12-1999

Level search schemes for scalable information retrieval

Xiaoyan Zhang

To the Graduate Council:

I am submitting herewith a thesis written by Xiaoyan Zhang entitled "Level search schemes for scalable information retrieval." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Michael W. Berry, Major Professor

We have read this thesis and recommend its acceptance:

Padma Raghavan, Peiling Wang

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Xiaoyan Zhang entitled "Level Search Schemes for Scalable Information Retrieval". I have examined the final copy of this thesis for form and content and recommended that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Michael W. Berry

Michael W. Berry, Major Professor

We have read this thesis and recommend its acceptance:

Padma Raghavan

Accepted for the Council:

Associate Vice Chancellor and
Dean of the Graduate School

Level Search Schemes
For Scalable Information Retrieval

A Thesis
Presented For

The Master of Science
Degree

The University of Tennessee, Knoxville

Xiaoyan Zhang

December, 1999

Acknowledgement

I would like to thank my major professor Dr. Michael Berry for his support and guidance throughout this research. I would also like to thank Dr. Padma Raghavan and Dr. Peiling Wang for being my committee members and their insightful comments and suggestions.

Support for this research was provided by the National Automotive Center under subcontract number 99732-PA-T60 with Lockheed Martin Energy Systems.

Abstract

Latent Semantic Indexing (LSI) has been demonstrated to outperform lexical matching in information retrieval. However, the enormous cost associated with the Singular Value Decomposition (SVD) of the large term-by-document matrix becomes a barrier for its application to scalable information retrieval. This thesis shows that information filtering using level search techniques can reduce the SVD computation cost for LSI. For each query, level search extracts a much smaller subset of the original term-by-document matrix with an average of 25% of the original non-zero entries. When LSI is applied to such subsets, the average precision only degrades by 5% due to level search filtering; however, for some document collections an increase in precision has been observed. Level search techniques are enhanced by a pruning scheme that deletes terms connected to only one document from the query-specific submatrix. An average 65% reduction in the number of non-zeros has been observed with a precision loss of 5% for most collections.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Latent Semantic Indexing (LSI) is a concept-based textual retrieval method initially developed by Deerwester *et al.* in 1991 [1][2][3]. It tries to overcome the problems of lexical matching by using statistically derived conceptual indices instead of individual words for retrieval. The bulk of LSI processing time is spent in computing the truncated SVD of the large sparse term-by-document matrices [4]. This thesis demonstrates a graph-based approach (level search) as an *information filtering* step for LSI. Level search greatly reduces the cost of SVD for LSI by extracting much smaller subsets from the initial term-by-document matrix for each query. The rest of this chapter will provide an overview of LSI, information filtering technology, and some basics of graph theory.

1.1 Latent Semantic Indexing (LSI)

In the traditional vector space model of information retrieval [5], both terms and documents are encoded as the vectors in a k-dimensional space. The choice of k can be based on the number of unique terms or concepts associated with the collection [6]. Normally, a value (*weighting*) is assigned to a component reflecting the importance of a term or concept in representing the semantics of the corresponding document. Efficiency in indexing via vector space modeling requires special encodings for terms and documents in a text collection. The encoding of term-by-document matrices for lower dimensional vector spaces using either continuous or discrete matrix decompositions are required for LSI-based indexing [6]. Different from lexical matching, LSI uses the Singular Value Decomposition (SVD) from linear algebra to uncover underlying associations among terms and documents for a semantic or conceptual subspace. LSI has been demonstrated to be 30% more effective than popular word-matching methods in producing a high number of relevant documents (recall) [3]. It is especially beneficial when: a) text descriptions are short; b) user queries or text are noisy (spanning across multiple semantic spaces) or c) cross language retrieval without the need for direct translation [5]. The LSI procedure can be categorized into 3 steps: matrix construction, SVD, and query matching.

1.1.1 Matrix Construction

A document collection is first processed using a *stoplist* to remove common words. From a data compression viewpoint, the stoplist eliminates the need to handle unnecessary words and thereby reduces the amount of time and space required to build searchable data structures [6]. A document collection with n documents and m indexable terms can be represented as an $m \times n$ term-by-document matrix M. Each column of the matrix M can be considered as a document vector and similarly each row can be considered as a term vector. The matrix element M_{ij} denotes the frequency in which term i occurs in document j [1]. The matrix M is usually sparse since each term does not appear in every document. In practice, *local* and *global weightings* are applied [8] to each matrix element in order to model the importance of terms both within and across the documents. Hence, the element M_{ij} can be written as

$$M_{ij} = L(i, j) \times G(i), \tag{1-1}$$

where $L(i, j)$ is the local weighting for term i in document j, and $G(i)$ is the global weighting for term i. Note that local weighting applies to a term in a specific document, and is therefore document specific. However, the global weighting of a term is constant for all documents. For simplicity, the global weighting can be set to 1.0 so that only the local frequency is used to define each matrix element. The matrix construction technique discussed in Section 2.1 is based on this simplicity.

Since a term-by-document matrix is typically large and sparse, only non-zero values are stored using formats such as compressed Row Storage (CRS) and Compressed Column Storage (CCS) [9]. The CCS format, also known as the Harwell-Boeing format [10], uses a row index array, a column pointer array and a value array to record the location and value of each non-zero element. All sparse matrices used in this study are stored in the Harwell-Boeing format.

1.1.2 Singular Value Decomposition (SVD)

The dominant computational step of LSI is to factor the term-by-document matrix into a product of 3 matrices using the SVD. As shown in Figure 1.1, the SVD breaks down the original matrix into the orthogonal matrices U and V containing left and right singular vectors of M, and the diagonal matrix Σ of singular values of M. Note each column (vector) of U and V is linearly independent. Let M_k denote the best rank-k approximation to matrix M. Then, the use of k factors (or the k-largest triplets) of M is equivalent to approximating the original term-by-document matrix by M_k. In Figure 1-1, singular vectors defined by the columns of U and V are considered as term and document vectors, respectively. The shaded regions in U (U_k) and V (V_k) and the diagonal elements in Σ (Σ_k) represent the low rank approximation of M by M_k.

Although the SVD is an essential part of the LSI process, it does incur a

M_k = Best rank-k approximation to M $U = \{U_1, U_2, \ldots, U_m\}$ (Term Vectors) $\Sigma = \text{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ (Singular Values) $V = \{V_1, V_2, \ldots, V_n\}$ (Document Vectors)	m = number of terms n = number of documents k = number of factors r = rank of M

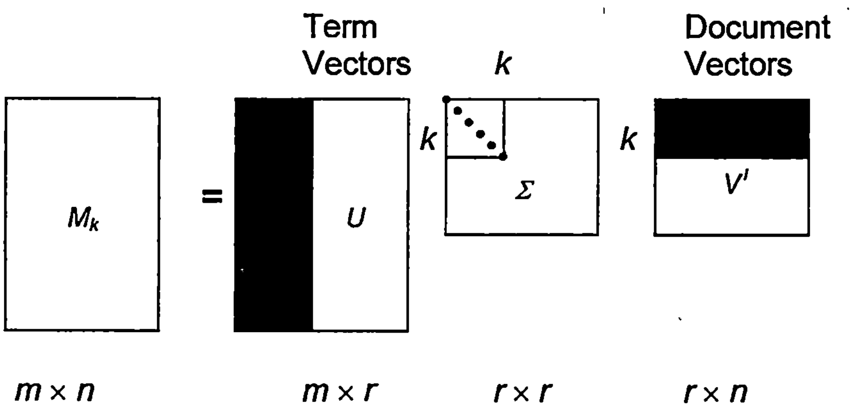

Figure 1-1: Mathematical representation of the matrix M_k [1].

significant computational cost [11][12]. In general, the cost of computing the SVD of a sparse matrix M can be expressed as

$$I \times \text{cost}\,(M^T M\, x) + k \times \text{cost}\,(M\, x), \qquad (1\text{-}2)$$

where I is the number of iterations required by a Lanczos-type procedure [10] to approximate the eigensystem of $M^T M$, and k is the number of computed singular values and their corresponding left and right singular vectors. In general, the cost of the SVD is directly proportional to the number of non-zero entries in the sparse matrix M.

1.1.3 Query Matching

A user's query must be represented in the k-dimensional semantic space and compared to documents for the purpose of information retrieval. Like any document, a query is composed of a set of words. After removing common words (stoplist, see Section 1.1.1) such as "a", "the" and "or" from the query, a vector $q = (q_1, q_2, \ldots, q_m)$ can be defined, where each element q_i is the frequency of the i-th term in the query. The projected query $\hat{q}$ can then be represented in a k-dimensional space by

$$\hat{q} = q^T U_k \Sigma_k^{-1}. \tag{1-3}$$

The sum of the corresponding k-dimensional term vectors is reflected in the $q^T U_k$ term so that right multiplication by Σ_k^{-1} differentially weights each dimension. Thus, the projected query vector $\hat{q}$ is located at the weighted sum of its constituent term vectors, and can then be compared to all document vectors using some similarity measure. The cosine between the query vector and the document vectors is certainly one common similarity measure that can be used to rank all documents with respect to the query. Typically, documents with cosine values greater than a certain threshold are returned to the user [3].

Having described the mechanics of LSI modeling, the following section introduces the concept of information filtering which can be used to reduce the computational burden of LSI.

1.2 Information Filtering

According to Korfhage [13], one problem associated with large, full-text database searching is that most searches are more likely to return a large volume of documents, some of which are irrelevant. The primary reason is that when full text documents are used instead of much shorter document surrogates, there is an increased chance of word co-occurrence in a non-relevant document.

Information filtering is one possible remedy for this problem. It relies on relatively inexpensive techniques to quickly eliminate large segments of a collection from consideration. Then, a relatively more expensive method can be applied to further process the filtered database and achieve satisfactory performance [14][15]. Although filtering is closely related to information retrieval [17], the goal is not to determine a specific document set to be retrieved. Rather, the goal is to produce a relatively small set containing a high portion of relevant documents. Thus, either a more precise method could be applied to identify the relevant documents, or the user can browse through the set to locate interesting documents. In both ways, the amount of effort required for filtering can be significantly less than that required for retrieval directly from the original large document collection [13]. For example, suppose that retrieval from a set of n documents requires n^2 steps and filtering requires only n steps for a collection of 10,000 documents. While direct retrieval from this set would require $10{,}000^2$ steps, filtering down to a set of 1000 documents (10% of the original collection) followed by retrieval from this smaller set would only require $10{,}000 + 1000^2 = 1{,}010{,}000$ steps. The next sections will discuss two common information filtering techniques.

1.2.1 User Profiling

User-based information can be exploited to assist effective filtering. One popular method is to maintain user's *profile* and use it for document routing and delivery. A user's profile [13] typically contains information specific to each user, such as the user's profession, age, education, personal interests, etc. Since this type of information is relatively stable, the new documents are constantly received and matched against the standing interests.

LSI has shown promising results with user's profiles in information filtering. An initial sample of documents can be analyzed using standard LSI/SVD tools as will be discussed in Chapter 2 [18]. A user's interest is represented as one (or more) vector(s) in the reduced dimension LSI subspace so that each new document can be matched against the profile vector(s). Documents that are judged similar to the profile are recommended to the user. Different methods of representing a user's profile have been reported [18][19] and the results are quite promising.

1.2.2 Passage Retrieval

The concept of *passage retrieval* is closely related to information filtering. Here, the goal is not to quickly eliminate a large portion of a collection, but to identify those passages (or documents) closely related to a given query within a broad document such as encyclopedia [13]. Research efforts have been made in this particular area. Hearst and Plaunt [20] have developed a method called "Text tiling" with a visual interface called "TileBars". This method displays the finer levels (section, paragraph) of each document the extent to which the document relates to the query. Salton and Allan [21] have developed a different display that arranges documents as arcs around an ellipse, with lines joining the documents to show use of the query terms. Much of passage retrieval research is based on the differential analysis of the key terms in a document. If term *A* appears in the query and frequently occurs in a portion D_1 within document *D*, then D_1 is returned instead of document *D*. The remainder of the document *D* is either discarded or held until the user decides to view it.

In summary, various approaches can be used to implement information filtering. The approach taken in this thesis (level search) applies graph theoretic techniques for information filtering. The following section will introduce some basic concepts of graph theory.

1.3 Graph Theory

An undirected graph $G = (V, E)$ consists of a set of vertices V and a set of edges E. Each edge in E is an unordered pair of vertices, while in a directed graph it is an ordered pair. Hereafter an undirected graph will be referred as a graph.

Vertices v and w are adjacent if (v, w) is an edge. A path is a sequence of vertices $v_1, v_2, \ldots, v_n$ such that (v_i, v_{i+1}) is an edge for $1 \leq i < n$ [22]. A path is simple, if all vertices on the path are distinct, with the exception that v_1 and v_n may be the same. The length of a path is $n-1$, the number of edges along the path. A subgraph of $G = (V, E)$ is a graph $G' = (V', E')$ where V' is a subset of V, and E' consists of edges (v, w) in E such that both v and w are in V '. Figure 1-3 illustrates a subgraph of the graph G shown in Figure 1-2.

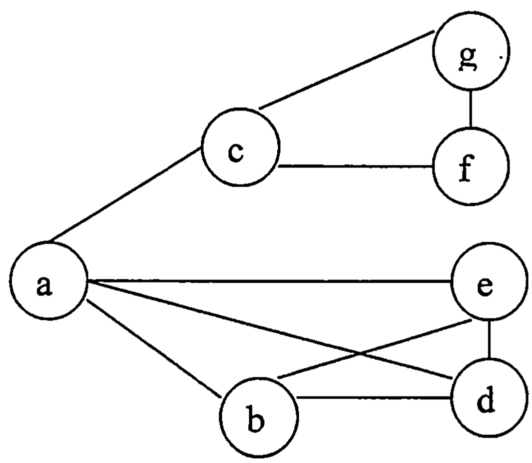

Figure 1-2: A graph $G = (V, E)$.

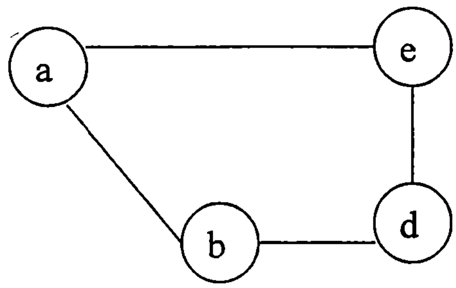

Figure 1-3: A subgraph *G'* = (*V'*, *E'*) of graph *G*.

Two systematic traversal patterns referred to as *depth-first* search and *breadth-first* search are used to visit the vertices of a graph. Breadth first search, also known as level search, is used in this thesis to implement information filtering strategies.

In breadth first search, searches are conducted as broadly as possible by visiting all the vertices adjacent to an arbitrary vertex *v*. The algorithm for breadth-first search (bfs) can be illustrated as follows:

List = v, an arbitrary vertex.
Repeat **bfs**(list)
    S $\in$ head (list)
    Mark S as visited
        For each x adjacent to S
        Mark x as visited
        Append x to list, until the list is empty.

Figure 1-4 shows the breadth-first search for the graph *G* in Figure 1-2. The dash lines in Figure 1-4 represent the edges connecting two vertices neither of which is an ancestor of the other.

How level search applies graph theory as an information filtering tool for LSI is explained in the following chapter. In Chapter 3, level search is applied as a filtering method for *Latent Semantic Indexing* (LSI) and the results are compared to traditional LSI. Finally, a case study on a large text collection is presented before conclusions and suggestions for future work are presented in Chapter 4.

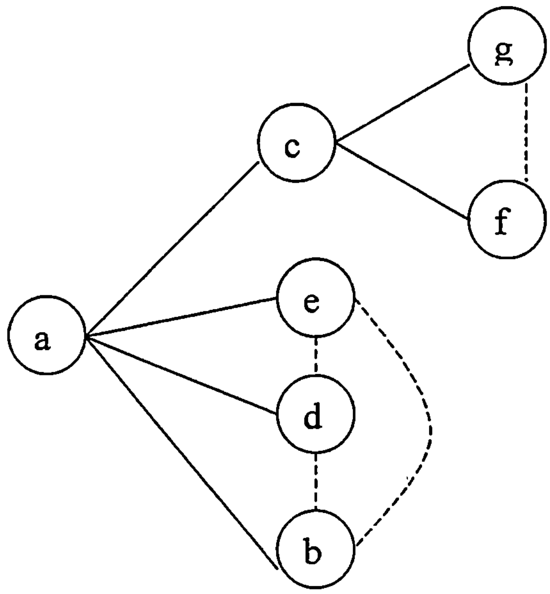

Figure 1-4: Breadth-first search for graph $G = (V, E)$ in Figure 1-2.

Chapter 2

Level Search Technique

Level search is a graph-based IR model, where each query term and each document is represented as the vertices of an undirected graph. Weightings are assigned to the edges of the graph so that documents with higher weighting values (than a certain threshold) are selected as relevant documents to the query. Through the use of weighting schemes, level search can be categorized into simple level search and advanced level search.

2.1 Simple Level Search

To illustrate the simple level search algorithm, consider a small document title collection composed of 5 documents and 10 terms (see Figure 2-1). Although not necessary for this example, a stemming technique [13] could be used when parsing terms for each document. Words with the same root could be considered as the same term.

A transposed 10×5 term-by-document matrix $A = [a_{ij}]$ can be constructed as follows, where each element a_{ij} is the number of times term i appears in document title j:

Terms	***Documents***
T1: Infant	D1: Infant /Toddler Food, Cookbooks & Recipes
T2: Toddler	D2: Vegetarian Recipes
T3: Food	D3: Italian Food
T4: Recipes	D4: Healthy Diet
T5: Healthy	D5: Super Baby Food
T6: Cookbooks	
T7: Baby	
T8: Diet	
T9: Vegetarian	
T10: Italian	

Figure 2-1: Document and term listing for a sample document title collection.

$$A^T = \begin{array}{c|cccccccccc} & T1 & T2 & T3 & T4 & T5 & T6 & T7 & T8 & T9 & T10 \\ \hline D1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ D2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ D3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ D4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ D5 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Since this term-by-document matrix is constructed directly from the document collection, it is denoted as the "initial matrix" throughout the following chapters. Given the query "infant food", a vector q (not the projected query vector $\hat{q}$ as in Equation 1-3) can be formed based on the terms:

$$q = [1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0],$$

where each element q_i denotes the frequency of the i-th term in the query. As Figure 2-2 illustrates, level search builds a graph from the query vector using breadth-first search (Section 1.3).

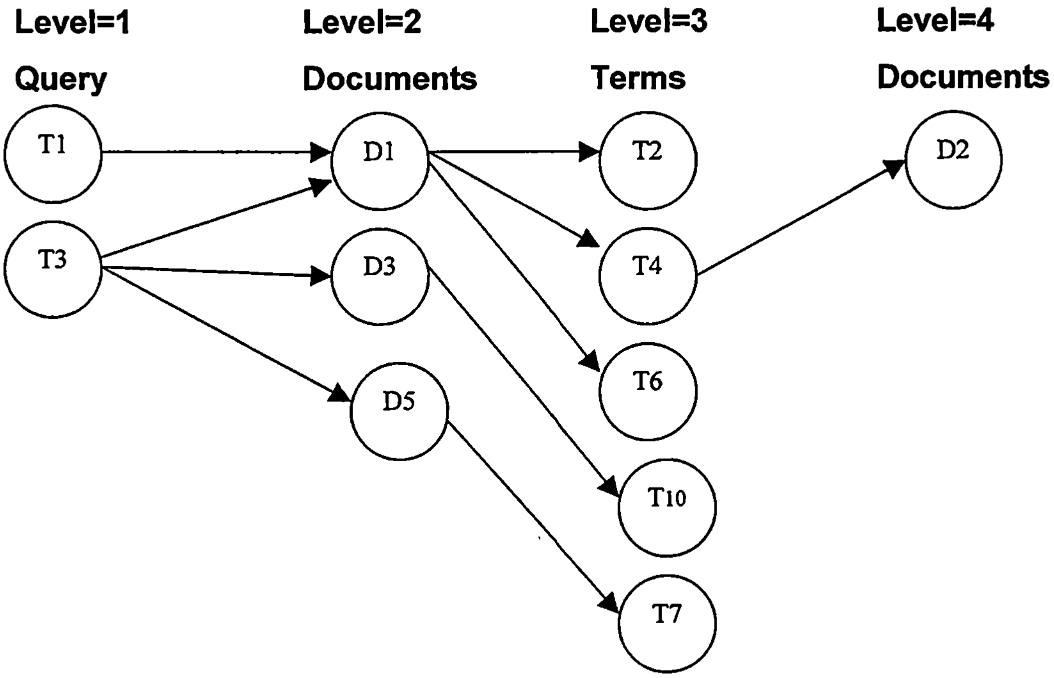

Figure 2-2: Graph construction for query "infant food" in a sample document title collection using simple level search.

As illustrated in Figure 2-2, the query "infant food" (composed of terms T1 and T3) defines level 1. The second level is composed of all the documents that contain terms T1 and T3, which are D1, D3, and D5. The third level is formed by a list of terms that appear in documents D1, D3, and D5. Each term can only appear in the level it is first visited (searched). Likewise, the fourth level contains all the documents that have the terms listed in level 3, which have not previously been selected. In a global sense, level search exploits the path from a query to its related documents as a process to categorize the documents and terms from the initial matrix into level *clusters*. Each odd level is a group of terms and each even level is a group of documents. Three documents are found to be relevant to the query at level 2 while at level 4 a total of four documents (D2 plus documents at level 2) are found. A new term-by-document matrix can be constructed for this query at any document level. At level 2, for example, terms T1 and T3 are only associated with three documents D1, D3 and D5, hence a term-by-document matrix $A = [a_{ij}]$ for the query "*infant food*" at level 2 can be constructed as:

$$A = \begin{array}{c|ccc} & D1 & D3 & D5 \\ T1 & 1 & 0 & 0 \\ T3 & 1 & 1 & 1. \end{array}$$

This 2 by 3 query-specific term-by-document matrix is a subset of the initial 10 by 5 term-by-document matrix. In other words, a new query such as "healthy food" would produce a different submatrix:

$$B = \begin{array}{c|cccc} & D1 & D3 & D4 & D5 \\ T3 & 1 & 1 & 0 & 1 \\ T5 & 0 & 0 & 1 & 0. \end{array}$$

Since the cost of SVD computation is closely related to the non-zero entries in the sparse matrix (Section 1.1.2, Equation 1-2), the ratio of non-zero entries in the submatrix to the original term-by-document matrix can be used to estimate the SVD cost reduction. The row and column reduction will also be calculated. For simplicity, the term *submatrix size* will be used as a general reference to the number of rows, columns and non-zero entries of a submatrix. Before presenting the results, the experimental methodology will be discussed.

2.2 Experimental Methodology

The document collections used in this thesis constitute standard benchmark collections. Each test collection consists of a document set, a collection of queries, and the "correct answers", that is, a list of relevant documents for each query. MEDLINE is a collection of medical abstracts; CISI is a collection of library science abstracts; TIME is a collection of news abstracts from TIME magazine. The FBIS collection is a subcollection of the TREC-5 [25] FBIS (Foreign Bureau Information Service) test set, which is obtained by exploiting available relevance judgements so that each selected document in the subcollection is relevant to at least one query. Note that relevance judgement in this research is provided by the same human being. Some of the characteristics of the collections are presented in Table 2-1.

Table 2-1: Some characteristics of the document collections.

Parameter	MEDLINE	CISI	TIME	FBIS
Number of Documents	1033	1460	425	4625
Number of Terms	5831	5609	10804	42500
Number of Non-zeros (matrix)	52009	68240	83602	1573306
Number of Queries	30	112	82	43
Avg. No of Documents/Term	8.92	12.17	7.74	37.02
Avg. No of Terms/Document	50.35	46.74	196.71	316.31
Density (%)	0.86	0.83	1.82	0.74
Avg. No of Terms/Query	10.27	21.76	7.80	39
Avg. Weighting/Term	0.58	0.46	0.50	0.40

Density (%): the percentage of non-zero entries in the matrix.

As shown in Table 2-1, all the term-by-document matrices are quite sparse (around 1% dense). Both MEDLINE and CISI collections have approximately 50 terms per document, while TIME and FBIS collections have over 200 terms per document (an indication of heterogeneity). Most queries are reasonably specific as they each contain approximately 20 terms [23]. The FBIS queries contain a higher number of terms because they are chosen from TREC *routing* topics (as opposed to *adhoc* topics) for that collection.

The complete testing procedure for level search and LSI is described in Appendix A. The performance measures used to evaluate document retrieval are discussed in the following sections.

Performance Measures

Recall and precision are two standard IR performance measures [7]. The recall (or recall ratio) R for the level search method is defined as

$$R = \frac{D_r}{N_r}, \tag{2-2}$$

where D_r is the number of relevant documents retrieved and N_r is the total number of relevant documents in the collection for a certain query.

The precision (or precision ratio) of level search method is defined as

$$P = \frac{D_r}{D_t},$$

where D_r is the same numerator from Equation (2-2) and D_t is the total number of documents retrieved.

Average Precision

The 11-point interpolated precision is also used in the IR community to assess retrieval performance [24]. However, the use of this measure requires a proper document-ranking scheme. Since level search does not provide such ranking, the 11-point interpolated average precision is only used for LSI performance measurements.

Assume that for each query there is an ordered document list based on how closely each document relates to the query. Let r_i denote the number of relevant documents up to and including position i in the ordered list. A pair of values (recall and precision) are computed for each document in the list. The recall of the i-th document is the proportion of relevant documents returned so far, that is,

$$R_i = \frac{r_i}{r_n}.$$

Here, r_n is the total number of relevant documents returned so far. The precision of the i-th document, P_i, is the proportion of documents returned so far that are relevant and is defined by

$$P_i = \frac{r_i}{i},$$

where i is the number of documents returned. If the pseudo-precision at recall level x ($x \in [0, 1]$) is defined as

$$P(x) = \max \{P_i \mid R_i \geq (x \cdot r_n), \text{i}=1,\ldots,n\},$$

then the N-point interpolated average precision for a single query is defined as

$$P = \frac{1}{N}\sum_{i=0}^{N-1} P[\frac{i}{N-1}].$$

The 11-point interpolated average precision is used for LSI performance assessment at several recall levels (0, 0.1, 0.2, 0.3,...,1.0). For each data collection, the mean 11-point average precision is computed by averaging the precision across all queries.

Precision Improvement

When evaluating the precision obtained by LSI (with and without level search), a relative criterion is needed. Here, precision improvement (PI) is defined as the average precision increase obtained by LSI due to filtering, i.e.,

$$PI = \frac{\textit{(Precision of LSI with level search / pruning) - (precision of LSI)}}{\textit{(Precision of LSI)}} \times 100.$$

Therefore, a precision improvement of "–0.20" suggests the precision of LSI decreases by 20% after applying level search or pruning. A precision improvement of "+0.20" suggests the precision of LSI increases by 20% after applying level search or pruning.

2.3 Simple Level Search Results

Simple level search has been applied to the MEDLINE, CISI and TIME data sets. Table 2-2 shows the average submatrix size, recall, and precision across all queries at level 2 and level 4. Generally speaking, all three data collections show a high recall at near 80% but relatively low precision (less than 20%) at level 2. At level 4, level search simply retrieves the whole document set (ratio of non-zero entries at 100%) with precision less than 3%. This behavior can be explained by taking a further look at the level search algorithm. At any document level, a group of documents related to the query can be collected. As level search traverses more levels, additional documents will be added. Since each document contains at least one term and each term can be found in at least one document, eventually level search will return all documents in the collection. In these experiments, level search reaches its saturation point at level 4, which suggests a rich connectivity among the documents. As presented in Section 2.2, the average document size (number of terms per document) for MEDLINE, CISI and TIME are 50, 47, and 197, respectively. Documents containing a higher number of terms tend to have more common terms with other documents, and therefore have a better chance of being retrieved at lower numbered levels.

For comparison purposes, tests on the same data sets using LSI produced the average precisions of 65.68%, 24.33%, 39.79% for MEDLINE, CISI, and TIME, respectively.

Table 2-2: The average submatrix size, recall, and precision obtained by simple level search at levels 2 and 4 for the MEDLINE, CISI and TIME data sets.

Parameter	MEDLINE		CISI		TIME	
	Level=2	Level=4	Level=2	Level=4	Level=2	Level=4
No. of Docs (% of original)	256 24.78	1033 100	827 56.64	1460 100	183 43.06	425 100
No. of Terms (% of original)	3685 63.20	5831 100	4967 88.55	5609 100	9239 85.51	10804 100
No. of Non-zeros (% of original)	14472 27.83	52009 100	41077 60.19	68240 100	43197 51.67	83602 100
Recall (%)	85.74	100	81.81	100	81	100
Precision (%)	13.31	2.25	4.71	2.78	2.18	0.92

The best precision values obtained for MEDLINE, CISI and TIME using simple level search are 13.31%, 4.71% and 2.18%, respectively. Obviously, they are significantly lower than the 11-point interpolated average precision values obtained through LSI. This observation indicates that simple level search alone offers no improvement in retrieval precision. However, there are still various parameters to be tested in level search. Based on data in Table 2-2, it appears that the submatrix size, recall, and precision are somewhat inter-correlated. As the submatrix increases to the original matrix size, the recall also increments towards 100%. Unfortunately, it is not desirable to sacrifice the submatrix size to gain recall. In this case, weightings are added to terms or documents to further identify relevant documents in a relatively small subset. This simple level search plus weighting technique is denoted as *advanced* level search and will be discussed in the next section.

2.4 Advanced Level Search

In this section, the average submatrix size, recall, and precision results for the MEDLINE, CISI and TIME collections after applying four different weighting strategies will be presented. In general, weightings are applied to either the term levels or the document levels. For each weighting strategy, a threshold value is chosen as the mean, median of the document/term weightings at the same level, or a fixed constant such as 0.5 and 0.75. Only documents/terms with weightings greater than the threshold will be used to generate the next level. In other words, those with weightings less than the threshold will be deleted from the submatrix.

2.4.1 Method 1: Using the Term Global Weightings

A global weighting (*G(i)* in Equation 1-1) is assigned to each term in any term level. Term deletion at the term level may occur since the global weighting is only applied to terms (rather than documents).

The average submatrix size, recall, and precision using the term global weightings are listed in Table 2-3 and Table 2-4. In general, a factor of 50 reduction in the number of non-zero entries is obtained compared to simple level search. At level 4, a subset with the ratio of non-zero entries ranging from 40% to 78% of the initial matrix can be obtained while the recall maintains at 100%. The typical example is the MEDLINE data set, where 0.5 is used as the term weighting threshold. Compared to the simple level search results for MEDLINE in Table 2-2, the ratio of non-zero entries is further reduced from 27.8% to 0.18% with recall decreasing from 85.74% to 69.87%. In other words, each query can

Table 2-3: The average submatrix size, recall, and precision obtained by advanced level search with term global weightings at level 2 for the MEDLINE, CISI and TIME data sets.

Threshold (%) / Parameter	MEDLINE				CISI			TIME		
	Mean	Median	0.5	0.75	Mean	Median	0.5	Mean	median	0.5
# of Docs	44	81	77	9	157	304	146	38	79	25
(% of original)	4.26	7.84	7.45	0.87	10.75	20.82	10	8.94	18.59	5.88
# of Terms	5	6	6	3	9	11	10	4	5	3531
(% of original)	0.08	0.10	0.10	0.05	0.16	0.20	0.18	0.04	0.05	32.68
# of Non-zeros	54	99	95	9	176	355	167	50	104	6529
(% of original)	0.10	0.19	0.18	0.02	0.26	0.52	0.24	0.06	0.12	7.81
Recall	51.21	48.47	69.87	15.56	32.09	50.64	32.01	56.22	69.28	29.92
Precision	33.09	21.26	2.25	32.42	9.7	7.08	9.33	6.71	4.5	4.31

Table 2-4: The average submatrix size, recall, and precision obtained by advanced level search with term global weightings at level 4 for the MEDLINE, CISI and TIME data sets.

Threshold (%) / Parameter	MEDLINE				CISI			TIME		
	Mean	Median	0.5	0.75	Mean	Median	0.5	Mean	median	0.5
# of Docs	1033	1033	1033	1033	1459	1458	1460	425	425	246
(% of original)	100	100	100	100	99.94	99.86	100	100	100	57.88
# of Terms	2744	2415	4569	3077	2423	2095	3864	5503	5442	9282
(% of original)	47.06	41.42	78.36	52.77	43.20	37.35	68.89	50.93	50.37	85.91
# of Non-zeros	24433	21573	40500	27357	29427	25930	45095	43171	42814	43661
(% of original)	46.98	41.48	77.87	52.60	43.12	38.00	66.08	51.64	51.21	52.22
Recall	100	60.96	100	100	99.97	99.87	100	100	100	47.21
Precision	2.25	1.83	2.25	2.25	2.80	2.84	2.83	0.92	0.92	0.57

be used to extract 0.18% of the non-zero entries from the initial term-by-document matrix and maintain 69.87% of the relevant documents. Such a result indicates that the threshold usage in submatrix control is promising in terms of further document extraction. Finally, no precision improvement is observed at level 4. In fact, when the submatrix size is further reduced, both precision and recall degrade.

2.4.2 Method 2: Using the Product of Local Frequency and Global Weighting

Unlike Method 1, Method 2 uses the product of the local term frequency *L(i, j)* and the global term weight *G(i)* as the weighting (see Equation 1-1). Therefore, they are assigned to the edges of the graph illustrated in Figure 2-3. Deletion in this case occurs at the document level. For multiple terms referring to the same document as D1 at level 2, the norm of the two weightings are assigned to D1, *i.e*, $(0.87^2 + 0.47^2)^{0.5} = 0.76$.

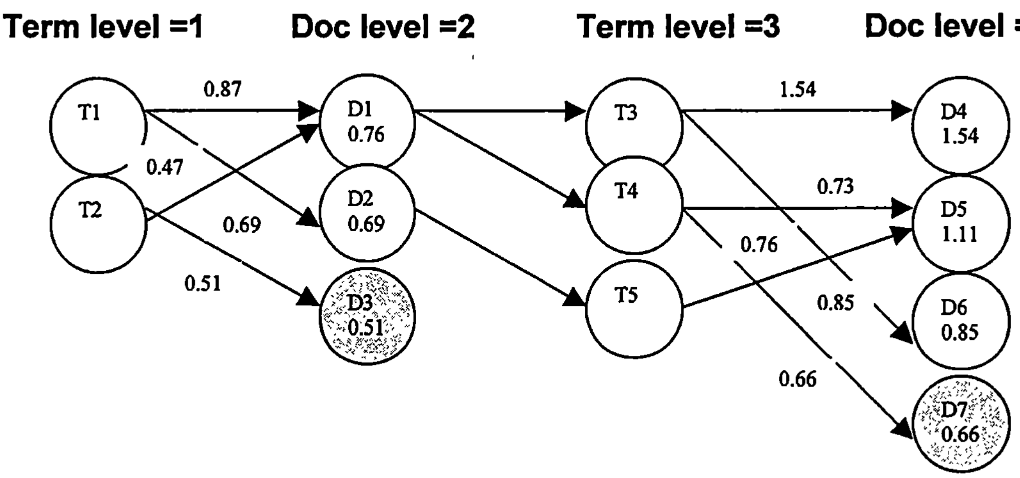

Figure 2-3: Level search structure using the product of local frequency and global weighting as weighting in advanced level search.

Note: circles with shadows are documents that are deleted from the submatrix.

The statistical results are presented in Table 2-5 and Table 2-6. For three document collections tested, this method tends to produce a consistent recall and precision across collections. At level 2, a high recall (~50%) is obtained for both of the CISI and TIME collections compared to Method 1 (~30%), where only term global weightings are used. However, the submatrix size is significantly larger than that generated by Method 1. Also notice that as level search goes deeper to level 4, the recall decreases. Such recall loss might be explained by the fact that when a more stringent criterion is applied to remove the irrelevant documents, some relevant documents are also deleted. There's a tradeoff between the submatrix size reduction and the recall loss. These two parameters are not independent of each other.

2.4.3 Method 3: Using Weightings in the Query Vector only

Method 3 is relatively simple. Weightings are only assigned to the terms in the query vector. The documents and terms in levels 2, 3, 4, ... are considered to be the *family members* of a term in the query vector as long as there is a path to such a term. Thus, those documents/terms inherit the same original weighting with such term. However, the document weighting does accumulate as more than one term appears in the same document. For example, assuming the query vector contains two terms with global weightings of 0.87 and 0.47, respectively, a document at level 2 which contains both of the two terms will have a weighting of 0.87+0.47 = 1.34.

Deletion occurs at each document level, and the threshold used is the mean and median weighting. The statistical results are listed in Table 2-7 and Table 2-8. Compared to Method 2, this method produces almost the same submatrix size and recall for each collection.

Table 2-5: The average submatrix size, recall and precision at level 2 for the MEDLINE, CISI and TIME data sets using the product of global weighting and local frequency as weighting in advanced level search.

Threshold / Parameter(%)	MEDLINE		CISI		TIME	
	Mean	Median	mean	Median	Mean	Median
# of Docs	79	128	460	412	72	91
(% of original)	7.65	12.39	31.51	28.21	16.94	21.41
# of Terms	2033	2664	3687	4047	6650	7569
(% of original)	34.87	45.69	65.74	72.15	61.55	70.06
# of Non-zeros	4601	7348	23846	21778	20780	24812
(% of original)	8.84	14.12	34.94	31.91	24.85	29.68
Recall	55.38	68.5	57.74	59.08	72.43	74.44
Precision	22.25	18.3	8.61	6.16	6.96	4.04

Table 2-6: The average submatrix size, recall and precision at level 4 for the MEDLINE, CISI and TIME data sets using the product of global weighting and local frequency in advanced level search technique.

Threshold / Parameter(%)	MEDLINE		CISI		TIME	
	Mean	Median	mean	Median	Mean	Median
# of Docs	474	517	66	61	188	202
(% of original)	45.88	50.04	45.1	41.1	100	47.53
# of Terms	5416	5500	5119	5002	9442	9521
(% of original)	92.88	94.32	91.27	89.18	87.23	87.97
# of Non-zeros	30581	32799	41829	40980	46566	47100
(% of original)	58.80	63.06	61.3	60.05	55.70	56.34
Recall	43.13	46.23	45.44	54.29	49.13	54.22
Precision	2.11	2.06	2.99	2.99	1.08	1.05

Table 2-7: The average submatrix size, recall, and precision obtained by advanced level search with weightings in the query vector only at level 2 for the MEDLINE, CISI and TIME data sets.

Threshold / Parameter(%)	MEDLINE		CISI		TIME	
	Mean	Median	Mean	Median	Mean	Median
# of Docs (% of original)	76 7.36	155 15.00	313 21.44	493 33.77	65 15.29	115 27.06
# of Terms (% of original)	1893 32.46	2900 49.74	3595 64.09	4370 77.91	6683 61.86	8139 75.33
# of Non-zeros (% of original)	4485 8.62	8791 16.90	17166 25.15	25982 38.07	19006 22.74	29685 35.51
Recall	64.59	81.9	55.11	68.84	69.43	76.92
Precision	27.25	18.7	7.4	5.83	5.92	3.22

Table 2-8: The average submatrix size, recall, and precision obtained by advanced level search with weightings in the query vector only at level 4 for the MEDLINE, CISI and TIME data sets.

Threshold / Parameter	MEDLINE		CISI		TIME	
	Mean	Median	Mean	Median	Mean	Median
# of Docs (% of original)	461 44.63	519 50.24	666 45.62	733 50.20	167 39.29	215 50.59
# of Terms (% of original)	5348 91.72	5473 93.86	5106 91.03	5212 92.92	10111 93.58	10444 96.67
# of Non-zeros (% of original)	30703 59.03	33584 64.57	39564 57.98	42629 62.47	50295 60.16	59030 70.61
Recall	59	63.26	55.91	59.84	54.03	65.59
Precision	3.0	2.83	3.47	3.40	1.38	1.28

2.4.4 Method 4: Using Local Frequency Only

In this approach, only the local frequency of each term-document association ($L(i, j)$, see Equation 1-1) is assigned to the edges of the graph. Deletion occurs at the document levels and for documents containing more than one term from the previous level, the norm of the weightings is assigned as described in Method 2. Here, only results for the MEDLINE data set (Table 2-9) are presented since no significant change has been observed for the other data collections.

Table 2-9: The average submatrix size, recall, and precision obtained by advanced level search using local frequencies as weighting at level 2 and level 4 for MEDLINE.

Threshold / Parameter	Level = 2		Level = 4	
	Mean	Median	Mean	Median
# of Docs (% of original)	75 7.26	211 20.42	519 50.24	1023 99.03
# of Terms (% of original)	1795 30.78	3410 58.48	5473 93.86	5829 99.97
# of Non-zeros (% of original)	4440 8.54	12027 23.12	33584 64.57	51711 99.43
Recall	60.84	83.11	63.26	100
Precision	29.14	15.68	2.83	2.27

2.4.5 Summary of Advanced Level Search

By using thresholds combined with various weighting schemes in advanced level search, a smaller subset of the relevant documents can be extracted from the initial matrix with a slight loss in the recall. The precision of all tests seems relatively low compared to the LSI results. Different weighting schemes seem to have a significant impact on the submatrix size and the recall. The choice of thresholds, (*i.e*, whether to use mean, median, or a constant) also has an important effect. However, there is no general conclusion on which weighting scheme and which threshold value would produce the best result for a specific document collection. Instead, the answer is quite collection specific. Further, it's observed that the submatrix size, recall, and precision are, to some extent, correlated to each other. Further reduction in the submatrix size would cause a corresponding decrease in recall.

2.5 Discussion

Simple level search and advanced level search are both simple IR techniques compared to vector space models such as LSI. They are both able to selectively extract a smaller document set with a relatively high recall. The best precision obtained is relatively low compared to LSI. Such precision performance properly indicates that level search alone, as an information retrieval technique is not satisfactory. The difficulties might lie in the fact that the terms and the documents are so fully connected (in a graph sense) that the further isolation of the relevant documents becomes harder. Using the proper weighting threshold is a fair method to differentiate the relevant documents, but it does not provide a quantitative document-ranking scheme such as the cosine between the query vector and the document vector used in LSI.

Although the precision is not promising, the fact that level search produces high recall values (~80%) while being able to reduce the ratio of the non-zero entries to as low as 0.1% suggests its possible use in information filtering for large document collections. As discussed early in Section 1.1, LSI involves complicated matrix SVD computations, which for large data sets could impose a tremendous cost. Therefore, simple techniques can be applied to reduce the large collection into a smaller set of potentially retrievable documents. The more advanced algorithms could then be applied to the smaller subset. The preprocessing step is intended for high recalls while the second step targets high precision (see Section 1.2). In the following chapter, the use of level search as an information filtering technique for LSI is discussed.

Chapter 3

Level Search for Information Filtering

As previously discussed in Section 1.1.2, the SVD computation for LSI could impose a high computational cost. The focus of this chapter is to demonstrate the viability of level search as a filter for LSI and thereby produce a more scalable indexing method.

Table 3-1 summarizes the submatrix size and recall obtained for level search applied to the MEDLINE, CISI and TIME collections from Chapter 2. It indicates that level search is capable of extracting 68% of the relevant documents for a specific query using as few as 27% of the non-zero entries from the initial term-by-document matrix. Level search certainly exhibits great potential as an information filtering technique. However, to determine whether level search works well as a filter for LSI, further experiments need to be conducted to collect quantitative data. In the following section, the performance of LSI with level search filtering will be presented and compared to traditional LSI. For testing level search with LSI, the best weighting scheme and threshold (see Appendix B) for each data collection is first chosen based on the results obtained in Chapter 2. Level search then uses those weighting schemes and threshold to generate the submatrix for each query, which is later used as input for LSI. Finally, the 11-point interpolated average precision, previously defined in Section 2.2, is used for LSI performance evaluation.

Table 3-1: The average recall and submatrix size for MEDLINE, TIME and CISI after level search filtering.

Collection	Matrix Size (Documents x Terms x Non-zeros)	Average Recall (%)	% Docs of original	% Terms of original	% of Non-zeros of original
MEDLINE	1033 x 5831 x 52009	85.74	24.78	63.20	27.83
TIME	425 x 10804 x 68240	69.42	15.29	61.86	22.74
CISI	1469 x 5609 x 83602	55.11	21.44	64.09	25.16
FBIS	4974 x 42500 x 1573306	82.05	28.52	55.01	52.92
Mean	-	67.79	18.15	53.36	27.00

3.1 LSI with Level Search Filtering

LSI with and without level search filtering has been applied to the same data collections for comparisons in retrieval performance. The precision-recall graphs for MEDLINE, CISI, TIME, and FBIS obtained by LSI with and without level search are presented in Figures 3-1, 3-2, 3-3, and 3-4, respectively. A summary of precision-recall data is also available in Table 3-2.

Based on Figures 3-1 and 3-2, some precision loss is observed for MEDLINE (23%) and CISI (19.5%) at all levels of recall. For the TIME and FBIS collections, however, the average precision obtained by LSI after level search is significantly higher (11% higher for TIME, 60% higher for FBIS). The precision-recall graph for TIME (Figure 3-3) illustrates that the precision increase is consistent at all recall levels. For the FBIS collection (Figure 3-4), the precision improves more at lower recall levels than at higher levels, which suggests LSI with level search is able to retrieve more relevant documents earlier on. The fact that level search sometimes improves LSI precision might suggest that level search can filter out poorly relevant documents from the term-by-document matrix relative to a specified query. Certainly this phenomenon is collection specific as results has been obtained are different for the other two data sets. From the collection parameters listed in Table 2-1 (Section 2.2), only the document size, that is, the

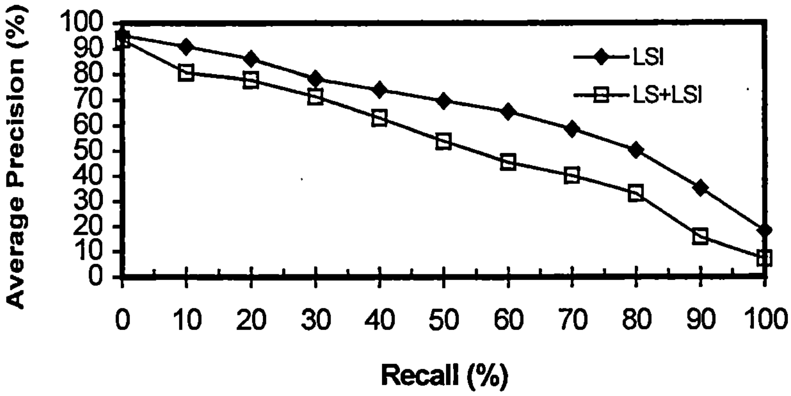

Figure 3-1. Precision-recall graph for LSI with/without level search for MEDLINE.

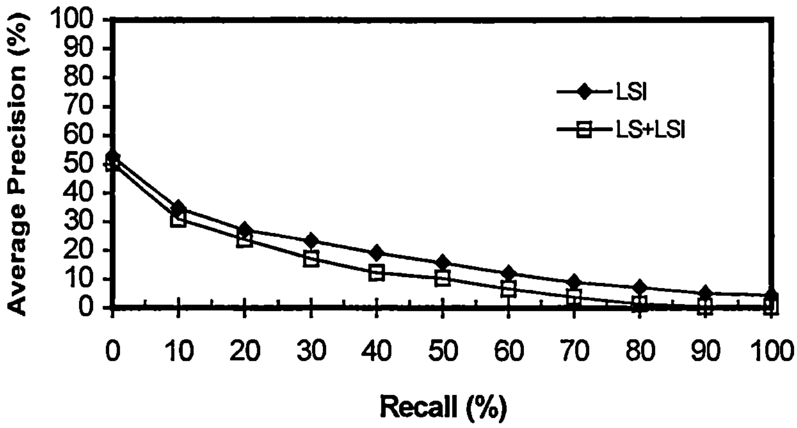

Figure 3-2. Precision-recall graph for LSI with/without level search for CISI.

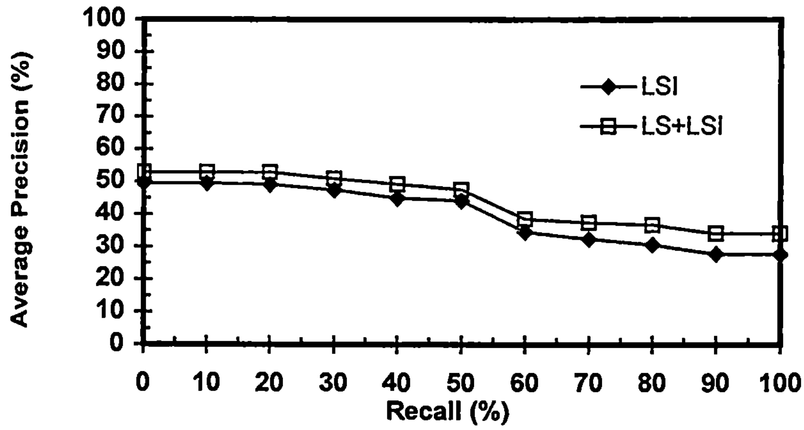

Figure 3-3. Precision-recall graph obtained by LSI with/without level search for TIME.

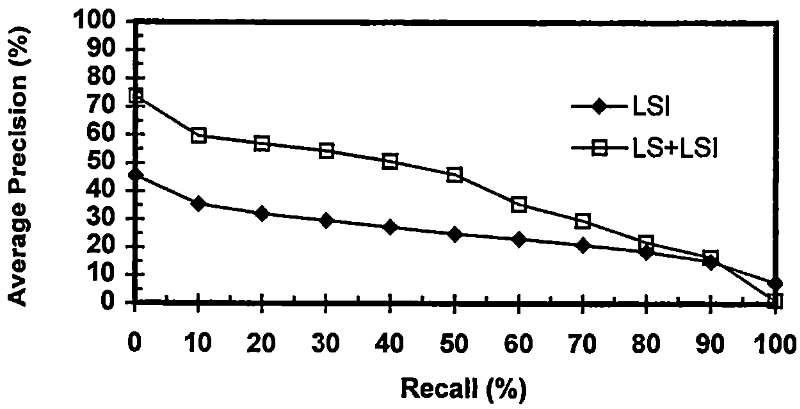

Figure 3-4. Precision-recall graph obtained by LSI with/without level search for FBIS.

Table 3-2: The average precision obtained by LSI with/without level search for MEDLINE, CISI, TIME and FBIS.

Data	MEDLINE		CISI		TIME		FBIS	
Recall (%)	**LSI**	Level search + LSI	**LSI**	Level search + LSI	**LSI**	Level search + LSI	**LSI**	Level search + LSI
0	95.5079	93.6611	52.6793	50.3852	49.5531	52.8894	45.5038	73.6172
10	90.9762	80.6942	34.7217	30.9908	49.5531	52.8894	35.4190	59.6553
20	86.1462	77.7925	27.1187	23.8619	49.0712	52.8894	32.0679	56.9082
30	78.2460	71.3966	23.3028	17.0897	47.4355	50.8872	29.6407	54.4102
40	74.1397	62.9686	19.1486	12.3355	44.9857	49.1979	27.2352	50.7824
50	69.7864	53.9041	15.7981	10.3691	44.1491	47.4862	24.9107	46.0793
60	65.4516	45.5724	12.0215	6.6405	34.4855	38.4783	23.2034	35.6658
70	58.6508	40.1698	8.8735	3.7699	32.3923	37.3873	20.9865	29.6718
80	50.1949	33.0229	7.1635	1.3098	30.6504	36.6801	18.6286	21.9396
90	35.0543	15.6465	5.0341	0.5169	27.7948	33.9889	15.1053	16.5892
100	18.2002	7.0652	4.3468	0.3252	27.6572	33.9811	7.5532	1.1020
Mean	**65.6800**	**52.9000**	**24.3300**	**19.7700**	**39.7900**	**44.2500**	**25.4600**	**40.5800**

average number of terms per document, shows possible correlation to such retrieval precision. Table 3-3 lists the average document size and the precision improvement for each data set. The MEDLINE and CISI collections have relatively low document size and LSI precision does not improve. The FBIS and TIME collections contain relatively large documents (about 4 times larger) and their precision by LSI improves by 12% and 60%, respectively. Empirically, this suggests that the document size might be one indicator of potential LSI precision improvement due to level search. Larger documents tend to have a higher percent of redundant terms which can be filtered out by level search.

The submatrix generated by level search is specific to each query for a data set. Since LSI will be applied to a much smaller submatrix as opposed to the larger term-by-document matrix, it is not clear if direct manipulation of the submatrix would help with either further reducing the submatrix size or improving precision. Therefore, a *pruning* technique has been applied to each query-specific submatrix for subsequent LSI modeling. The average precision measurement is taken and the results are presented in the following sections.

Table 3-3: The average document size and precision improvement for each document collection.

Collection	MEDLINE	CISI	TIME	FBIS
Mean	50.35	46.74	196.71	316.31
Median	47.00	45.00	158.00	187.50
Min	6.00	6.00	30.00	30.00
Max	181.00	171.00	1356.00	5243.00
Precision Improvement* (%)	-19.50	-23.00	12.00	60.00

* Precision improvement as defined in Section 2.2.

3.2 Level Search with Pruning

Pruning refers to the technique of selectively deleting some edges of a graph. In this context, pruning deletes terms associated with only one document in the submatrix obtained by level search. Query terms are not susceptible to such deletion since they are considered important to the original information need. It is important to note that pruning does not affect the documents in the submatrix. Therefore, level search with pruning should produce the same recall but with a reduced size of submatrix for each query as compared to level search alone. LSI can then be applied to each query-specific submatrix after pruning and the average precision can be calculated and compared.

By design, pruning should reduce the size of each query-specific submatrix by deletion of poorly connected terms. Table 3-4 illustrates the submatrix size obtained from level search with and without pruning. It indicates that level search filtering with pruning further reduces the number of submatrix non-zero entries by 20% for most of the collections. It eliminates the poorly connected terms from the submatrix obtained by level search. The expectation here is to render level search as a more cost-effective filtering technique for LSI without significant precision loss. Table 3-5 presents the average precision obtained by LSI using

Table 3-4: The average submatrix size obtained by level search (LS) with/without pruning (P) for MEDLINE, CISI, TIME, and FBIS.

Submatrix Size (%)	MEDLINE		CISI		TIME		FBIS	
Symbol	LS+P	LS	LS+P	LS	LS+P	LS	LS+P	LS
# of Docs (% of original)	256 24.78	256 24.78	313 21.44	313 21.44	64 15.06	64 15.06	1319 28.52	1319 28.52
# of Terms (% of original)	1419 24.34	3686 63.21	1497 36.69	3595 64.09	2131 19.72	6683 61.86	23381 55.01	37700 88.70
# of non-zeros (% of original)	11377 21.88	14472 27.83	14334 21.00	17166 25.15	12963 15.51	19006 22.74	809754 51.47	832678 52.92

Table 3-5: The average precision obtained by LSI with level search and optional pruning.

Data	MEDLINE		CISI		TIME		FBIS	
Recall	Level search + pruning + LSI	Level search + LSI	Level search + pruning + LSI	Level search + LSI	Level search + pruning + LSI	Level search + LSI	Level search + pruning + LSI	Level search + LSI
0	93.2367	93.6611	51.1761	50.3852	50.6716	52.8894	51.8923	73.6172
10	78.8616	80.6942	32.0951	30.9908	50.6716	52.8894	39.8812	59.6553
20	74.0408	77.7925	24.4439	23.8619	50.4974	52.8894	37.0992	56.9082
30	68.0738	71.3966	17.4823	17.0897	48.5585	50.8872	34.6361	54.4102
40	60.6240	62.9686	12.7622	12.3355	47.3386	49.1979	31.2246	50.7824
50	53.8128	53.9041	10.8110	10.3691	46.7259	47.4862	28.0838	46.0793
60	42.9452	45.5724	6.9562	6.6405	37.8213	38.4783	20.9394	35.6658
70	36.8453	40.1698	3.7395	3.7699	36.7248	37.3873	17.5826	29.6718
80	28.1039	33.0229	1.6112	1.3098	35.7319	36.6801	12.7787	21.9396
90	16.4562	15.6465	0.8944	0.5169	33.3596	33.9889	8.4002	16.5892
100	6.4178	7.0652	0.5777	0.3252	33.2605	33.9811	1.4476	1.1020
Mean	**50.86**	**52.9**	**14.78**	**19.77**	**42.85**	**44.25**	**25.82**	**40.58**

the pruned submatrix on input. Based on Table 3-5, a slight precision loss (less than 5%) is observed for the MEDLINE, CISI and TIME. However, for FBIS there is nearly a 40% average precision loss due to submatrix pruning. Also, pruning only reduces the number of submatrix non-zeros for FBIS by 3% (far less than 20% obtained by other collections). The reason why pruning affects the FBIS collection so differently may be related to the heavy usage of common terms. As previous mentioned, pruning only reduces the terms connected to one document in the submatrix. Obviously, the FBIS collection has fewer terms in this category since pruning only reduces the number of terms in the submatrix from 88% to 55%. In summary, pruning further reduces the submatrix non-zero entries by approximately 20% with an approximate precision loss of 17% across all collections.

3.3 A Case Study

LATIMES [25] is a large and heterogeneous collection of newspaper articles from the Los Angeles Times. The specific subcollection of articles used in this study was obtained by applying a relevance feedback technique described in Section 2.2. Table 3-6 lists some of the characteristics of this collection. LATIMES has an average of 230 terms per document and an average of 29 terms per query, which is larger than the average of MEDLINE, CISI and TIME, yet smaller than those of the FBIS collection.

The same level search filtering and pruning experiments described above were conducted on LATIMES. The average ratio of non-zero entries obtained by level search at level 2 is 36.08%. Pruning further reduces it to 31.8% (Table 3-7). Both level search with/without pruning produces the same average recall at 85.24% across all queries. Table 3-8 presents the average precision obtained by LSI with/without level search filtering and optional pruning. It is observed that the LSI precision increases slightly after pruning.

Table 3-6: Characteristics of the LATIMES collection.

Number of Documents	1086
Number of Terms	17903
Number of non-zeros (matrix)	250241
Number of Queries	48
Avg. No of Documents/Term	13.97
Avg. No of Terms/Documents	230.42
Density (%)	1.28
Avg. No of Terms/Query	28.85
Average Weighting/Term	0.52

Table 3-7: The average submatrix size for level search with optional pruning for LATIMES.

Method	# of Rows	# of Columns	# of Non-zeros
Level search (% of original)	227 20.90	14734 92.30	90280 36.08
Level search with pruning (% of original)	227 20.90	7257 40.54	79573 31.80

Table 3-8: The average precision (%) obtained by LSI with/without level search filtering or pruning for LATIMES.

Recall (%)	LSI	Level search + LSI	Level search + pruning + LSI
0	78.1077	76.1847	78.6740
10	75.2234	74.2576	77.8439
20	69.1141	68.4560	72.1236
30	66.1056	63.8817	66.8088
40	58.3200	60.0478	61.6757
50	56.687	55.7065	58.6169
60	50.8539	48.5353	51.7005
70	46.9282	44.1532	47.2156
80	42.5049	38.6960	41.1634
90	34.9114	28.6483	31.8591
100	29.5024	22.2882	25.1615
Mean	**55.2500**	**52.8100**	**55.7100**

Chapter 4

Conclusions

Figure 4-1 presents the average ratio of non-zero entries after level search filtering or pruning for all the collections including LATIMES. Figure 4-2 illustrates the average precision obtained by LSI with/without level search or pruning accordingly. Based on these two figures, level search can reduce the average number of non-zero entries of the term-by-document matrices (for LSI processing) by almost 65%. At the same time, it's capable of achieving an average recall near 80% for selected collections. Subsequent LSI based query matching can produce an average precision of over 80% of traditional LSI. For some collections, level search filtering can improve the precision somewhere between 10% and 60%. Pruning further reduces the ratio of non-zero entries by 20% with a slight precision loss of 17% across collections. In summary, level search with optional pruning provides a cost-effective filter for LSI in scalable information retrieval.

In Chapter 2, it was observed that for some collections level search improves the precision performance of LSI. Although the average document size might be one of the properties triggering such precision improvement, more quantitative testing is needed to further predict the correlation. Finally, the pruning technique deletes terms connected to one document in the submatrix. Actually this criterion could be expanded to 2, 3, 4, ... documents. In Chapter 1, term local frequency ($L(i, j)$ in Equation 1-1) has been defined as the number of times a term appears in

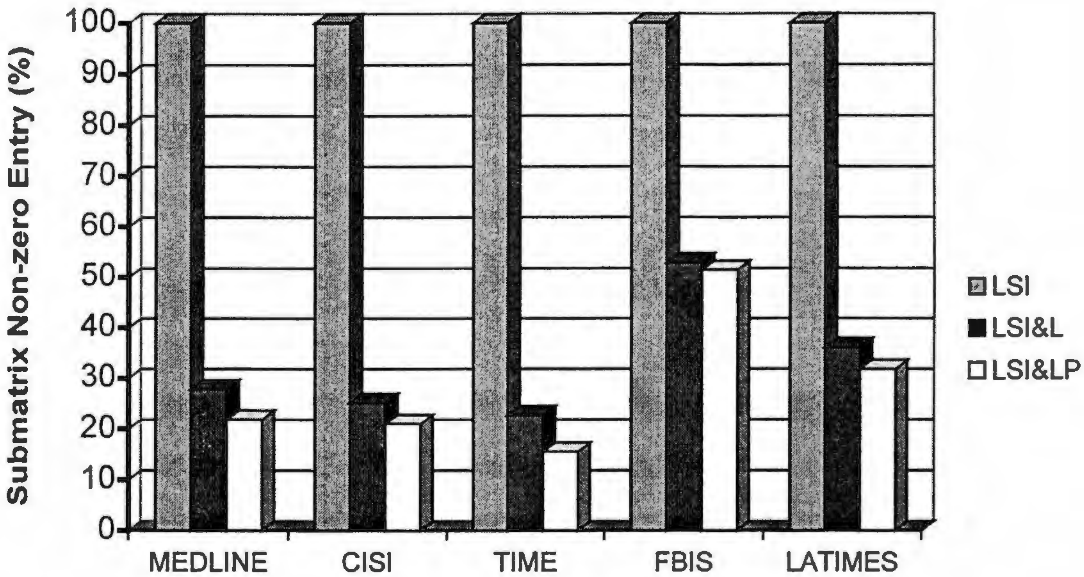

Figure 4-1: LSI input matrix non-zero entry comparisons after level search filtering (L) and pruning (P).

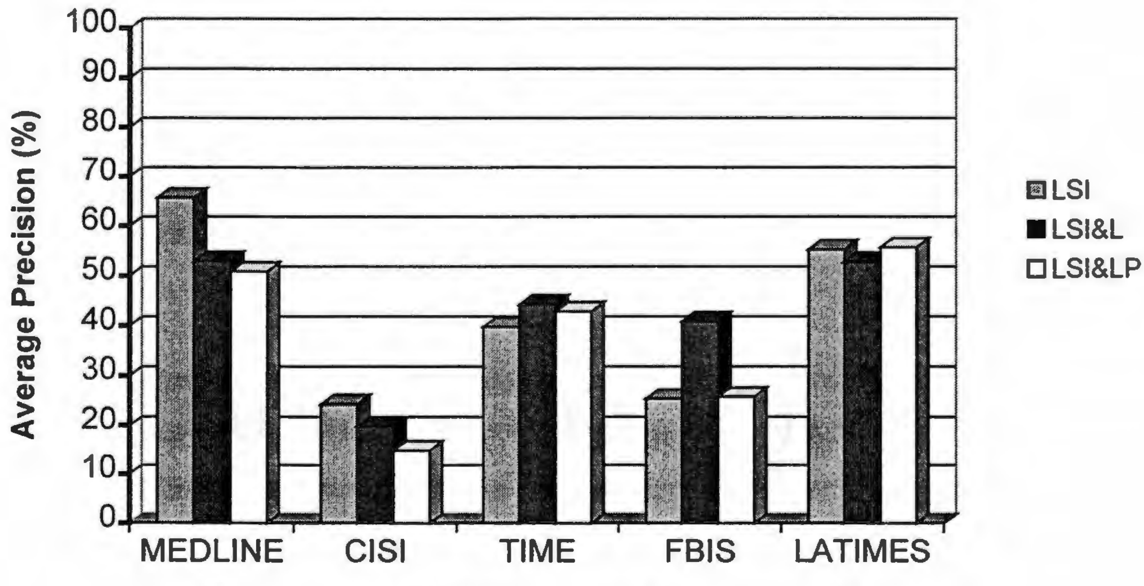

Figure 4-2: LSI average precision comparisons with/without level search (L) and/or pruning (P).

distinct documents. Here, considering each term has a global weighting indicating its importance to the indexing, such weighting could be combined with the local frequency as a new criterion for pruning. This approach could produce significantly different results by further selectively deleting poorly connected terms, especially for the FBIS collection.

Bibliography

Bibliography

[1] M. Berry and S. Dumais, *Using Linear Algebra for Intelligent Information Retrieval*, SIAM Review, 37 (1995), pp. 573-595.

[2] M. Berry, Z. Drmac, and E. Jessup, *Matrices vector spaces, and information retrieval*, SIAM Review, 41 (1999), pp. 335-362.

[3] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, And R. Harshman, *Indexing by Latent Semantic Analysis*, Journal of the American Society for Information Science, 41(1990), pp.391-407.

[4] M. Berry, S. Dumais and T. Letsche, *Computational Methods for Intelligent Information Access*, Proceedings of Supercomputing'95, San Diego, CA, December 1995.

[5] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1993.

[6] M. Berry and M. Browne, *Understanding Search Engines: Mathematical Modeling and Text Retrieval;* SIAM Book Series: Software, Environments, and Tools, 1999.

[7] G. Golub and C. Van Loan, *Matrix Computations*, third ed., Johns Hopkins University Press, Baltimore, 1996.

[8] S. Dumais, *Improving the Retrieval of Information from External Sources*, Behavior Research Methods, Instruments, Computers, 23(1991), pp.229-236.

[9] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van Der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Interactive Methods*, SIAM, Philadelphia, 1994.

[10] M. Berry, *Large Scale Singular Value Computations*, International Journal of Supercomputer Applications, 6(1992), pp. 13-49.

[11] M. Berry, T. Do, G. O'Brien, V. Krishna, and S. Varadhan, *SVDPACKC: Version 1.0 User's Guide,* Tech. Report CS-93-194, University of Tennessee, Knoxville, TN, October 1993.

[12] M. Berry, B. Hendrickson, and P. Raghavan, *Sparse Matrix Reordering Schemes for Browsing Hypertext, Lectures in Applied Mathematics*, Volume 32, 1996, pp.115.

[13] R. Korfhage, *Information Storage and Retrieval*, Wiley Computer Publishing, 1997.

[14] M. Morita, and S. Yoichi, *Information Filtering based on User Behavior Analysis and Best Match Text Retrieval.* In Proceedings of the 17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, Dublin, PP. 272-281.

[15] M. Persin, *Documentation Filtering for Fast Ranking*, In Proceedings of the 17th Annual International ACM/SIGR Conference on Research and Development in Information Retrieval, Dublin, pp. 339-348.

[16] S. Dumais, *What you see is what you want: Combining evidence for effective information filtering*. Post abstract in Proceedings of the 18th Annual International ACM/SIGR Conference on Research and Development in Information Retrieval, Seattle, Washington, pp. 369, 1995.

[17] N. Belkin and W. Croft, *Information Filtering and Information Retrieval: Two Sides of the Same Coin?*, Communications ACM, 35, pp. 29-38, 1992.

[18] P. Foltz, *Using Latent Semantic Indexing for Information Filtering*, in Proc. ACM Conference on Office Information Systems (COIS), pp. 40-47, 1990.

[19] P. Foltz and S. Dumais, *Personalized Information Delivery: An Analysis of Information Filtering Methods*, Communications ACM, 35, pp. 51-60, 1992.

[20] M. Hearst and C. Plaunt, *Bringing Natural Language Information Retrieval out of the Closet*. SIGGHI Bulletin 22, No. 1: 42-48, 1993.

[21] G. Salton and J. Allen, *Text Retrieval Using the Vector Processing Model*. Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, Nevada, pp. 9-22, 1994

[22] A. Aho, J. Hopcroft, J. Ullman, Data Structure and Algorithms, Addison-Wesley Publishing Company, 1987.

[23] T. Kolda, *Limited-Memory Matrix Methods with Applications*, Ph.D Dissertation, Applied Mathematics Program, University of Maryland at College Park, pp.58, 1995.

[24] D. Harman, Ed., *Proceedings of the Third Text Retrieval Conference* (*TREC-3*), Gaithersburg, MD, 1995, Department of Commerce, National Institute of Standards and Technology. NIST Special Publication 500-225, 1995.

[25] E. M. Voorhees and D. Harman, Eds., *Proceeding of the Fifth Text Retrieval Conference (TREC-5)*, Gaithersburg, MD, 1996, Department of Commerce, National Institute of Standards and Technology, NIST Special Publication 500-238, 1996.

Appendices

Appendix

A. Test Procedures

The testing procedure used in this research consists of 3 major steps: key/matrix generation, query vector generation and subsequent level search or LSI implementation.

Key/Matrix Generation

Each test suite consists of a collection of documents (*document file*) to be searched on, a set of queries (*query file*), and a list of answers of relevant documents to each query (*answer file*). The first step in level search and LSI is document collection parsing and generation of the term-by-document matrix. As Figure A-1 shows, the parsing program takes the document file as input, deletes the common words defined in the stoplist (Section 1.1), generates the term-by-document matrix in Harwell-Boeing format and writes the keywords to a *term list* file. Weightings can be applied to each element of the term-by-document matrix. The matrix and the term list will be used as input for query vector generation.

Query Vector Generation

Natural language queries need to be transformed to vector representations. A program will take the query file as input and match each word to the terms in the term list. The output is a separate query vector file containing a list of indexing terms with assigned term numbers and proper term weightings.

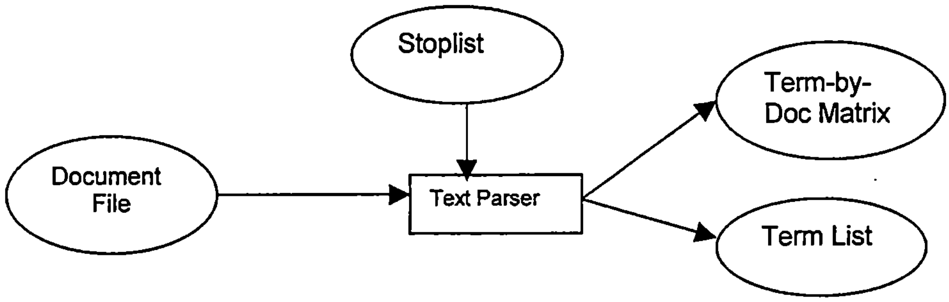

Figure A-1: The process flowchart for key/matrix generation during level search and LSI.

Level Search Driver Program

The level search code (written in C) takes the term-by-document matrix in Harwell-Boeing format and each query vector as input and constructs the level graph. The output consists of query-specific submatrices in MATLAB format.

LSI Driver Program

The LSI driver program (written in MATLAB 5.1) will take exactly the same input file as the level search driver. However, it only reports the average precision (see Section 2.2) across all queries to each text collection. The number of factors *k* (Section 1.1) used for all LSI testing is set at 100.

B. Testing Parameters for Level Search Filtering

Collection	Best Weighting	Best Threshold	Average Recall (%)	% of Non-zeros of original
MEDLINE	None	None	85.74	27.83
CISI	Weighting in query vector	Mean	69.42	22.74
TIME	Weighting in query vector	Mean	55.11	25.16
FBIS	Weighting in query vector	Mean	82.05	52.92
LATIMES	Weighting in query vector	Mean	85.24	36.08

Vita

Xiaoyan 'Kathy' Zhang was born in NanTong, China in 1973. She received her Bachelors of Science degree in Engineering from East China University of Science and Technology in 1995. After graduation she worked for Carrier Corporation as Environmental, Health & Safety Data Analyst for one year. She continued her study In Environmental Engineering and received her Master of Science degree from the University of Tennessee, Knoxville in August 1998. She earned her second Master of Science degree in Computer Science in December 1999. She also worked as an intern in Information Retrieval research group at Telcordia Technologies (formerly Bellcore) in the summer of 1999.