the inverse of the singular values matrix

Once the query is in the same space as the documents, the dot product (see Equation 3-1) may be used to calculate the similarity of the query to all documents and rank the results accordingly.

3.4.3 Adding Terms to the Latent Semantic Indexing Model

In order to minimize processing time and memory requirements, it is common practice to include only terms that occur in more than one document in a collection in the indexing process. Since singleton terms account for very little variance, their influence on the solution after truncating the matrix of singular values is negligible. These singleton terms can be folded back into the LSI term space later, in the same way that queries can be folded into the document space:

$$t_q = X_d D S^{-1}$$

Equation 3-8

where,

t_q is a folded-in singular vector of the same length as the number of LSI constructs.
X_d is a raw vector of weights for the term across all documents
D are the left singular vectors
S^{-1} is the inverse of the singular values matrix

The result of the calculations in Equation 3-8 is that the new term is placed in LSI space at the centroid of the terms in the documents that the new term occurs in. This has the effect of associating the new term closely with the documents that contain it.

3.4.4 Selecting the Ideal Number of Constructs

Constructs are the orthogonal dimensions derived with the SVD. These constructs account for successively smaller portions of variance in the term and document spaces. After some point, the constructs no longer account for useful information and need not be included in the model. The ideal number of constructs to retain in an LSI model is still an open problem. In the LSI model, retaining too few constructs omits useful information while retaining too many results in noise being included in the model. Unfortunately, existing heuristics for choosing an ideal number of constructs are unlikely to help with LSI.

In other disciplines where Principle Component Analysis (PCA) is used, a number of methods exist for choosing the number of factors to retain. The primary goal with these methods is to derive interpretable constructs. With LSI the goal is not to interpret the constructs but merely to use them to create an approximation of the original term-document matrix to overcome synonymy, and term dependence problems. Therefore, heuristic methods such as the Kaiser-Guttman rule (eigenvalues > 1), the scree plot, the broken stick method, or proportion of total variance method are not appropriate for LSI.[44] Statistical methods based on examining differences between eigenvalues such as tests for sphericity and Barlett's test of eigenvalue are also unlikely to be of much help. This is because the eigenvalues from an LSI solution decay very smoothly after the first few.

The ideal number of constructs to retain depends not only on the document collection, but the nature of queries that will be applied to it. Queries containing very specific terms tend to perform better when more constructs are retained, whereas queries with a large number of imprecise words may benefit from the imprecise mapping of terms to documents. With test collections, the number of constructs to retain in an LSI model is typically decided by measuring retrieval performance over a set of test queries using several models. With a non-experimental collection, this type of testing requires the creation of a set of test queries. Generating such a test set and applying it to successive LSI models with different numbers of constructs is a time consuming process. With large

collections, the maximum number of constructs is limited by constraints imposed by the computer platform.

3.4.5 Benefits / Drawbacks

Despite its complexity, LSI offers a number of advantages over other search strategies. It has the potential to account for synonymous terms without incorporating a manually constructed thesaurus. Term dependencies are modeled intrinsically by incorporating words into constructs that reflect correlations in their usage across multiple documents.

Experimental evidence suggests that these properties help in practice. Deerwester, Dumais, Furnas, Landauer and Harshman found that LSI performed up to 30 percent better than VSM on two small test collections.[45] Foltz found LSI comparable to the VSM for filtering technical memos.[46] Using an extension of the model that applies a second SVD on the local region of relevant documents, Hull achieved significantly better recall and precision than the VSM in a series of filtering experiments.[47,48] In experiments with the very large TREC test collection, Dumais reports that LSI fared slightly worse than the VSM in TREC-2, and slightly better than the VSM in the TREC-3 evaluations.[49,50]

Just as with the VSM, the LSI model ranks documents in order of their estimated relevance to a query. LSI does not require a user to learn a complicated searching vocabulary or syntax. One of the more useful aspects of LSI over the VSM is that it reduces the length of document vectors. This property is especially useful for visualization, as we will see in the next chapter of this thesis.

There are also some limitations to the LSI method. The initial SVD of a term-document matrix is computationally intensive, $O(k^2 * m)$; where k is the desired number of constructs and m is the rank of the original term document matrix.[51] This cost only needs to be incurred once for static collections. LSI also has a high storage cost because, unlike the VSM, the approximated term document matrix is no longer sparse, its elements are floating point numbers. Unlike the VSM, queries must be compared with each document in the entire collection because all documents are similar to any query to some extent. In terms of theoretical limitations, LSI may be less effective than the VSM when

there are only a small number of terms that reliably indicate relevance to a particular topic. The reason for this is that LSI only detects these terms indirectly through their relationships with the construct vectors. Finally, LSI suffers from the same "black box" problem as the VSM; LSI makes it difficult for the user to tell how or why the system chose one document over another.

Chapter 4

4. Visualization Techniques in Information Retrieval

The previous chapter discussed the advantages and drawbacks of major search strategies. LSI is the only one of those methods to inherently address synonymy, polysemy and term dependence. Those advantages are offset by disadvantages with both LSI and the VSM.

First, ranked lists can get very long and difficult to navigate. Often the output must be broken up into a series of "pages" which the user scrolls through. Second, it is difficult to gauge the strength and nature of the relationship between a retrieved document and its query. By condensing "similarity" to a single value, a retrieval system artificially imposes a one dimensional measure of "likeness". In fact, documents may be similar in a variety of ways. For instance, articles that fit the query, `side effects of cyclosporine`, may differ on the type of subjects used (animal or human), the experimental design (randomized clinical trial or case study), and the particular side effects of interest. From a user's point of view, one would like to see the similarity between documents represented along multiple dimensions. The single dimensional similarity scores also make it difficult to tell how far down the list one should look for relevant articles.

Another problem with ranked lists is that there is no clear indication of the relationships between retrieved documents. If the user locates an interesting article, there is no way for her to quickly locate similar articles in the context of the current query. Finally, a user can not determine how well her query discriminates between relevant and irrelevant documents. The user must correct her query without feedback in a case where

none of the top-ranked documents are relevant. Tools based on the techniques described in this chapter for visualizing query results can address some of these problems.

The goal of this chapter is to describe a method for visualizing the results of searches based on the LSI model. Research in visualization methods arose from a need to address the limitations of ranked output. A number of techniques, such as reference points, spring embedders, and clustering have been applied to the VSM and other models A novel method called LLSI/Cluster is also presented. This section describes each method and discusses the benefits and drawbacks of each.

4.1 Reference Points

The reference point approach is one of the first attempts to incorporate visualization into an IR system interface. With this method, a space is defined for a small number of reference points and document objects are plotted within this space based on their triangulation to the points. Korfhage[52] devised a reference point visualization system called VIBE based on the VSM. In this system, a query, known relevant documents, and other information are the reference points about which the other documents can be positioned. Dubin[53] explored VIBE as a term selection strategy by using individual terms as reference points Hemmje, Kunkel, and Willet[54] describe an extension to VIBE called LyberWorld which displays documents and terms in three dimensions. Figure 4.1 shows an example VIBE display using an example query, `risk for myocardial infarction and hypertension`. The term nodes determine the positioning of the hypothetical relevant documents, one, two, three and four. For instance, document one is related to `myocardial` and `risk`, but none of the other reference points.

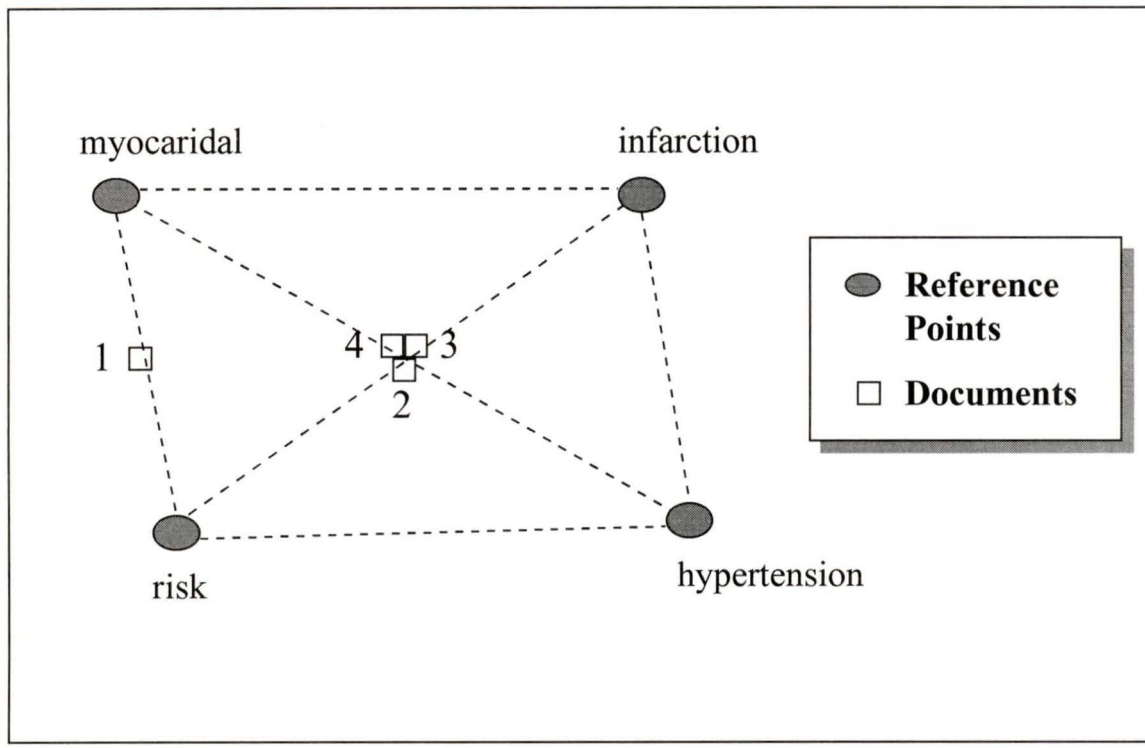


Figure 4.1: An example VIBE reference point display

The reference point presentations are often difficult to interpret, particularly with longer queries. Figure 4.1 shows documents 2, 3, and 4, midway between several reference points. It is not clear why these documents are positioned where they are. Any two or more reference points could position each of these documents as shown. Systems based on reference points have not been objectively evaluated so it is difficult to tell if reference point methods have any potential to improve retrieval effectiveness.

4.2 Spring Embedders

Spring embedding is an approach to graph drawing based on a physical system. The algorithm simulates a physical system in which a graph's edges are replaced by springs, and the nodes are replaced by rings that connect the springs and repel one another. From the initial random configuration of ring positions, the system oscillates until it stabilizes at a minimum energy configuration. Kamanda and Kawai[55] introduced a

spring embedder algorithm that places data in two dimensions. Kamar and Fowler[56] extended the algorithm to position objects in three dimensional spaces.

In an IR context, the extension of the springs corresponds to the distances among documents in a set. Distance may be measured by cosine, Euclidean distance, or some other measure. The rings are the documents themselves. Chalmers and Chitson[57] used a two dimensional spring embedder to display up to 300 documents. Swan and Allen[58] describe a three dimensional spring embedder to graphically display up to 200 ranked articles in response to a VSM based search. Figure 4.2 shows a screen-shot from Swan and Allen's system of a set of documents embedded in three dimensions. The researchers added a user definable threshold to allow a user to control the level of clustering. Documents closer than a user-definable threshold have their springs appear at full force. Documents further away than the threshold have their spring forces attenuated. These springs for these documents are not shown in the display.

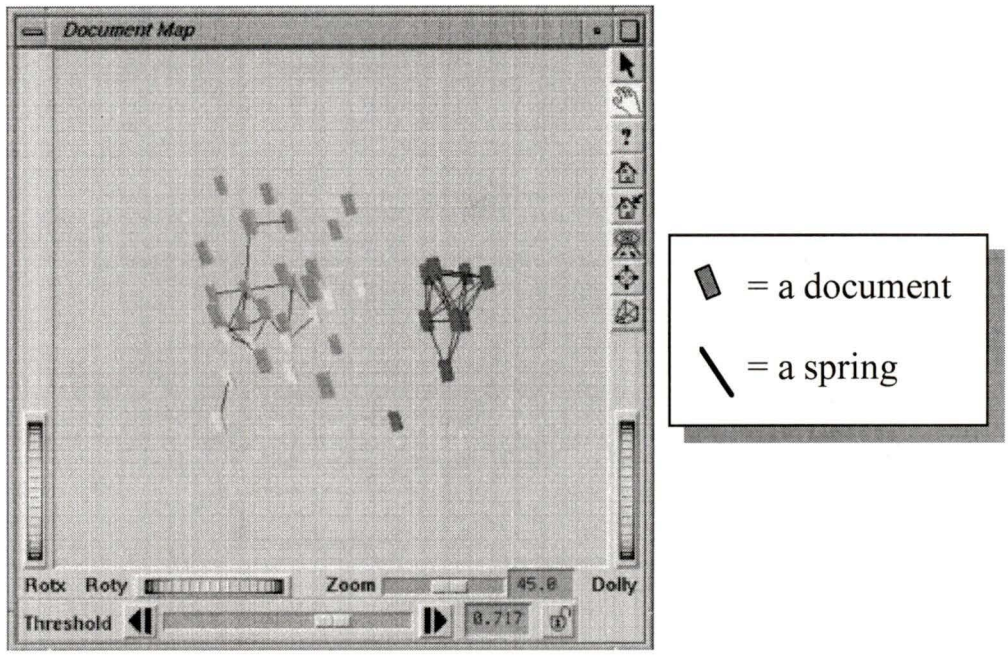


Figure 4.2: Example "Document Map" based on Spring Embedding[i]

Spring embedder algorithms can graph any set of documents with known distances among them. A drawback of the spring embedding is that solutions are indeterminant; given the same set of input data in a different order or starting position, the algorithm may produce very different looking graphs. Non-IR research in visualization that employs spring embedders indicates that users find this lack of consistency in data representation confusing.[59] Another limitation is that spring embedder algorithms have a running time of $O(n^3)$; this is a greater time complexity than even the SVD.

In an evaluation of the system in Swan and Allen's study, four users were asked to conduct two searches on a system equipped with a visualization tool based on a spring embedder and two on a system with ranked output. Although users in Swan and Allen's study said that they enjoyed the visualization aspect of the tool, there were too few

---

[i] Taken with permission from Swan RC and Allan J. Improving interactive information retrieval effectiveness with 3-D graphics. 1996. Online; Available: http://ciir.cs.umass.edu/info/psfiles/irpubs/ir-100.ps.gz

subjects to reach reliable conclusions regarding whether the tool improved retrieval precision and recall. The quality of document clustering was not examined in this study.

4.3 Clustering

The Cluster Hypothesis asserts that "closely related documents tend to be similar to the same requests."[60] That is, if documents D_1 and D_2 are relevant to query Q_1, then D_1 and D_2 will both be relevant to query Q_2. Collection wide clustering studies have generally shown that just because two documents are relevant to a particular query, they are not both relevant to an unrelated query. Therefore, evidence suggests that the Cluster Hypothesis does not hold true for an entire collection.[61] More recent work indicates that on a retrieved set of documents, rather than the entire collection, relevant documents do indeed form valid clusters. Therefore, deriving dynamic clusters based on the result set of a particular query can be a useful vehicle for presenting retrieved documents.

Cutting, Karger, Pedersen and Tukey[62], Cutting, Karger and Pedersen,[63] and more recently Hearst and Pedersen[64] used a non-hierarchical clustering algorithm to organize as many as 1,000 top ranked articles. The system, Scatter/Gather, employs an underlying VSM. The clustering algorithm uses as input document vectors from top ranked articles and the number of clusters desired. The non-hierarchical clustering algorithm groups documents into clusters based on their proximity in the term-document matrix space. Scatter/Gather presents document clusters to the user in a series of windows (see Figure 4.3). In Hearst and Pedersen's experiment on a large heterogeneous test collection, the researchers reported that over all queries in the test set, the cluster with the most relevant documents contained the majority of the relevant articles for that query; over 80 percent on average. Therefore, clustering the results of a search proved to be an effective method of separating relevant from non-relevant articles; it allows users to focus in on a particular document cluster of interest. The authors also calculated mean precision in the best clusters for each query in those clusters with the most relevant documents. The precision was compared to mean precision for an equivalent number of retrieved documents using VSM rankings. Averaged over all queries, Scatter/Gather afforded an average

improvement of 50 percent in mean precision over the baseline VSM performance for retrieved set sizes of 100, 250, 500 and 1,000 documents.

Figure 4.3 shows a sample Scatter/Gather display based on the query `auto car vehicle electric`. In this example, the top ranked 100 articles from a VSM based search are clustered into five groups.

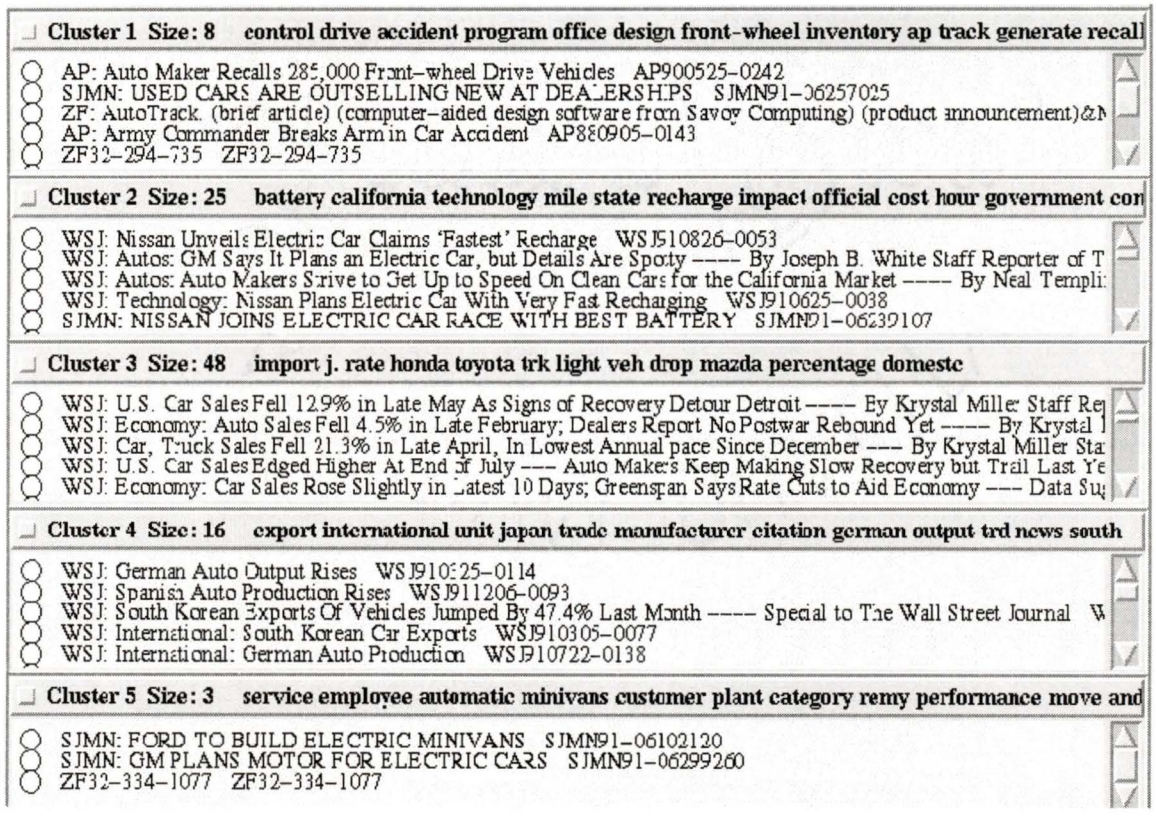


Figure 4.3: Scatter/Gather results on query: `auto car vehicle electric` with a cutoff of 100 documents[i]

Query specific clustering appears to be an effective method for organizing a large number of retrieved documents. Clustering algorithms, particularly those that are non-hierarchical, are fast and deterministic unlike spring embedders. There are limitations in

[i] Taken with permission from Hearst MA. Pedersen JO. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1996;76-84.

the presentation of their results. Some queries produce results that fail to form interpretable clusters. Also, it is difficult to convey the quality of clustering because the clusters are generated from a vector space made up of thousands of dimensions. Moreover, Scatter/Gather does not provide any information about the relationships between clusters.

4.4 A New Method: Local Latent Semantic Indexing/Cluster

In the previous chapter on search strategies, the Singular Value Decomposition was shown to be an effective procedure for reducing the dimensionality of the term-document matrix in the LSI model. Although it is possible to combine LSI with spring embedding or clustering, there is another alternative. Hull experimented with carrying out a second SVD on a set of known relevant documents to improve performance of an LSI model for the routing problem.[47] Hull used the information from what he termed the "local LSI space" of known relevant documents to predict the relevance of new documents that the system had not yet seen. This section describes an adaptation of the local LSI space for use in the visualization of ad-hoc search results.

In our research, we are assuming that the search is ad-hoc; there is no *a priori* knowledge about which documents are relevant or not. By drawing on the findings of experiments with Scatter/Gather, it seems likely that highly ranked relevant documents in the retrieved set from an LSI search will be different in some discernible manner from non-relevant documents in the same set. Performing an SVD on document vectors from the retrieved document set, and retaining two or three local constructs may help to make the differentiation. In the same manner as LSI, the truncated right singular vectors are scaled by the local truncated local singular values. For ease of terminology, the local truncated scaled right singular values matrix will be referred to as the local document space. Local constructs are the orthogonal dimensions derived from the local SVD. The local document space may be presented to the user as a two or three dimensional scatterplot. The coordinates in the local document space are clustered to increase the

contrast of the display. The process of conducting a local SVD on a set of document vectors, and then clustering the coordinates of the local document space, will be referred to as LLSI/Cluster.

An example illustrates the motivation for conducting a local LSI on the result set. Consider a collection indexed with LSI of 2,344 documents and 100 global constructs. The query `does caffeine improve exercise performance` is applied to the collection and the top 100 ranked articles are found. Conceptually speaking, the goal is to define a score for each document along an orthogonal set of "theme" constructs. Just as with the global LSI constructs, these local constructs or themes are not directly interpretable. Loosely speaking they are the most important "themes" that are derived statistically from documents in the result set. The rows of the LSI document space from these 100 articles are used to create a matrix of LSI document vectors by LSI constructs, to which another SVD is applied, and two local constructs are retained. After scaling the documents by the singular values from the local SVD, we now have a set of coordinates with which we can plot the 100 articles along two local constructs (see Figure 4.4). In this case, three local constructs could have just as easily been retained to allow the graph to be plotted in three dimensions.

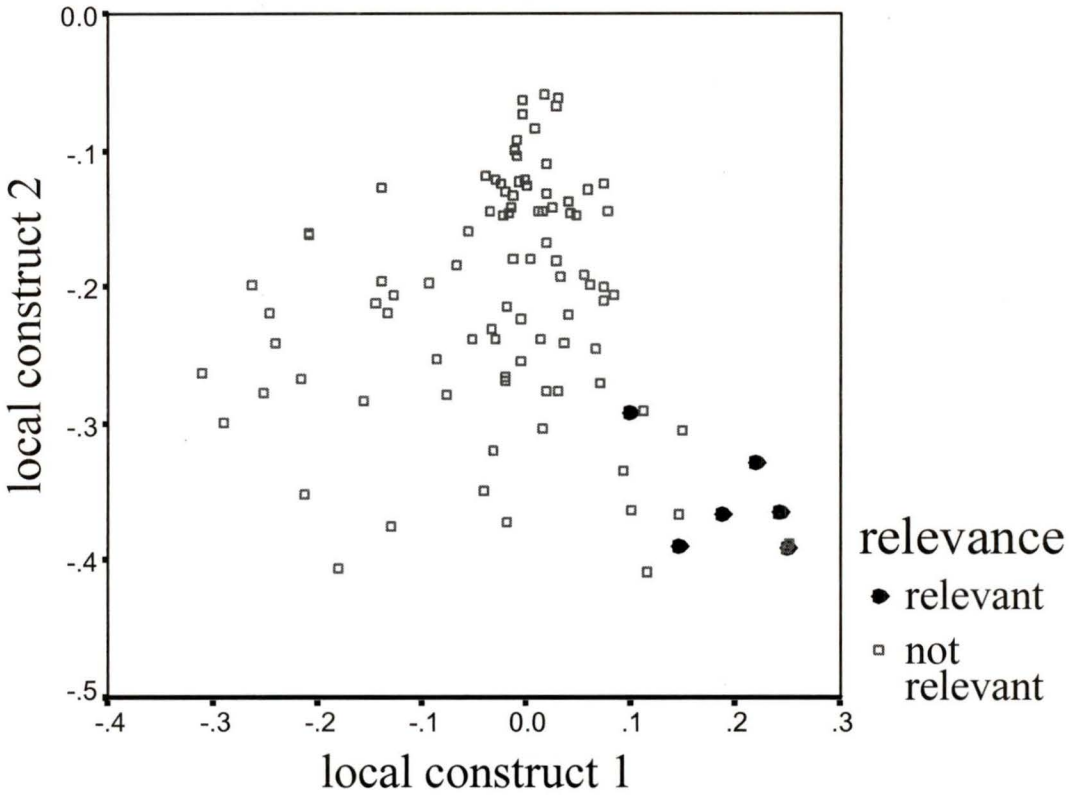


Figure 4.4: Example Local LSI plot in two dimensions

Figure 4.4 shows the locations of the top ranked 100 articles plotted with respect to their scores on the two local constructs. The squares represent non relevant articles, and the larger circles denote relevant articles for this query. No clustering has been applied for this example. As expected, the six relevant articles for the query are all located in the same region of the graph. After scanning the titles of a number of documents, one might infer that local construct one has something to do with caffeine; all articles with positive values on construct one are related to this topic. Local construct two appears to be related to exercise; documents with a stronger focus on exercise tend to have a smaller ordinate value. When plotted in three dimensions, these document objects form even more prominent clusters.

The LLSI/Cluster method has several advantages over other visualization techniques. Unlike methods that rely on reference points, LLSI/Cluster supports relevance feedback because it can function with queries of any length, even entire

documents. Unlike spring embedders, LLSI/Cluster solutions are deterministic; given the same set result set of documents, the system will always produce the same output. Unlike the clustering-only methods described, LLSI/Cluster allows one to assess the relationships between clusters as well as the dispersion and cohesiveness of each cluster.

The LSI method is an effective means of dealing with the ambiguities of language, and generating ranked output. Because of the inherent complexity of LSI and the difficulty in the interpretation of ranked results, we have presented techniques for visualizing search results. Several visualization methods were presented; each reduces dimensionality of the search space. A novel method called LLSI/Cluster has several advantages over the others. The next chapter describes a retrieval system based on the LLSI/Cluster method.

Chapter 5

5. Local Latent Semantic Indexing/Cluster System Description

This chapter describes the design and implementation of a system based on LLSI/Cluster. The goal of the chapter is to present the reader with background information, and an example illustrating how the system might be used. Background information about programming tools and protocols used is presented along with a process model and the clustering algorithm.

The LLSI/Cluster system is designed to function as a Common Gateway Interface (CGI) application.[65] CGI is the part of a World Wide Web server that can communicate with other programs running on the server. With CGI, the Web server can call up a program and pass it data from an HTML form originating on the client side. The program processes the data, passes it back to the server, and the server sends the program's response back to the client's Web browser. The CGI approach was chosen because it leverages the existing capabilities of current World Wide Web browsers and VRML display tools that work with Web browsers.

The core retrieval engine is written in C++, a high-level, compiled, object oriented language based on C. Because C++ code is compiled it executes much faster than interpreted languages such as Perl. The routines used in LLSI/Cluster are very computationally intensive, so C++ was used to implement the core search routines. C++ has been in existence since 1980; C++ development environments are stable and well supported and there are a large number of mature and robust libraries available. One such library, called Newmat,[66] was used to perform necessary SVDs for the local LSI procedures.

Unlike C++, Perl and JavaScript are interpreted languages. Perl makes up for its lack of speed by providing functionality for pattern matching and string handling. These features make it an ideal language for CGI programs. JavaScript is also an interpreted language whose primary purpose is to control objects and events within a Web browser window.[67] JavaScript makes it possible to manipulate the attributes of an object within a VRML scene without redrawing the entire scene.

Virtual Reality Modeling Language (VRML) V1.0 is an interpreted language for defining the placement and attributes of objects in three dimensional space.[68] In VRML, objects, referred to as nodes, can be defined in terms of combinations of basic colors and shapes, e.g., spheres, cones, and cubes. Nodes can also be given unique identifiers and attributes such as anchors. Anchors act as hyper-links to other web pages, or CGI programs. VRML is an ideal vehicle for plotting three dimensional data because it is simple to code and allows a high degree of customization. VRML is well suited for viewing data because VRML browsers permit more freedom of movement than the static three dimensional displays used by many commercially available statistical programs.

In order to carry out document pre-processing and the global singular value decomposition, existing third party software was used. SVDPACKC is a freely distributed C program that implements optimized algorithms for computing the SVD of large, sparse matrices.[69] SVDPACKC is used to derive the reduced space term, document and singular value matrices for use with the author's LSI retrieval engine. The SMART[70] system, a freely distributed implementation of the VSM, is used to parse collections for use with the LSI retrieval engine.

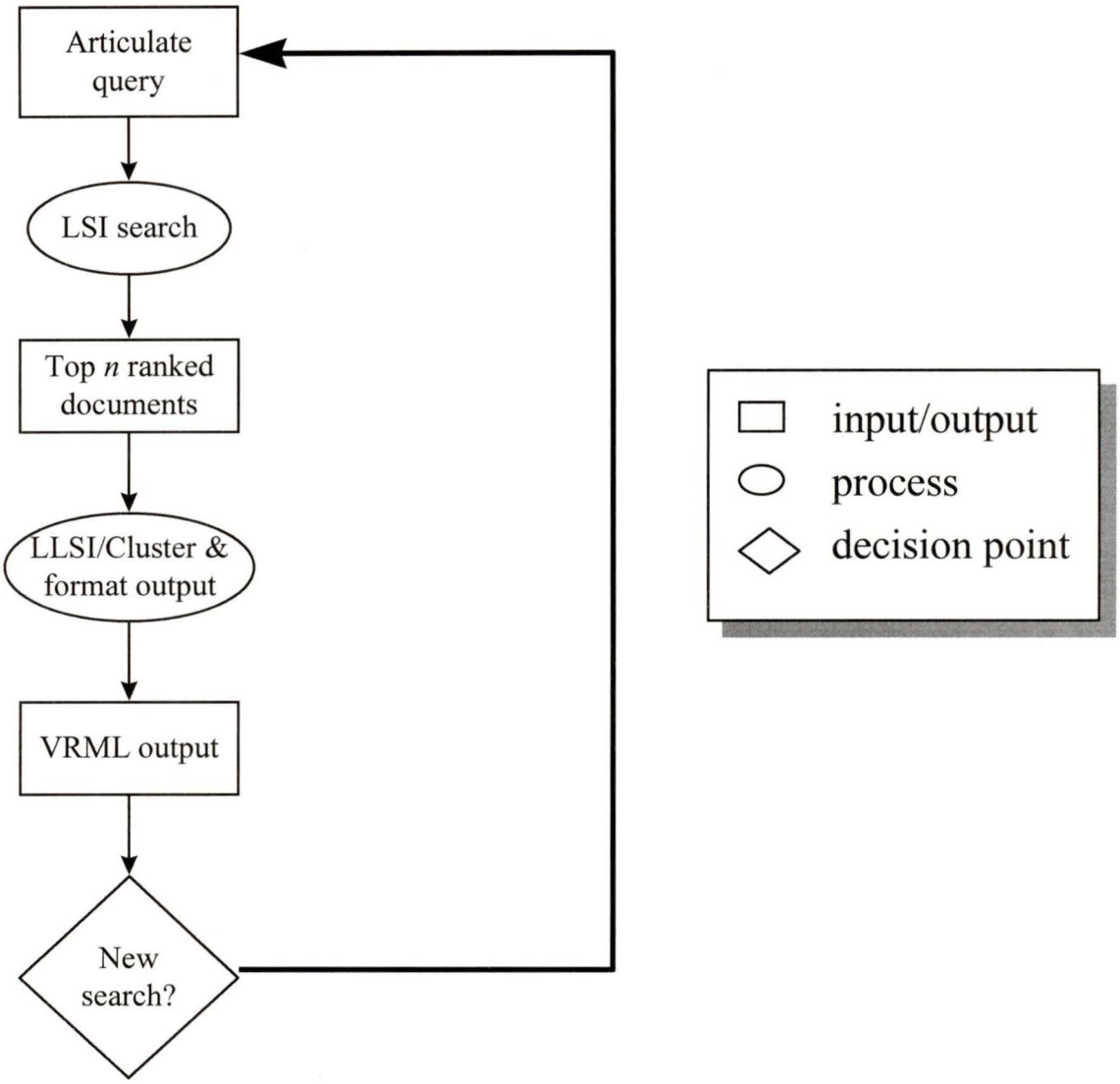


Figure 5.1: A flow diagram of the process model for interactive use of LLSI/Cluster

Figure 5.1 illustrates the information flow of the system in an interactive setting. First, the user chooses a collection to be searched, the number of clusters desired, and then composes a query. A threshold of *n* must be set to indicate the number of documents that the user would like to view initially (see Figure 5.2).

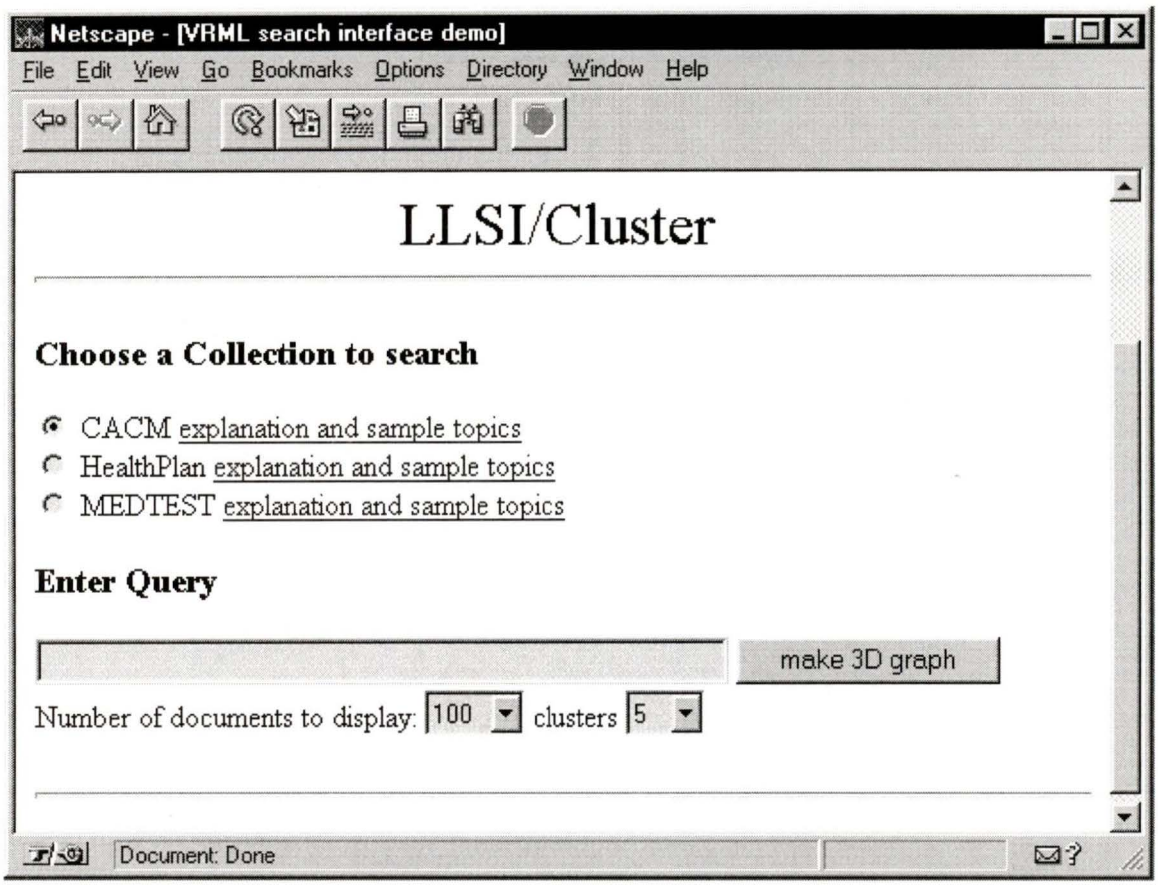


Figure 5.2: The query input screen for the LLSI/Cluster system

The query is passed to the retrieval engine. The engine converts the query into a pseudo-document and compares it with documents in the collection. The top *n* ranked documents are passed to the LLSI/Cluster module which performs an SVD on a matrix comprised of the LSI document vectors producing a three x *n* matrix of coordinates for each of the *n* top ranked documents. This data is passed to a clustering routine that groups the documents into *k* clusters. The document id, cluster membership and coordinates from the local document space is passed to a Perl script which creates VRML code to send to the browser. A separate Perl script executes whenever the user clicks on a document object; the script locates the pertinent document in a file, formats the document into HTML code, and sends the formatted document back to the Web browser.

5.1 The Document Clustering Algorithm

LLSI/Cluster employs a hierarchical clustering algorithm to organize a set of results into more visually discernible clusters. The algorithm takes a set of coordinates for each of the top *n* ranked documents and a fixed number of clusters as input. Experimentation with variable numbers of clusters and a stopping rule[71] were not productive; for this reason a fixed number of clusters were used.

There are two major types of clustering algorithms. Both have been used in previous experiments in document clustering. Agglomerative hierarchical clustering procedures result in a succession of solutions with a decreasing or increasing number of clusters. Agglomerative hierarchical algorithms begin with all cases as a unique cluster, producing an decreasing number of clusters at each iteration as existing clusters fuse together. Top-down hierarchical algorithms begin with all cases as a single cluster; they produce a succession of solutions containing an increasing number of clusters. Non-hierarchical or partitioning algorithms only produce one solution for the number of clusters requested. Non-hierarchical methods are faster, but not as flexible because they require the user to specify the number of clusters in advance. For our purposes, flexibility was considered more important than speed, so an agglomerative hierarchical method was chosen.

Agglomerative hierarchical clustering methods form clusters by successively fusing the closest individual data elements into larger groups. One method commonly used to determine which individuals or clusters to fuse together, is called the *average linkage between groups method*, also called UPGMA (unweighted pair-group method using arithmetic averages).[72] The UPGMA defines the distance between two clusters as the average of the distances between all pairs of cases in which one member of each pair is from each of the clusters. The Squared Euclidean distance is used in this research as a measure for calculating the distance between objects.

5.2 An Example Search with LLSI/Cluster

The following example illustrates an interactive search with the LLSI/Cluster method. Suppose the user is interested in the relationship between myocardial infarction and hypertension. The user types in his query and the system returns the figure in Figure 5.3 which is a three dimensional graph of the top 100 documents in local LSI space clustered into five groups.

Figure 5.3 shows the resulting graph from the query, `relationship between myocardial infarction and hypertension`. The three axes represent the three LLSI constructs or themes. Each cube and sphere represents a document. Relevant documents are shown here as circles, and non-relevant articles as squares. Relevance was determined by the creators of the test collection. Cluster membership is denoted by color in the actual system and can be seen more clearly in Figure 5.4. Documents further from the origin have a stronger association with one or more of the constructs than those near the origin. In this example, the vast majority of relevant documents are contained in a single cluster in the lower left portion of the graph in Figure 5.3.

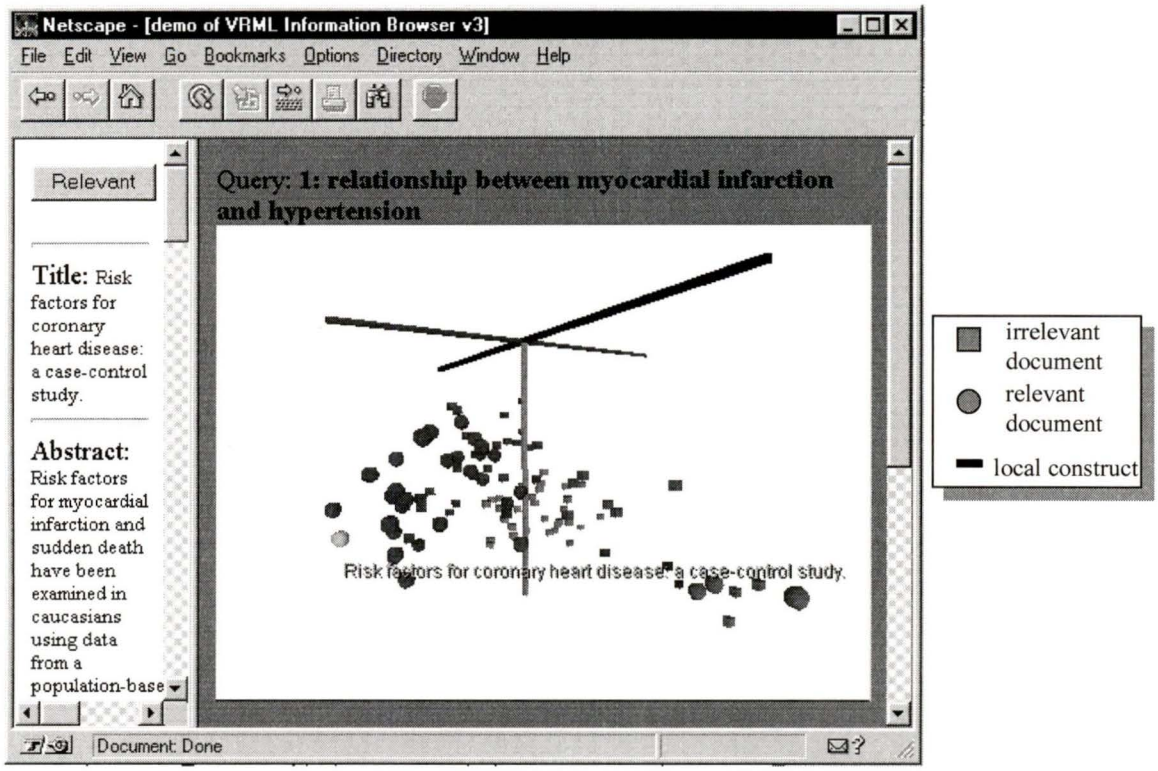


Figure 5.3: A screen shot of the LLSI/Cluster interface

The entire document scene can be rotated in any direction in order to view the graph from different vantage points. The frame on the left is the document window. When the user points the mouse arrow at a document, the document title is displayed. Selecting a document by clicking on it with the left mouse button displays the entire citation in the left hand frame. Figure 5.3 shows the title, *Risk factors for coronary heart disease: a case-control study* with its corresponding article.

Figure 5.4 shows the same output to the `relationship between myocardial infarction and hypertension` query as Figure 5.3. The graph is rotated so that construct one is pointing out of the page. Construct two is the vertical axis and construct three is the horizontal axis. In the LLSI/Cluster display, documents are color coded by cluster. For illustrative purposes only, cluster membership is denoted here by outlining the boundary of each cluster and labeling them.

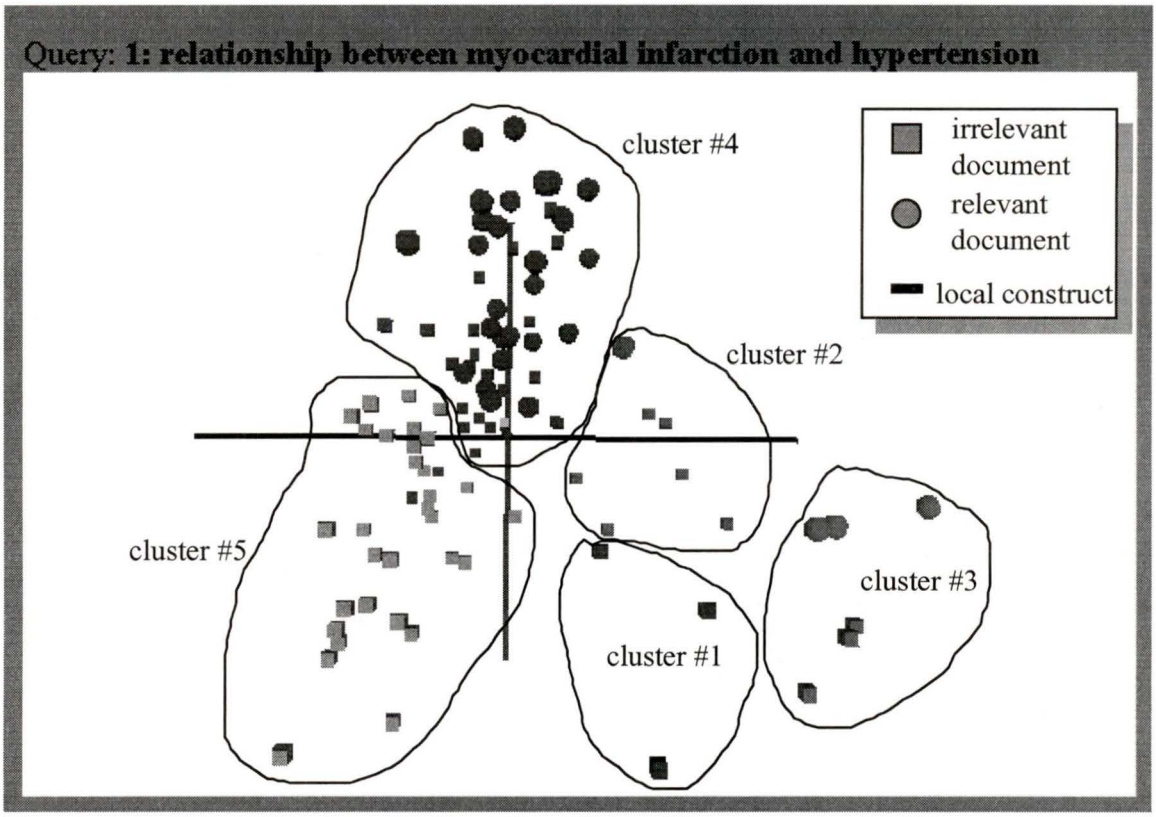


Figure 5.4: Cluster membership of documents from an example query

Cluster four is the largest with 53 documents and it contains the majority of relevant articles. A theme of many documents in this cluster is risk factors for myocardial infarction; hypertension is a commonly investigated risk factor. The second largest cluster, with 31 documents, is cluster five. This cluster includes mainly articles about beta-blockers and related drug therapies for myocardial infarction. Cluster three contains only six articles, three of which are relevant. The articles in the third cluster mainly discuss the effects of various medications for treating hypertension and myocardial infarction on blood haemodynamics.[i] Clusters one and two do not appear to have consistent themes; they each contain a variety of different articles. This lack of cohesiveness is reflected visually by the high degree of dispersion in these clusters. The articles in cluster one are quite far apart, reflecting the differences in their content. Cluster

[i] haemodynamics is the study of forces involved in the circulation of blood.

two contains one relevant article that is similar in content to the majority of relevant articles in nearby cluster four. A user would browse various clusters by rotating the scene, scanning document titles and then decide to concentrate on clusters three and four.

In this example the clusters have logical groupings even though the system has not included all relevant documents into one major cluster. Documents in cluster four center around the topic of hypertension as a risk factor for myocardial infarction. Documents in cluster three are concerned primarily with evaluating the effects of medications to treat hypertension and acute myocardial infarction on blood flow (haemodynamics). Although these two types of documents are both relevant to the query, they are distinct from one another.

This chapter illustrates the use and capabilities of the LLSI/Cluster system. It is not yet clear to what extent relevant documents will cluster together with a number of queries and a different test collections. Moreover, it is not clear to what extent LLSI/Cluster improves or degrades search performance with respect to LSI or the VSM.

Chapter 6

6. Statement of Problem

The LLSI/Cluster method offers some distinct advantages for searching document collections such as MEDLINE in comparison to the Boolean, VSM, and LSI models. The LSI engine will account for synonymous terms provided that they are used in a similar context. It produces ranked output that can be easily manipulated. LLSI/Cluster can be used to display LSI output with the following additional advantages: (1) users can easily explore similarities between documents, (2) users gain insight into the organization imposed by the system, and (3) more information can be displayed in the visual context than with ranked output.

It is unlikely that retrieval performance will significantly improve with respect to LSI. There is considerable loss of information in LLSI/Cluster when creating a very small number of local constructs from a much larger number of global ones. With LSI, documents are ranked according to their proximity to the query based on scores on approximately 100 dimensions. Local LSI representations only have three dimensions. When LSI performance is good, it will be very difficult to improve on it with a model based aggregate information. In cases where LSI performance is poor, it unlikely that there are enough global dimensions to create an accurate local representation of the query results for LSI or LLSI/Cluster.

LLSI/Cluster, as described in chapter five, can be used to visualize document semantic content, and assist in discriminating between different types of documents. We need to investigate to what extent LLSI/Cluster groups relevant documents together. This functionality is critical to the usefulness of the procedure for users. Furthermore, we need to determine whether LLSI/Cluster results in improved precision over LSI or the VSM.

Three major hypotheses are under investigation. They are:

1. Relevant documents tend to cluster together. That is, the distribution of documents in the best cluster compared to all other clusters will be significantly higher than would be expected from a random distribution.
2. LLSI/Cluster mean precision (based on the best cluster) will be comparable to the LSI model (based on a cut-off number of documents in the best cluster).
3. LLSI/Cluster mean precision (based on the best cluster) will be higher than the VSM (based on equivalent cut-offs).

These hypotheses will be investigated in a set of experiments outlined in the next chapter.

Chapter 7

7. Methods

A series of retrieval experiments were performed on three sub-collections of MEDLINE. This section describes the methodology for testing. The test data, instrumentation, procedures, evaluation measures, and statistical tests used are presented is this chapter.

7.1 Test Data

Three test collections were used in this study. The first, MEDTEST, is comprised of 2,344 articles from MEDLINE. The other two collections were created from samples of OHSUMED, a much larger collection of clinical MEDLINE articles. Table 7.1 summarizes all three test collections.

Table 7.1: Summary information for test collections

Characteristics	MEDTEST	OHSU1	OHSU2
number of documents	2344	4884	2909
number of terms (occurring in more than one document)	5180	9647	6821
number of queries	75	50	50
average number of documents relevant to a query	14.32	5.4	5.4
ratio of relevant documents to total documents (signal to noise)	42.3%	5.5%	9.3%
average number of indexed terms in a document	57	59	63
average number of documents in which an indexed term occurs	13	16	15
average number of terms per query	4.69	4.82	4.82

7.1.1 The MEDTEST Collection

The MEDTEST collection was originally created in a study evaluating MEDLINE in clinical settings and was later adapted for the evaluation of retrieval systems.[73] The collection consists of 2,344 documents from MEDLINE and 75 queries created by novice and expert physician searchers, and a list of documents judged to be relevant to each query (see Appendix C for a sample article and queries). A limitation of MEDTEST is that it is a clustered collection created from the union of Boolean MEDLINE queries. There are 991 unique documents in the collection that are relevant to one or more queries. This is an unrealistically high proportion of relevant documents (over 42% of the articles are relevant to at least one query). See Table 7.1 for summary information.

7.1.2 The OHSUMED Collections

OHSUMED is a clinically-oriented MEDLINE subset, consisting of 348,566 references covering all references from 270 medical journals over a five-year period (1987-1991).[74] OHSUMED is a representative cross section of all clinically related MEDLINE articles over this five year period.

The OHSUMED collection was built as part of a study assessing the use of MEDLINE by physicians in a clinical setting.[75] Novice physicians using MEDLINE generated 106 queries; the collection contains relevant articles for 101 of these queries. Each query was later replicated by four searchers: two physicians experienced in searching and two medical librarians. The results were assessed for relevance by a different group of physicians who used a three point scale: definitely, possibly, or not relevant. A follow-up experiment by Hersh and Buckley using SMART identified additional relevant articles not found in the initial experiment.[20] In total, the collection contains 4,886 articles rated as possibly or definitely relevant to one or more queries. Therefore, less than two percent of the documents in OHSUMED are relevant to one or more queries. Due to memory, storage, and CPU speed limitations of the test platform,

results on two smaller collections, OHSU1 and OHSU2, taken from the OHSUMED collection are reported here.

7.1.2.1 OHSU1

This collection is a sub-collection of the much larger OHSUMED set. The goal was to construct a collection that has approximately twice as many articles as MEDTEST and has more realistic characteristics than MEDTEST. The OHSU1 collection was created by starting with 4600 randomly selected "not relevant" documents (documents that were not relevant to any queries) from the 1991 subset of OHSUMED. The random selection was made using a random number generator to select documents from a pool of approximately 50,000 non-relevant candidate documents. Then 50 queries out of 101 were chosen randomly, and the definitely relevant documents for these queries were added to the 4600 already chosen. This resulted in a final set of 4,884 relevant documents. Of the 4,884 documents, 271 of the 4,884 documents are relevant to one or more queries, a signal to noise ratio of 5.5% (see Table 7.1).

7.1.2.2 OHSU2

Like OHSU1, the OHSU2 collection is also a subset of OHSUMED. The OHSUMED collection contains a set of articles that have been judged as either relevant or non-relevant. OHSU2 was constructed by pooling all of the judged, non-relevant articles from the 1991 segment and combining them with relevant articles from the same 50 queries used in the OHSU1 tests. This sampling procedure allows us to rule out the effect of individual queries as a reason for poor retrieval performance on the OHSU1 collection. The relevant articles were also from the same 1991 segment. The result is a collection with a lower signal to noise ratio than MEDTEST, but a higher signal to noise ratio than OHSU1. The resulting test set contains 2,909 articles, 271 of which are relevant to one or more queries (see Table 7.1).

7.2 Instrumentation

The test platform is a Pentium processor (100), IBM compatible machine, running both the Linux operating system and Windows 95.[i] Document parsing and summarizing routines are written in Perl.[ii] Berry's SVDPACKC program runs on Linux and is used to calculate SVD's on each collection. The SMART[70] system, which also runs on Linux, is used to parse each collection for LSI, as well as to compute comparative VSM results. The LLSI/Cluster system, described in detail in Chapter five is used to produce the LLSI/Cluster and LSI results. The author wrote other software in C++ and Perl to calculate recall and precision, and perform various other batch testing functions.

7.3 Procedures

The procedures section outlines the steps taken to prepare the test collections for experimentation with LLSI/Cluster, LSI and the VSM.

7.3.1 Latent Semantic Indexing and Local Latent Semantic Indexing/Cluster

The procedure is the same for all test collections, although the MEDTEST queries are different from those for OHSU1 and OHSU2. First, documents are pre-processed in order to construct the term by document matrix for SVD transformation. Documents are then parsed to calculate term frequencies and weights. Only the title and abstract sections are included in the indexing process. An example document is included in Appendix C. MeSH headings are not indexed in order to avoid biasing results in favor of LSI and LLSI/Cluster. Only terms that occur in more than one document are included in the term document matrix. After constructing a term by document matrix of term frequencies, the

---

[i] Linux is a freely available implementation of the UNIX Operating System. Linux runs on IBM compatibles and other personal computers.

[ii] Perl is general purpose interpreted scripting language that runs on all UNIX platforms.

elements are weighted to reflect their level of importance in the document and the collection as a whole using the weighting scheme discussed in Chapter three (see Equation 3-4). An SVD of the term-document matrix is performed and a suitable number of constructs is retained. The precise number of constructs retained varies depending on the test collection. Terms that occur in only one document are now folded back into the LSI term space. These terms were omitted during the earlier indexing stage. For retrieval, each query is also transformed into a vector of terms. These terms are weighted, and the weighted query vector is folded into the LSI document space. Then, the distance between the folded-in query and all of the documents is calculated using the dot product and documents are ranked according to their similarity to the query vector. The procedure is repeated for each query in the set, producing a set of 100 or 200 candidate documents ranked from most similar to least similar to the query.

7.3.2 Local Latent Semantic Indexing/Cluster

For each query, a different three dimensional local document space is constructed. The rows of the global document LSI space that were part of the 100 or 200 top-ranked documents made up the matrix for the LLSI/Cluster procedure. Retrieval set sizes of 100 and 200 were chosen arbitrarily. Due to computational constraints imposed by the clustering algorithm, and the local LSI procedure, sets larger than 200 would take over 10 minutes to compute per query with the test platform. The results of an SVD of the matrix of top ranked documents produces a set of 100 local constructs, of which only the first three are retained. The three dimensional local document space provides a set of coordinates for each document. These coordinates are used as input for the UPGMA clustering algorithm. For each query, the clustering algorithm partitions the documents into five groups which can be displayed in VRML scene in a Web browser. For experimental purposes, information about document locations and cluster membership was stored in a file that could later be used for aggregating results.

7.3.3 Vector Space Model Baseline

The SMART system is used for parsing, indexing, and retrieving documents for the VSM control group. Using only the title and abstract fields, documents for VSM comparison runs are parsed and weighted in exactly the same way as the LSI runs. For retrieval, each query is transformed into a vector of terms that are weighted using the TF*IDF method (see Equation 3-4). Using the dot product, the query is compared with all documents that contain at least one of the terms in the query and the most similar documents are returned. The LSI, LLSI/Cluster, and VSM are subjected to the same pre-processing routines in SMART.

7.4 Evaluation Measures

There are two questions under investigation here. The first is; to what extent can LLSI/Cluster group together relevant documents? The second is; does this clustering help improve retrieval precision? To address the first question the documents are clustered into five groups and these clusters are sorted according to the number of relevant documents that each cluster contains. The number of clusters to display was chosen arbitrarily. The goal is was to have as many clusters as possible without over burdening a user. In Hearst and Pedersen's research,[64] users reported that they did not want to examine more than five clusters. Based on their findings, we chose to restrict the results to five clusters as well. The best cluster is defined as the one that contains the highest number of relevant documents. The worst cluster contains the fewest relevant documents. For each query with at least two relevant documents, the number of relevant documents in the best cluster are added to a running total. All other relevant documents are placed into a separate bucket amalgamated from the other four clusters. A chi-square analysis can then be performed, comparing the observed distribution of relevant documents in the best cluster with what would be expected if relevant documents were distributed randomly into the best cluster or one of the other clusters.

In order to compare retrieval precision, we needed to create a results ranking method for LLSI/Cluster. Unlike the VSM, or LSI, documents in the LLSI/Cluster output have no inherent ranking. For purposes of evaluation, we rank documents in the best cluster by their distance from the origin. Documents furthest from the origin are counted first. Ranking documents based on how close they were to the cluster centroid was also tried in pilot testing, but this was found to be not nearly as effective as using distance from the graph origin. Armed with a way to rank output with each method, we can now calculate precision.

The formula for calculating precision is defined in chapter two (see Equation 2-1). In our case, we will start by calculating a measure of precision for each document in the best cluster. Mean precision for a particular query can be found by averaging all precision values calculated for each document in the best cluster. Averaging precision within a query is necessary to smooth out any irregular effects due to the particular number of documents in a given cluster. For instance, if there are 10 relevant documents for a particular query, and 25 in the entire cluster, the LLSI/Cluster method may find all 10 first, while LSI may only find the 10 after choosing irrelevant documents first. If exact precision at 25 documents is used, the two methods will have the same scores. Equation 7-1 describes the calculation for mean precision in the best cluster.

$$\text{mean precision (per query)} = \frac{\sum \text{precision}}{\text{number of documents in best cluster}}$$

Equation 7-1

Equation 7-2 describes the calculation for aggregating mean precision over all queries in a test collection.

$$\text{mean precision (aggregate)} = \frac{\sum \text{mean precision (per query)}}{\text{number of queries}}$$

Equation 7-2

The mean aggregate precision value shown in Equation 7-2 is used to compare results between LLSI/Cluster, the VSM, and LSI.

7.5 Statistical Data Analysis

The hypothesis that relevant articles will tend to cluster together is tested by conducting a chi-square analysis. The analysis is of the observed versus expected distribution of total relevant documents in the best cluster and the total relevant documents in the rest of the clusters. The hypotheses that mean aggregate precision over all queries with LLSI/Cluster is comparable to (1) LSI and (2) higher than the VSM) will be tested using the Wilcoxon signed ranks test (two tailed). This non-parametric test is used because of the expected violation of normality of variance across individual queries that is common in information retrieval experiments.[76] The Wilcoxon test considers both the magnitude of the difference scores and their directions, which makes it more powerful than the Sign test, another non-parametric statistic. All tests are carried out separately on each of the three test collections.

7.6 Limitations

Due to the nature of this research and the test collections used, there are a number of limitations that need to be noted. The test collections are fairly small and may not be representative of much larger collections. Because test collections and batch tests are used, some assumptions about relevance are made for the evaluation. These are:

1. All relevant documents are equally important.
2. The information in relevant documents is additive.
3. Relevance judgments in the test collections are accurate and universally appropriate.

Although these assumptions may not hold in all cases, they are a reasonable first-order approximation. Another limitation with the study is that it is not possible to state how

well users can locate the best cluster in interactive use of the tool. Lastly, the number of clusters in the LLSI/Cluster output is restricted to five. This number was chosen based on reported findings from experimental results with the Scatter/Gather system which found that users did not care to examine any more than five clusters.[64]

Chapter 8

8. Results

This Chapter presents the results of the three sets of experiments outlined in Chapter seven. The results of MEDTEST experiments are presented first, followed by results from the OHSU1 and OHSU2 collections.

8.1 MEDTEST Results

Figure 8.1 shows a plot of the eigenvalue scores of the largest 320 constructs resulting from LSI indexing. The first eigenvalue is not shown.

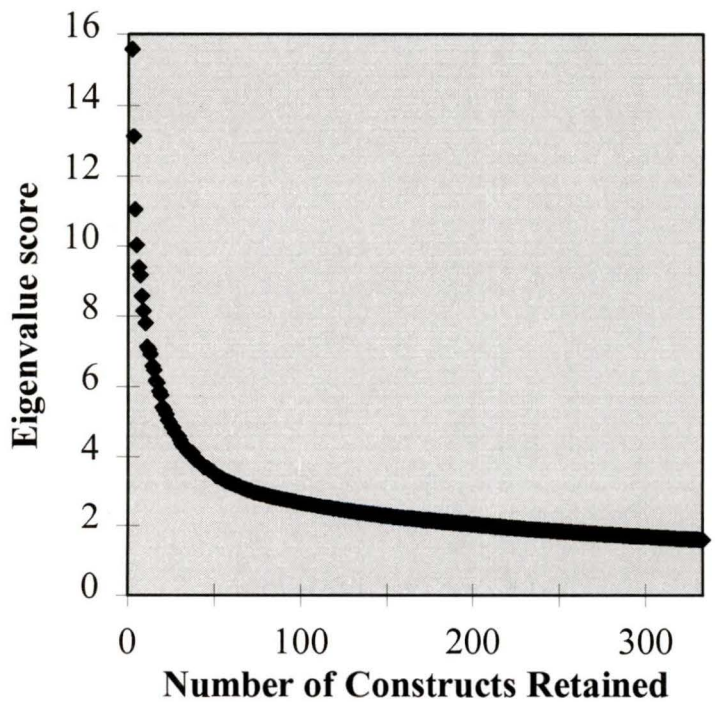


Figure 8.1: Eigenvalue score over number of constructs retained in the LSI model with MEDTEST

In many forms of research employing Principle Components plots such as the one in Figure 8.1 are used to determine the number of constructs or factors to retain. Such a technique does not help in this case, because the eigenvalues degrade very smoothly after the first few. The ideal number must be determined by information retrieval performance and practical considerations, i.e., the resulting size of the document and term spaces that can realistically be calculated, stored, and manipulated on the test platform. Figure 8.2 shows mean precision by number of constructs retained in the model. The values in Figure 8.2 were calculated by averaging precision over recall at 0.1 intervals. The figure illustrates that mean precision plateaus at 80 to 100 factors. For this reason, 100 constructs were retained for the LSI and LLSI/Cluster models with this test collection.

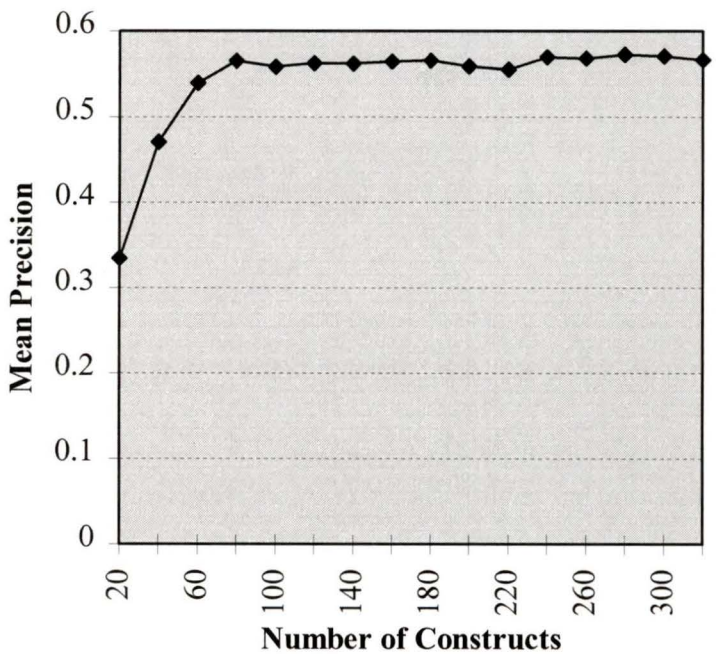


Figure 8.2: Mean precision by number of constructs retained in the LSI model

Table 8.1 shows the observed and expected total number of relevant documents in the best cluster, and in all other clusters over all queries with two or more relevant

articles. A chi-square analysis of the results in Table 8.1, indicates that the distribution of relevant documents across these two groups is far from uniform (chi-square = 550.70, df = 1, $p < 0.01$).

Table 8.1: Observed, and expected distribution of relevant documents

	best cluster	rest of clusters
observed	621	362
expected	286.61	696.39

The next question is whether or not LLSI/Cluster has the potential to improve retrieval performance. The mean aggregate precision in the best cluster (the one with the most relevant documents) is compared to mean aggregate precision of the VSM and LSI. The ranked list of documents for the VSM and LSI models is truncated at the population of the best cluster. Table 8.2 depicts the mean precision of each method over all 75 queries from the MEDTEST collection using 100 global constructs for the global LSI and LLSI/Cluster models and retrieval sets of 100 and 200. The small changes in VSM precision for different numbers of documents retrieved are due to slight differences in the cut-off value. The cut-off point is determined separately for each query based on the number of documents in the best cluster of LLSI/Cluster results.

Table 8.2: Mean aggregate precision for VSM, LSI, and LLSI/Cluster on the MEDTEST collection

n retrieved	VSM	LSI	LLSI/Cluster
100	0.407	0.461	0.451
200	0.388	0.435	0.435

The LLSI/Cluster method is comparable to the VSM and LSI with both retrieved set sizes. There are no significant differences between LLSI/Cluster and the VSM or LLSI/Cluster and LSI.

The results from the MEDTEST collection are encouraging but there are some drawbacks. Over 42 percent of the collection is made up by target documents. This means

that there is an unrealistically high proportion of documents relevant to one or more of the queries. This high proportion of relevant documents is not representative of a real MEDLINE collections. Experiments should be conducted with collections that have a more realistic distribution of relevant and non-relevant articles.

8.2 OHSU1 Results

With OHSU1, an analysis of average precision averaged over 10 recall points showed that performance continued to improve up to 319 LSI constructs. This number is the maximum number of dimensions that could be calculated, and manipulated on the test machine. Therefore, 319 constructs were retained in the LSI and LLSI/Cluster models. Averaged over all 50 queries, with LSI more than 89% of the relevant articles are ranked higher than 100, and 94% are ranked higher than 200.

Table 8.3 shows the observed and expected total number of relevant documents in the best cluster and the rest of the clusters for the 31 queries where two or more relevant documents were found in the top 100. A chi-square analysis of the results in Table 8.3, indicates that the distribution of relevant documents across clusters is non-uniform (chi-square = 452.91, df = 1, $p < 0.01$).

Table 8.3: Observed and expected distribution of relevant documents for OHSU1

	best cluster	rest of clusters
observed	155	80
expected	56.05	178.95

Table 8.4 displays mean aggregate precision for the OHSU1 collection. Once again, mean precision was calculated at a cut-off point equal to the number of documents in the best cluster from LLSI/Cluster.

Table 8.4: Average precision for VSM, LSI, and LLSI/Cluster on the OHSU1 collection

n retrieved	VSM	LSI	LLSI/Cluster
100	0.316	0.294	0.266
200	0.256	0.254	0.221

With both retrieval set sizes, LLSI/Cluster and LSI are comparable, with no significant differences between the methods. The VSM is significantly better than LLSI/Cluster on the 100 documents retrieved set; $Z = -2.136$ $p < .05$, two tailed; 200 retrieved set, $Z = -2.574$, $p < 0.05$, two tailed).

The poor results for this test set can be explained in part by the artificial way in which the OHSU1 collection was created. The collection was constructed by adding random documents from a much larger pool to a small set of relevant documents. The consequence of this procedure is that most of the articles have very little in common with one another. This characteristic differs from real collections that are constructed from a set of topics of ongoing interest. To confirm that the sampling procedure, and not the set of queries, caused the poor results with LSI and LLSI/Cluster, another experiment on another sub-collection of OHSUMED has been compiled using the same queries but a different set of non-relevant documents. The results of this next experiment will help to determine whether the poor results with LSI and LLSI/Cluster are due to the sampling method used.

8.3 OHSU2 Results

An analysis of mean precision similar to that reported in Figure 8.2 showed that performance continued to improve up to 200 dimensions, the maximum calculated. Consequently, 200 dimensions are retained in the LSI global and LLSI/Cluster models for the OHSU2 test collection.

On average, over 94% of the relevant documents are ranked at 100 or better and over 96% of relevant documents are ranked higher than 200. Table 8.6 shows

the observed and expected number of relevant documents in the best cluster and the rest of the clusters for the 35 queries in the OHSU2 collection with two or more relevant documents found in the top 100. A chi-square analysis of the results inTable 8.5, reveals that the distribution of relevant documents across clusters is non-uniform (chi-square = 101.02, df = 1, $p < 0.01$).

Table 8.5: Observed and expected distribution of relevant documents for OHSU2

	best cluster	rest of clusters
observed	167	79
expected	90.91	155.09

Mean aggregate precision at 100 and 200 documents retrieved is presented in Table 8.6.

Table 8.6: Mean precision for the VSM, LSI, and LLSI/Cluster on the OHSU2 collection

n retrieved	VSM	LSI	LLSI/Cluster
100	0.229	0.198	0.268
200	0.196	0.195	0.233

Over both retrieval set sizes, mean precision with LLSI/Cluster is higher than VSM or LSI. LLSI/Cluster (mean, 0.268, SD = 0.226) is significantly better than global LSI (mean = 0.198, SD = 0.232) when 100 documents are retrieved ($Z = -1.969$, $p < 0.05$, two tailed). LLSI/Cluster mean precision is higher than the VSM with both retrieval sizes, but not significantly so. Once again, performance drops off slightly for all methods when a retrieval set of 200 is used compared to mean precision with a retrieved set of 100.

Table 8.7 summarizes mean aggregate precision results for all three collections at result sets of 100 and 200.

Table 8.7: Summary table of mean aggregate precision

Collection	*n* retrieved	VSM	LSI	LLSI/Cluster
MEDTEST	100	0.407	0.461	0.451
	200	0.388	0.435	0.435
OHSU1	100	0.316	0.294	0.266
	200	0.256	0.254	0.221
OHSU2	100	0.229	0.198	0.268
	200	0.196	0.195	0.233

Chapter 8

9. Discussion

The results described in Chapter eight were conducted on three different collections with dissimilar characteristics. MEDTEST is a small collection constructed from the results of other experiments with MEDLARS, the Boolean search tool developed by the National Library of Medicine. MEDTEST contains a disproportionately large number of relevant documents per query. Many of the non-relevant documents are closely related to one or more of the queries. The OHSU1 collection contains a random subset of documents from a clinical base of MEDLINE articles seeded with a small percentage of relevant documents. OHSU2 is similar to MEDTEST in that it is essentially a compilation of retrieved documents, except that are fewer relevant articles per query in the OHSU2 collection. OHSU1 and OHSU2 have more realistic characteristics than MEDTEST.

9.1 Discussion of Aggregate Results

Experiments with all three test collections confirmed the hypothesis that relevant documents tend to cluster together in a three dimension local LSI space. On MEDTEST, OHSU1 and OHSU2 the distribution of relevant documents among the best cluster and the rest of the clusters was strongly non-random ($p < 0.01$, in all cases). This indicates that the method consistently grouped together relevant documents. This finding corroborates the results of Hearst and Pedersen[64] and Swan and Allen[58] who also reported that retrieved documents clustered. These results also lend support to the modified cluster hypothesis proposed by Hearst and Pedersen[64] that states that relevant documents from a retrieved set tend to cluster together.

In all cases, mean precision was higher for retrieved set sizes of 100 than sets of 200 with the LLSI/Cluster, LSI and VSM models. This is not surprising because the average cluster size is smaller with a retrieved set of 100 documents. With all three collections and retrieval set sizes, mean aggregate precision with LLSI/Cluster was comparable to LSI. With the OHSU2 collection, LLSI/Cluster mean precision was significantly higher than that of LSI. There is a straightforward explanation for this finding. In the MEDTEST collection, performance with LSI was already quite high, making further improvements difficult. With the OHSU2 collection, LSI performance is comparatively much lower. Another reason is that in 14 OHSU2 queries there is only one relevant document. When these queries are left out of the analysis, mean aggregate precision with LSI is not significantly lower than that of LLSI/Cluster (see section 9.2.3)

The result of comparisons between LLSI/Cluster and VSM mean aggregate precision failed to support the hypothesis that LLSI/Cluster precision would be higher than that achieved with the VSM. LLSI/Cluster mean aggregate precision was higher than the VSM on the MEDTEST and OHSU2 collections; but in each case the differences between the two methods was not statistically significant. Contrary to expectations, LLSI/Cluster mean aggregate precision was significantly worse than the VSM on the largest test collection, OHSU1. This result is surprising given the encouraging results reported by Dumais[49,50] with LSI on the much larger TREC collections. A closer look at experiments carried out by Dumais shows that the percentage of terms indexed in the VSM control group was much lower than normal. In those studies the experimenter only included terms for indexing that occurred in greater than four different documents. Our research includes all terms in the VSM model; this provides a less biased comparison between LLSI/Cluster and the VSM.

9.2 Discussion of Queries

A discussion of query results from the experiments with each collection further illustrates the conditions in which LSI/Cluster performed better and worse than LSI or the

VSM. Query results from MEDTEST, OHSU1, and OHSU2 are discussed in turn in this section.

9.2.1 MEDTEST

With the MEDTEST collection, the LSI model's ability to match on constructs results in improved performance when compared to the word-based VSM. For example, for one query concerning `temporal lobe epilepsy and psychosis`, the VSM ranks the relevant documents "Emotions and personality in complex partial seizures" and "Covert complex partial seizures in psychotherapy" 597th and 1075th respectively because they have no terms in common with the query. LSI ranks the same documents fifth, and 34th correctly inferring the theme of the articles. LLSI/Cluster presents these results well; all relevant documents are tightly grouped in the same cluster.

A similar example relates to the `query pharmacodynamics, uses and side effects of cyclosporine`. One relevant document describes the toxicity of cyclosporine used in allotransplantation in a clinical trial but does not specifically use the terms, "side effects." The VSM ranks this document 90th; LSI ranks it 9th. Again, the results of this query are well presented with LLSI/Cluster; of the 20 relevant documents, 19 are in a single cluster containing 18 non-relevant articles.

On queries where LLSI/Cluster performs worse than the VSM or LSI, the local LSI does a poor job of separating relevant from irrelevant articles. For example, for the query `Barterr's disease` the eight relevant documents are dispersed between three different clusters, far from the edge of the graph. Mean precision in the best cluster which constains four relevant articles and 11 irrelevant articles is only 13 percent.

9.2.2 OHSU1

Overall, the VSM scored better than LLSI/Cluster on 30 of the 50 queries. These findings appear to be due to the advantage of indexing on specific terms with this collection. On query 33 `lupus nephritis, diagnosis and management` LLSI/Cluster only clusters four relevant articles into a single cluster of 18 documents.

These four articles are scattered throughout the cluster, the other two relevant documents are in separate clusters. On this query, the VSM ranks all six relevant documents in the top seven, picking up on the rare terms, lupus and nephritis.

On another query, `fibromyalgia/fibrositis, diagnosis and treatment`, only five of the seven relevant documents are ranked higher than 100 by LSI and LLSI/Cluster. The five relevant documents are in two clusters, the first contains 51 documents and the three relevant articles in it are near the center. The VSM ranks all seven relevant documents in the top seven. Again, the VSM's ability to match on specific terms is an advantage here.

A third query, `isolated systolic hypertension, shep study` poses little difficulty for the VSM because four of the articles contain the only occurrence of the term `shep` in the entire collection. This key word is far too rare a term to be adequately modeled by LSI or LLSI/Cluster. LSI seems to get distracted by the many articles that discuss hypertension. With the OHSU1 collection, the VSM's ability to model individual terms outweighs the theoretical benefits of matching on a reduced set of constructs.

9.2.3 OHSU2

LLSI/Cluster performs much better on the OHSU2 test set than with the OHSU1 collection. LLSI/Cluster mean precision is higher on 32 of the queries than LSI, and higher on 28 than the VSM. The fact that LLSI/Cluster performed significantly better than LSI is encouraging but somewhat surprising.

There are a number of reasons for these results. First, LSI does not do well on queries where there is only one relevant document. LLSI/Cluster has an advantage in these cases because the single relevant document is guaranteed to be in the best cluster. When these queries are removed from the analysis, the means are more consistent with the MEDTEST results: LLSI/Cluster precision = 0.29, LSI precision = 0.26, and VSM precision = 0.25. Second, the majority of the documents are related to one query or another to some degree because they came from a larger retrieved set; OHSU2 was

created by combining sets of top ranked documents from a larger test collection. This characteristic undoubtedly causes LLSI/Cluster to group documents in a more meaningful way for OHSU2 than for OHSU1. The clustered nature of the collection also gives LLSI/Cluster an advantage over the VSM. Unlike LSI and LLSI/Cluster, the VSM does not model the co-occurrences of terms between documents in the collection.

The disparity in mean aggregate precision with LLSI/Cluster on OHSU1 and OHSU2 appears to be due to the sampling procedure rather than the set of queries. OHSU1 and OHSU2 both include the same set of 50 queries. Unlike OHSU1, which contains a random collection of non-relevant documents from the 1991 year of OHSUMED, OHSU2 contains only documents from the 1991 OHSUMED set that were retrieved during Hersh and Buckley's[20] experiments with the entire OHSUMED collection. Therefore, a large percentage of the non-relevant documents in OHSU2 share some terminology with relevant documents. In turn, there is more overlap in term usage with OHSU2 than OHSU1. In OHSU2 the average percentage of documents in which an indexed term occurs in is 0.52. In OHSU1 this figure is only 0.32 percent, less than two-thirds the level of redundancy in OHSU2. This greater redundancy in term usage in OHSU2 appears to be primarily responsible for the improved performance with LLSI/Cluster over the OHSU1 results.

9.3 Improving Latent Semantic Indexing Performance

Certainly, LSI and LLSI/Cluster performance could have been improved simply by including more constructs in the model for both the OHSU1 and OHSU2 collections. However, this strategy poses practical problems. Storing the term space and document space matrices from OHSU1 requires almost 36 megabytes of disk space, while the entire text of the collection only requires eight megabytes. The rate of data storage requirements for LSI grows at the rate of k^2, where k is the rank of the original term-document matrix. One year of clinical MEDLINE abstracts -- a collection of 50,000 documents or more -- could require 500 or more constructs to adequately model it. The storage requirements for such a collection is approximately 500 Megabytes (see Table 9.1).

the dispersion of clusters and the relationships between them. The experiments conducted in this thesis suggest that local LSI representation may not degrade retrieval performance. One caveat to this statement is that enough constructs must be retained in the LSI model. The experiments conducted in this thesis show that fewer constructs are needed when there is a higher overlap of term usage throughout documents in the collection.

Due to the practical limitations imposed by LSI, LLSI/Cluster is difficult to implement on realistically sized document sets extracted from MEDLINE. Table 9.1 shows the LSI indexing time and storage space requirements for MEDTEST, OHSU1 and the entire 50,217 document segment of OHSUMED from the 1991 year. For OHSUMED91 indexing time and storage requirements are extrapolated from the results obtained with MEDTEST and OHSU1. The OHSUMED91 estimates are based on a model with 300 constructs.

Table 9.1: Indexing and storage requirements for LSI

	MEDTEST	**OHSU1**	**OHSUMED91**
terms (occuring in at least 2 documents)	5180	9647	14403
total documents	2344	4884	50217
Indexing time[i]	420 seconds	8920 seconds	23260 seconds
Storage required	6.4 Megabytes	36 Megabytes	867 Megabytes

In spite of the high indexing time and prodigious storage currently required, LLSI/Cluster may become suitable for larger collections within a short time. Five years ago, the experiments carried out in this thesis would not have been possible at all using a micro computer. Moore's law[ii] indicates that it may be practical in the near future to compute and store large LSI models.[78]

---

[i] Based on the running time of SVDPACKC on a single CPU Pentium 100 with 48 Megabytes of RAM.

[ii] Moore, a semiconductor researcher, predicted in 1965 that the density of components on an integrated circuit would continue to double every 18 months. So far, his prediction has borne out.

The success of the LLSI/Cluster method suggests that such a visualization tool may be useful for users who know only vaguely what they are looking for. Belkin, Oddy and Brooks,[79] Marchionini[8] and others argue that this situation is the norm rather than the exception. Existing search tools that produce ranked lists work best when the query is well articulated. LLSI/Cluster, on the other hand, allows the user to identify a broad topic of interest and then focus in on a more specific area. Such functionality may be very useful for many MEDLINE users.

The results presented here also point out some advantages and limitations of LSI. On the MEDTEST collection, the ability to match on constructs rather than terms clearly enhanced LSI performance for a number of queries. Earlier studies with LSI also produced good results with collections similar in size and makeup to MEDTEST.[40,80] However, these small collections which LSI excels on have very different characteristics from real collections; they have greater inherent redundancy. In these collections[i] the percentage of documents in which a term occurs is over 0.8 percent of the total number of documents. For MEDTEST and OHSU2 this figure is over approximately 0.5 percent in both cases. In the OHSU1 collection, a given term occurs in only 0.3 percent of all articles in the collection. One would expect LSI to be more effective in collections where there is greater redundancy; this is in fact what occurs. Standard small test collections created by combining results from many queries, clearly have more redundancy than a real collection would. For these, LSI tends to show better performance than it does for collections made up largely of randomly selected documents.

9.6 Recommendations for Further Research

Two recommendations for further research are posed here. The first concerns the current lack of systematic examination of the problem of how many constructs to retain in an LSI model. The second recommendation proposes an alternative to LLSI/Cluster based on the VSM.

---

[i] MED, CRAN and CISI are three standard collections available at ftp://ftp.cs.cornell.edu/pub/smart/

Choosing the number of dimensions to keep in an LSI model is a problem ripe for future research. The strategies employed in other disciplines for choosing a stopping point with Principal Components Analysis are of little help with LSI (see section 3.4.4). The eigenvalues from an LSI decomposition decay nearly linearly after the first few. This makes it impossible to estimate the ideal number of constructs to retain in a model by using discrimination methods based on changes in the magnitude of the eigenvalues. In addition, the specificity or vagueness of queries also affects the ideal number of constructs. In a real-world search environment these parameters are not fully known in advance. A systematic examination of these parameters would be helpful.

Given the complexity of computing, storing, and maintaining an LSI model, one might ask if the advantages afforded by LSI are worth the effort. Although LSI is an elegant method for dealing with synonymy, polysemy and term dependence, the results from the OHSU1 collection illustrate that there is a critical loss of information in an LSI model that retains too few constructs. In such a case, retrieval performance is reduced rather than enhanced. If not enough constructs are retained, searches containing relatively rare terms can fail.

To address the problem of missing information, the full-rank term-document space could be retained for as long as possible. For instance, a VSM could be employed instead of LSI to rank order documents at a global level. A local region could then be derived from an SVD on the top *n* ranked VSM document vectors. In this case, terms which occur in top ranked documents form the rows of an input matrix and the documents themselves form the columns. An SVD of this matrix retaining the three largest constructs results in a set of coordinates for each document in three dimensional local space. The difference between this process and LLSI/Cluster is that the local space based on the VSM is derived from a subset of the original term-document matrix, whereas LLSI/Cluster representations are derived from a document-construct matrix.

There are a number of advantages to using a VSM model instead of LSI. First, the VSM is simpler to implement and maintain. Second, unlike LSI, no information in the term-document matrix is lost with the VSM. This should ensure that precision with a

visualization method is no worse than that of ranked VSM output. Finally, substituting a VSM method for LSI would make it possible to index much larger collections than with LSI.

Chapter 10

10. Conclusions

One symptom of the information age is that physicians and other health professionals are not keeping abreast of advancements in knowledge reported in the literature. A resulting obstacle to optimal health care is that attempts to locate pertinent literature are often hindered by the vast number of documents, and a lack of effective methods to locate relevant information.

Two major goals are outlined in this thesis. The first is to identify and evaluate alternative methods of indexing and ad-hoc searching of MEDLINE databases. the second goal is to devise and evaluate a method for presenting search results in an alternative fashion to a ranked list. Such a method should free the user from the constraints of list-based output.

This thesis addresses two research problems in the course of experiments comparing a new method for IR visualization and two existing list-based methods. The first question is whether the new method, LLSI/Cluster consistently clusters relevant articles together in three dimensional space. The second question is to what degree does LLSI/Cluster improve the accuracy with which relevant articles can be found compared to LSI and the VSM.

The Boolean model, VSM, Linguistic and knowledge-based methods, and LSI are presented in chapter three as methods for indexing and searching document collections. Each method has a number of advantages and drawbacks. The Boolean model is fast and easy to implement, but complicated to use, does not rank output, and does not allow one to incorporate information from documents directly into a new search. The VSM ranks output, but does not account for synonymy, polysemy or term dependence. Linguistic and knowledge-based methods directly address these characteristics of language, but systems

based on these principles have not scaled up well. Like the VSM, LSI ranks output, and it intrinsically accounts for the ambiguities in language. A major drawbacks of all of the above methods is that they produce output in a way that is difficult to manage.

Visualization methods for IR address some of the restrictions of list-based output. Visualization tools allow a user to explore similarities between documents, gain insight into the organization imposed by the system, and view more information than with ranked output. A novel method called LLSI/Cluster was shown to have some advantages over reference points, spring embedders, and clustering. LLSI/Cluster can accommodate long queries, and relevance feedback, unlike reference points. Unlike systems based on spring embedding, LLSI/Cluster produces unique solutions for a given set of input. LLSI/Cluster has an advantage over clustering methods in that it visually conveys the dispersion within clusters and the spatial relationships between clusters.

Chapter five describes a system developed by the author based on the LLSI/Cluster methodology described in section 4.4. The system was developed in C++, Perl, and JavaScript and made use of a matrix library, the SMART retrieval system, and SVDPACKC. The LLSI/Cluster system runs as a CGI application on the Windows95 and NT platforms.

The evaluation of the LLSI/Cluster method was based on the following three questions:

1. Does LLSI/Cluster consistently group similar documents together?
2. Is precision with LLSI/Cluster comparable to LSI?
3. Is precision with LLSI/Cluster better than the VSM?

In order to address these questions, experiments were conducted using three different test collections made up of MEDLINE documents.

LLSI/Cluster consistently grouped relevant documents in a single cluster on each of the three test collections. On two of the test collections, LLSI/Cluster afforded no loss

in retrieval precision over LSI and the VSM. However, on the largest of the collections, mean aggregate precision was significantly worse than the VSM.

Several implications arise from these results. The finding that relevant documents were distributed non-randomly, supports the modified cluster hypothesis proposed by Hearst and Pedersen.

There are three practical implications arising from the experiments in this thesis. First, LLSI/Cluster may be an especially useful method for broad queries; those that contain few terms, or contain mostly common terms. Second, LLSI/Cluster is only effective if enough constructs are retained in the model. At this point in time, LLSI/Cluster is restricted to relatively small collections due to the limitations of current hardware.[i] Unlike most research with LSI, this thesis attempted to apply LSI to a non-clustered collection. Our results illustrate that LSI performs better on test collections than on collections with a more realistic level of redundancy.

We propose two avenues for further research. First, the LSI community would benefit from a systematic investigation of the parameters that affect the ideal number of constructs to retain in an LSI model. Another possible line of investigation for further research is based on an alternative to LLSI/Cluster. The alternative approach we suggest is based on the VSM rather than LSI, and should therefore scale up to larger segments of MEDLINE.

---

[i] The actual size depends on available random access memory and secondary storage capacity.

11. References

[1] Price DJ. *Little science, big science*. New York: Columbia University Press, 1963.

[2] Library of Congress. *Library Services*. Unpublished manuscript, 1997. Available: http://lcweb.loc.gov/loc/libserv/

[3] Williamson JW, German PS, Weiss R, Skinner EA, Bowes F. Health sciences information management and continuing education of physicians. *Annals of Internal Medicine*, 1989;110:151-160.

[4] Borgman CL. Why are online catalogs hard to use? Lessons learned from information retrieval studies. *Journal of the American Society for Information Science*, 1986;37:387-400.

[5] Covell DG, Uman GC, Manning PR. (1985). Information needs in office practice: are they being met? *Annals of Internal Medicine*, 1985;103:327-380.

[6] Hersh WR. *Information retrieval: A health care perspective*. Springer-Verlag, New York, 1996.

[7] Furnas GW. Landauer TK, Gomez LM, Dumais ST. The vocabulary problem in human-system communication. *Communications of the ACM*, 1987;30:964-971.

[8] Marchionini G. Interfaces for end-user information seeking. *Journal of the Society for Information Sciences*, 1992;43:156-163.

[9] Medical Library Service. *MEDLINE - Some details*. Unpublished manuscript, 1996. Available: 1383 West 8th Avenue, Vancouver BC, V6H 4C4.

[10] Purcell G. & Shortliffe, EH. Contextual models of clinical publications for enhancing retrieval from full-text databases. *Journal of the American Medical Informatics Association*, 1995:851-857.

[11] Funk ME, Reid CA. Indexing consistency in MEDLINE. *Bulletin of the Medical Library Association,* 1983; 71(2):176-83. In Purcell G. & Shortliffe, EH. Contextual models of clinical publications for enhancing retrieval from full-text databases. *Journal of the American Medical Informatics Association*, 1995:851-857.

[12] Hersh W, Hickam D. A comparision of retrieval effectiveness for three methods of indexing medical literature. *American Journal for the Medical Sciences*, 1992;303:292-300.

[13] Sewell W. & Teitelbaum S. Observations of end-user on-line searching behavior over eleven years. *Journal of the American Society for Information Science*, 1986;37: 234-245.

[14] Haynes BH. McKibbon KA. Online access to MEDLINE in clinical settings: A study of use and usefulness. *Annals of Internal Medicine*, 1990;112(1):78-84.

[15] Hersh WR. Hickham DH. The use of a multi-application computer workstation in a clinical setting. Bulletin of the Medical Library Association, 1994;82:382-389.

[16] Haynes R, McKibbon K, Walker C, Ryan N, Fitzgerald D, Ramsden M. Online access to MEDLINE in clinical settings. *Annals of Internal Medicine*, 1990;112:78-84.

[17] Fagan JL. *Experiments in automatic phrase indexing for document retrieval: a comparison of syntactic and non-syntactic methods*. Ph.D. dissertation, Cornell University, Sept 1987. In Lewis DD. & Sparck-Jones K. Natural language processing for information retrieval. *Communications of the ACM*, 1996;39(1):92-101.

[18] Hull DA. Stemming algorithms: A case study for detailed evaluation. *Journal of the American Society for Information Science*, 1996;47(1):70-84.

[19] Frakes WB. Stemming algorithms. In Frakes WB & Baeza-Yates R. Eds. *Information retrieval: Data structures and algorithms*. Englewood Cliffs, New Jersey: Prentice-Hall, 1992.

[20] Hersh W, Buckley C, Leone TJ, Hickman D. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994: 192-201.

[21] Harman D. (Ed). *The First Text REtrieval Conference (TREC-1)*. National Institute of Standards and Technology Special Publication 500-207, Gaithersburg, Md., 1993.

[22] Hull DA. *Information retrieval using statistical classification*. PhD dissertation, Department of Statistics, Stanford University, 1994.

[23] Luhn H. A new method of recording and searching information. *American Documentation*, 1953;4(1):14-16.

[24] Salton G. The SMART project-status report and plans. In *The SMART retrieval system: Experiments in automatic document processing*. Ed. Salton G. Englewood Cliffs, New Jersey: Prentice-Hall, 1971:3-10.

[25] Zipf GK. *Human behavior and the principle of least effort*. Addison Wesley, Reading Massachusetts, 1949. In Salton G, McGill MJ. *Introduction to modern information retrieval*. New York, McGraw-Hill, 1983.

[26] Salton G, McGill MJ. *Introduction to modern information retrieval*. New York: McGraw-Hill. 1983.

[27] Lewis DD, Spark Jones, K. Natural language processing for information retrieval. *Communications of the Association for Computing Machinery*, 1996;39(1): 92-101.

[28] Evans DA, Ginther-Webster K, Hart M, Lefferts RG, Monarch IA. Automatic indexing using selective NLP and first order thesauri. *RIAO 91*, 1991;624-644.

[29] Salton G, Buckley C, Smith M. On the application of syntactic methodologies in automatic text analysis. *Information Processing and Management*, 1990;26:73-92.

[30] Dillon M, Gray AS. FASIT: A fully automatic syntactically based indexing system. In Hersh WR, Hickam DH, Leone TJ. (1993). Words, concepts, or both: Optimal indexing units for automated information retrieval. *Journal of American Medical Informatics Association*, 1993;644-648.

[31] Vries JK, Marshalek B, D'Abarno JC, Yount RJ, Dunner LL. An automated indexing system utilizing semantic net expansion. *Computers and Biomedical Research*, 1992;153-167.

[32] Lewis DD, Spark Jones, K. Natural language processing for information retrieval. *Communications of the Association for Computing Machinery*, 1996;39(1):92-101.

[33] Voorhees. EM. Using WordNet to disambiguate word senses for text retrieval. In Korfhage R, Rasmussen E, Willet P. (eds.) *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993:171-180.

[34] Evans D, Hersh W, Monarch I, Lefferts R, Handerson S. Automatic indexing of abstracts via natural language processing using a simple thesaurus. *Medical Decision Making*, 1991;11:S108-S115.

[35] Broglio J, Callan JP, Croft WB. INQUERY system overview. Computer Science Department, University of Massachusettes. 1993. Available: http://ciir.cs.umass.edu:80/info/inquery.html

[36] Hersh WR, Hickam DH, Haynes B. A performance and failure analysis with a MEDLINE test collection. *Journal of the American Medical Informatics Association*, 1994;1:51-60.

[37] Salton G. Allan J, Buckley C. Automatic structuring and retrieval of large text files. *Communications of the ACM*, 1994;37(2): 97-108.

[38] Borko H. Bernick M. Automatic document classification. *Journal of the Association for Computing Machinery*, 1963;10:151-162.

[39] Koll M. WEIRD: An approach to concept-based information retrieval. *SIGIR Forum*, 1979;13:32-50.

[40] Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 1990; 41(6):391-407.

[41] Letsche TA. *Toward large-scale information retrieval using latent semantic indexing.* Masters Thesis, Department of Computer Science, University of Tennessee, 1996.

[42] Healy MJR. *Matrices for statistics*. London: Oxford University Press, 1986.

[43] Golub G. & Reinsch C. Handbook for automatic computation II, linear algebra. New York: Springer-Verlag, 1971.

[44] Jackson D. Stopping rules in principle components analysis: A comparison of heuristical and statistical approaches. *Ecology*. 1993;74(8):2204-2214.

---

[45] Deerwester S, Dumais ST, Furnas GW, Landauer TK, Harshman R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 1990; 41(6):391-407.

[46] Foltz PW, Dumais ST. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 1992;35(12):51-60.

[47] Hull D. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995:229-237.

[48] Shutze H, Pederson J, Hull DA. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995:229-237.

[49] Dumais ST. Latent semantic indexing (LSI) and TREC-2. In *The Second Text REtrieval Conference (TREC-2)*. National Institute of Standards and Technology Special Publication 500-215, Gaithersburg, Md., 1994.

[50] Dumais ST. Latent semantic indexing (LSI): TREC-3 Report. In *The Third Text REtrieval Conference (TREC-3)*. National Institute of Standards and Technology Special Publication 500-225, Gaithersburg, Md., 1995.

[51] Vaithyanathan S, Anick P. A multiple PCA (MPCA) model for hierarchical decomposition of large document collections. IBM research report, RJ 10082 (91898), 1997.

[52] Korfhage RR. To see or not to see: Is that the query? In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1991:134-141.

---

[53] Dubin D. Document analysis for Visualization. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995:199-204.

[54] Hemmje M, Kunkel C, Willett A. LyberWorld - a visualization user interface supporting fulltext retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994:199-204.

[55] Kamada T. Kawai S. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 1989;31:7-15.

[56] Kamar A. Fowler RH. A spring modeling algorithm to position nodes of an undirected graph in three dimensions. Dept. Computer Science, University of Texas Technical Report, 1994. Available: http://bahia.cs.panam.edu/info_vis/spr_tr.html

[57] Chalmers M. Chitson P. Bead: Explorations in information visualization. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992:330-337.

[58] Swan RC & Allan J. Improving interactive information retrieval effectiveness with 3-D graphics. Unpublished manuscript, 1996. Available: http://ciir.cs.umass.edu/info/psfiles/irpubs/ir-100.ps.gz

[59] McDaniel, J. Personal communication, August 5, 1997.

[60] van Rijsbergen CJ. *Information Retrieval*. London: Butterworths, 1975.

[61] Willett P. Recent trends in hierarchical document clustering: A critical review. *Information Processing and Management*, 1988;24(5):577-597.

[62] Cutting DR, Karger DR, Pedersen JO, Tukey JW. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992;318-329.

[63] Cutting DR. Karger DR & Pedersen JO. Constant interaction-time Scatter/Gather browsing of very large document collections. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993;126-135.

[64] Hearst MA. Pedersen JO. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1996;76-84.

[65] Gundavaram S. *CGI programming on the World Wide Web*. Sebastopol: O'Reilly and Associates, 1996.

[66] Davies R. Newmat (version 9) [computer software], 1997. Available http://nz.com/webnz/robert/

[67] Flanagan D. *JavaScript: The definitive guide*. Sebastopol: O'Reilly and Associates, 1997.

[68] Bell G, Parisi A, Pesce M. *The virtual reality modeling language: Version 1.0 specification*, 1996. Available: http://www.vrml.org/VRML1.0/vrml10c.html.

[69] Berry M, Do T, O'Brien G, Krishna V. & Varadhan S. *SVDPACKC (Version 1.0) User's Guide*. Department of Computer Science, University of Tennesee, 1993. Available: http://www.cs.utk.edu/~lsi/ link: ut-cs-93-194.

[70] Buckley C. SMART (version 11) [computer software], 1994. Available: ftp://ftp.cs.cornell.edu/pub/smart/smart.11.0.tar.z

[71] Calinski RB. & Harabasz J. A dendrite method for cluster analysis. *Communications in statistics*, 1974;3:1-27.

[72] Kaufman L. & Rousseeuw PJ. *Finding groups in data: An introduction to cluster analysis*. New York: Wiley, 1990.

[73] Haynes R, McKibbon K, Walker C, Ryan N, Fitzgerald D, Ramsden M. Online access to MEDLINE in clinical settings. *Annals of Internal Medicine*, 1990;112:78-84.

[74] Hersh W, Buckley C, Leone TJ, Hickman D. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994:192-201.

[75] Hersh WR, Hickam DH, Use of a multi-application computer workstation in a clinical setting, *Bulletin of the Medical Library Association*, 1994;82:382-389.

[76] Hull D. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993:329-338.

[77] Yang Y. An evaluation of a statistical approaches to MEDLINE indexing. *Proceedings of the 1996 Annual Full Symposium of the American Medical Informatics Association (AMIA'96)*, 1996:358-362.

[78] Schaller R. Moore's law: Past, present, and future. *IEEE Spectrum*, 1997;34(6):52-59.

[79] Belkin NJ. Oddy RN & Brooks HM. Ask for information retrieval: Part 1. background and theory. In *Readings in information retrieval*. San Francisco, CA: Morgan Kaufmann, 1997.

[80] Dumais ST. Enhancing performance in latent semantic indexing (LSI) retrieval. Bellcore Technical Report, 1992.

Appendix A: A Concrete Example of Latent Semantic Indexing

The following example illustrates the LSI method. The matrix algebra used in the example is described in Appendix B. Table A.1 is a simple data set consisting of nine article titles. Only words appearing in more than one title were selected for indexing—these words are italicized. In an actual IR system, the entire abstract or document would be used. The example article titles can be categorized into two main topics: hypertension (labeled h1-h5) and treatment of substance abuse (labeled a1-a4). The values in Table A.2 represent the frequencies that each term occurs in each document. In a practical setting, more complex term weights would be used.

Table A.1: Example document titles

h1: *Report* of the Federal-Provincial working *group* on the *prevention* and control of *High Blood Pressure*
h2: Primary *prevention* of essential *hypertension:* survey of a WHO scientific *group*
h3: Controlling your *blood pressure*: a *guide* to the *prevention* of *heart disease*
h4: Speaking of *heart disease*: popular conceptions of *hypertension* in American *culture*
h5: *High blood pressure* and *culture* change: acculturation and *disease* in the West Indies.

a1: An outcome-based assessment of *residential* drug and *alcohol treatment* provider *programs* in BC.
a2: A case study of 20 in-patient *alcohol* rehabilitation centers in Seattle
a3: Are *residential programs* necessary for moderate severity *substance* misuse?
a4: Special *report* on *treatment* facilities for *substance* abuse

Table A.2: Term frequencies in each of the nine documents[i]

terms	documents								
	h1	**h2**	**h3**	**h4**	**h5**	**a1**	**a2**	**a3**	**a4**
prevention	1	1	1	0	0	0	0	0	0
hypertension	0	1	0	1	0	0	0	0	0
group	1	1	0	0	0	0	0	0	0
high	1	0	0	0	1	0	0	0	0
blood	1	0	1	0	1	0	0	0	0
pressure	1	0	1	0	1	0	0	0	0
heart	0	0	1	1	0	0	0	0	0
disease	0	0	1	1	1	0	0	0	0
culture	0	0	0	1	1	0	0	0	0
report	1	1	0	0	0	0	0	0	1
residential	0	0	0	0	0	1	0	1	0
treatment	0	0	0	0	0	1	0	0	1
programs	0	0	0	0	0	1	0	1	0
alcohol	0	0	0	0	0	1	1	0	0
substance	0	0	0	0	0	0	0	1	1

In this very simple example, only two constructs are necessary to produce an acceptable solution. Once the document (*D*) and term (*T*) spaces have been derived from an SVD of the matrix in Table A.2, they are scaled by the eigenvalues in *S*. In this example two eigenvalues are retained Figure A.1 shows scaled documents and terms plotted in the reduced dimensional LSI space.

[i] A zero in a cell indicates that the term does not occur in that particular document.

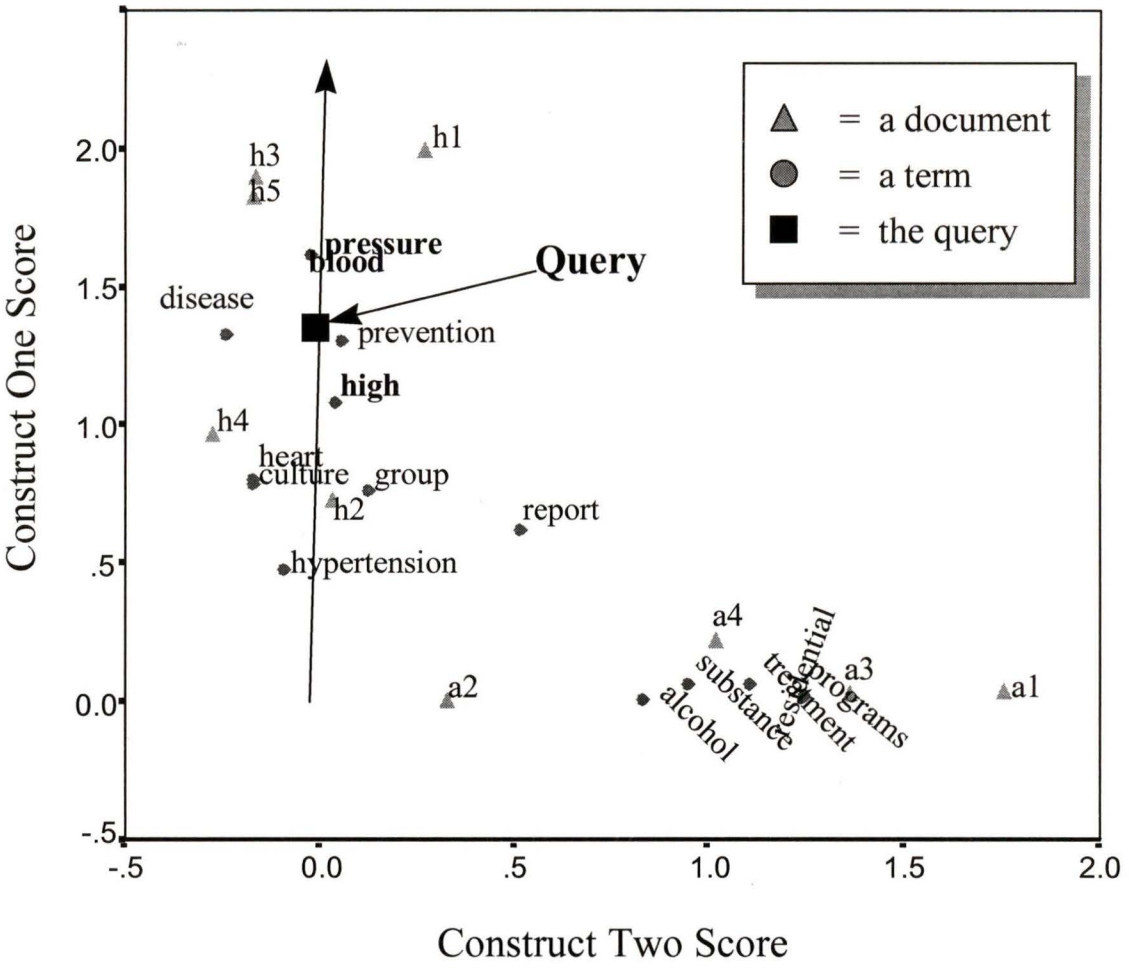


Figure A.1: Two dimensional plot of example terms and documents in LSI space

Figure A.1 is two-dimensional plot of 15 terms and nine documents resulting from an LSI analysis of the matrix shown in Table A.2. Documents are depicted as triangles and terms are shown as circles. The query `Chronic high blood pressure` is presented as a pseudo document at the square labeled by the word "Query". The arrow represents the direction of the query as it originates from the origin of the graph. The distance of an article to the query arrow is its degree of dissimilarity with respect to the query.

A simple test of the model is to find documents relative to the query `chronic high blood pressure`. Vector space models or Boolean term matching techniques would return documents h1, h3, and h5 because they each contain at least one word occurring in the query list. However, two other documents, h2, and h4, would be missed because they do not contain any words in common with the query. With LSI, a query is represented in the LSI space as pseudo-document. Three of the terms in the example query exist in one or more documents in the collection. The word `chronic` does not exist in the space so it is simply omitted. To locate documents similar to the query, the pseudo-document is placed at the centroid of the terms `high`, `blood`, and `pressure`,

and scaled for comparison to documents. The query terms are shown in bold type in Figure A.1. Then, those documents that are close to the query vector in the LSI space are returned to the user. In the example depicted in Figure A.1, documents h1-h5 are nearby the query pseudo-document, but not documents a1-a4. Notice that even documents h2 and h4 are near the query even though they do not contain any of the words in the query.

Appendix B: SVD Example

This Appendix presents a numerical example using the same term by document matrix of hypertension and alcohol treatment articles discussed in Appendix A.

Here is the example 15-term by 9-document matrix from the LSI section in Appendix A.

$X =$

1	1	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0
1	0	1	0	1	0	0	0	0
1	0	1	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	0	1	1	1	0	0	0	0
0	0	0	1	1	0	0	0	0
1	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	1	1

Any rectangular matrix, such as the term by document matrix X, can be decomposed uniquely into the product of three matrices:

$X = T_0 S_0 D_0'$

such that T and D have orthonormal columns and S is diagonal. This is the singular value decomposition of X.

Computing the SVD of the X matrix results in the following three matrices for T_0, S_0, D_0 (rounded to three decimal places to make them fit on the page).

T (9-dimensional left-singular vectors for 15 terms)
S (diagonal matrix of 9 singular values)
D (9-dimensional right-singular vectors for 9 documents)

$T_0 =$

0.371	0.022	0.292	0.409	-0.201	0.294	-0.065	-0.029	-0.214
0.136	-0.038	-0.201	0.623	0.116	-0.289	0.114	-0.011	-0.098
0.219	0.048	0.367	0.400	-0.083	-0.244	0.165	-0.008	0.032
0.307	0.016	0.167	-0.283	-0.002	-0.417	0.220	0.005	0.151
0.460	-0.010	0.092	-0.273	-0.120	0.122	-0.010	-0.016	-0.095
0.460	-0.010	0.092	-0.273	-0.120	0.122	-0.010	-0.016	-0.095
0.230	-0.069	-0.369	0.220	0.066	0.379	-0.243	0.069	0.551
0.377	-0.096	-0.475	-0.051	0.080	0.077	-0.061	-0.018	-0.225
0.225	-0.070	-0.400	-0.060	0.197	-0.461	0.169	0.003	0.021
0.178	0.204	0.337	-0.022	0.484	-0.089	-0.235	0.109	0.511
0.005	0.495	-0.152	0.008	-0.230	0.085	0.319	-0.134	0.191
0.020	0.440	-0.028	-0.004	0.244	-0.120	-0.482	-0.603	-0.273
0.005	0.495	-0.152	0.008	-0.230	0.085	0.319	-0.134	0.191
0.003	0.331	-0.114	0.007	-0.427	-0.319	-0.509	0.574	-0.065
0.020	0.378	0.001	-0.007	0.527	0.257	0.254	0.502	-0.368

$S_0 =$

3.528	0	0	0	0	0	0	0	0
0	2.514	0	0	0	0	0	0	0
0	0	2.218	0	0	0	0	0	0
0	0	0	1.864	0	0	0	0	0
0	0	0	0	1.583	0	0	0	0
0	0	0	0	0	1.359	0	0	0
0	0	0	0	0	0	1.301	0	0
0	0	0	0	0	0	0	0.693	0
0	0	0	0	0	0	0	0	0.559

$D_0 =$

0.565	0.107	0.607	-0.023	-0.026	-0.156	0.050	0.064	0.518
0.206	0.012	0.206	0.768	-0.106	-0.176	0.165	-0.070	-0.501
0.538	-0.065	-0.166	0.017	-0.186	0.731	-0.299	-0.015	-0.137
0.275	-0.109	-0.651	0.393	0.290	-0.217	-0.017	0.063	0.446
0.519	-0.068	-0.237	-0.504	0.022	-0.410	0.236	-0.061	-0.434
0.009	0.700	-0.201	0.010	-0.407	-0.199	-0.271	-0.430	0.080
0.001	0.131	-0.051	0.004	-0.270	-0.235	-0.392	0.828	-0.117
0.009	0.544	-0.137	0.004	0.042	0.314	0.686	0.337	0.027
0.062	0.407	0.140	-0.018	0.793	0.035	-0.356	0.012	-0.233

Except for rounding error,

$X=T_0S_0D_0'$

T_0 and D_0 have orthogonal, unit length columns so that $T_0T_0'=I$, and $D_0D_0'=I$

To approximate X, only the first two singular values and the corresponding vectors from the T and D matrices are kept. In this reduced model,

$X \approx \hat{X} = TSD'$

Where:

$T =$

0.371	0.022
0.136	-0.038
0.219	0.048
0.307	0.016
0.460	-0.010
0.460	-0.010
0.230	-0.069
0.377	-0.096
0.225	-0.070
0.178	0.204
0.005	0.495
0.020	0.440
0.005	0.495
0.003	0.331
0.020	0.378

$S =$

3.528	0
0	2.514

$D' =$

0.565	0.206	0.538	0.275	0.519	0.009	0.001	0.009	0.062
0.107	0.012	-0.065	-0.109	-0.068	0.700	0.131	0.544	0.407

Multiplying out the matrices TSD' gives $\hat{X}$, the estimate of X.

$\hat{X}$ =

0.746	0.270	0.701	0.354	0.675	0.051	0.008	0.041	0.103
0.261	0.098	0.265	0.142	0.256	-0.063	-0.012	-0.048	-0.009
0.449	0.160	0.407	0.199	0.392	0.091	0.016	0.072	0.096
0.617	0.224	0.581	0.293	0.559	0.038	0.006	0.031	0.083
0.914	0.334	0.875	0.448	0.843	-0.002	-0.002	0.000	0.090
0.914	0.334	0.875	0.448	0.843	-0.002	-0.002	0.000	0.090
0.441	0.165	0.449	0.242	0.433	-0.114	-0.022	-0.087	-0.020
0.727	0.271	0.732	0.392	0.707	-0.156	-0.031	-0.120	-0.016
0.430	0.161	0.438	0.237	0.423	-0.116	-0.023	-0.089	-0.023
0.410	0.135	0.304	0.116	0.290	0.365	0.068	0.285	0.248
0.144	0.019	-0.071	-0.130	-0.075	0.870	0.164	0.676	0.507
0.159	0.028	-0.033	-0.101	-0.038	0.775	0.146	0.602	0.454
0.144	0.019	-0.071	-0.130	-0.075	0.870	0.164	0.676	0.507
0.095	0.012	-0.048	-0.087	-0.051	0.582	0.109	0.452	0.339
0.142	0.026	-0.024	-0.084	-0.028	0.666	0.125	0.517	0.391

The $\hat{X}$ matrix does not exactly match the original term by document matrix X, although it becomes a closer approximation as more singular values are retained. An approximate fit is desirable in this case, given that some of the terms in the X matrix are synonymous.

Appendix C: Sample Test Collection Citation

The sections of each article in the MEDTEST and OHSUMED collections are delimited by a code character to aid in the automated indexing process. For instance, in the MEDTEST collection, the delimeters are:

```
.I      document-ID
.T      title
.A      authors
.S      source
.W      abstract
.M      MeSH headings
```

The following is a sample article from the MEDTEST collection:

```
.I 2337
.T
Anaesthesia for trans-sternal thymectomy in myasthenia gravis.
.A
Redfern N ; McQuillan PJ ; Conacher ID ; Pearson DT

.S
Ann R Coll Surg Engl 1987 Nov;69(6):289-92

.W
A retrospective review is presented of the thirty patients who
underwent trans-sternal thymectomy for myasthenia gravis in our
unit from 1980-85. The clinical status of these patients is
contrasted to that of more severely debilitated patients described
by other authors. The problems encountered by the anaesthetist in
the perioperative care of patients with mild myasthenia gravis are
discussed. Management of the perioperative anticholinesterase
regime is described and a case presented for the use of
suxamethonium for intubation. A less invasive postoperative regime
is advocated in which tracheostomy and nasotracheal intubation are
avoided, and anticholinesterase therapy is re-introduced orally as
soon as possible after surgery.
.M
Adolescence
Adult
Aged
Anesthesia/*METHODS
Case Report
Female
Human
Intraoperative Care/METHODS
Male
Middle Age
Myasthenia Gravis/*SURGERY
Retrospective Studies
*Thymectomy
```

Glossary of Acronyms

CGI

The Common Gateway Interface (CGI) is a standard that allows World Wide Web servers to communicate with other programs that run on the server.

IR

Information Retrieval (IR) refers to the study and practice of indexing and retrieving unstructured textual and non-textual information.

LLSI

Local Latent Semantic Indexing is an extension of Latent Semantic Indexing (LSI). In LLSI, a singular value decomposition is performed on a set of LSI document vectors. The resulting right eigenvector matrix can be used as a reduced feature set, or for visualization purposes.

LSI

Latent Semantic Indexing is an indexing and retrieval method that derives orthogonal indexing constructs from the patterns of word use throughout an entire collection. The goal of the LSI is to overcome problems of synonymy, polysemy and term dependence by mapping terms and documents to a small set of orthogonal constructs.

MeSH

Medical Subject Headings are a controlled vocabulary of over 17,000 terms that is used to tag all articles indexed by MEDLINE. MeSH headings are hierarchical in nature.

OHSUMED

A fairly large test collection (350,000 abstracts) taken from MEDLINE, OHSUMED originated at Oregon Health State University.

PCA

Principle Components Analysis (PCA) is a statistical dimension reduction procedure that is typically implemented using the Singular Value Decomposition. PCAs are widely used by researchers in the sciences and social sciences for determining which variables in a larger set form coherent subsets that are relatively independent of one another.

SDI

Selective Dissemination of Information (SDI) is synonymous to routing. In this process, the IR system has some information about a user's preferences. The system uses this information to predict which incoming documents will be of interest to the user.

SMART

SMART began as an experimental implementation of the Vector Space Model developed at Cornell University. The system is now used by researchers world wide.

SVD

The Singular Value Decomposition (SVD) is a dimension reduction technique for decomposing any matrix into three sub-matrices. The SVD is at the heart of Latent Semantic Indexing.

TREC

The Text Retrieval Conference is held annually, and currently boasts over 50 active participants. Each year, a new set of documents and queries are distributed to participants. The size of the collection is very large (e.g., 3 gigabytes).

UPGMA

The Unweighted Pair-Group Method using arithmetic Averages (UPGMA) is a hierarchical clustering method. UPGMA defines the distance between two clusters as the average of the distances between all pairs of cases in which one member of the pair is from each of the clusters.

VIBE

A reference point based visualization system initially developed by Robert Korfage at the University of Illinois and later improved upon by David Dubin.

VRML

Virtual Reality Modeling Language (VRML) is an interpreted language that allows one to define simple three dimensional scene. A VRML compliant browser or plug-in to a Web browser is required to view a VRML scene.

VSM

The Vector Space Model (VSM) is a method for indexing and retrieving documents that ranks results by order of estimated relevance to a query. Unlike Boolean models, the VSM accounts for the distribution of terms over an entire corpus as well as a document.

Vita

Surname: Miller Given Names: Michael Hugh

Place of Birth: London, Ontario, Canada

Educational Institutions Attended:

University of Victoria	1987 to 1990, 1992 to 1997
Dalhousie University	1990 to 1991

Degrees Awarded:

B.A. (Honours)	University of Victoria	1992

Honors and Awards:

DR. JULIUS F. SCHLEICHER GRADUATE SCHOLARSHIP	1995-1996
ITCH '97 Student Poster Award	1996
COACH Student Highest Achievement Award	1997

Publications:

Miller MH. A bottom-up, concept based methodology for indexing and searching MEDLINE. In *Information Technology and Community Health Proceedings '96.* 1996:22-27.

Miller MH. 3-Dimensional Information Retrieval with Local Latent Semantic Indexing. In *COACH 97 Proceedings*, 1997.

Miller MH. A Method for Representing Search Results in Three Dimensions. To appear in *Journal of the American Medical Informatics Association Symposium on Computer Applications in Medical Care '97,* 1997.

1. Partial Copyright License

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the university designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis:

Applying Local Latent Semantic Indexing for Information Retrieval Visualization

Author: [redacted]

Michael Hugh Miller
October 14, 1997.