MATRIX COMPUTATIONS FOR QUERY EXPANSION IN INFORMATION RETRIEVAL

by

Fan Jiang

Department of Computer Science
Duke University

Date: ______________________

Approved:

____
Michael L. Littman, Supervisor

____
Alan W. Biermann

____
Xiaobai Sun

____
Gregory B. Newby

____

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Computer Science
in the Graduate School of
Duke University

September 2000

ABSTRACT

(Computer Science)

MATRIX COMPUTATIONS FOR QUERY EXPANSION IN INFORMATION RETRIEVAL

by

Fan Jiang

Department of Computer Science
Duke University

Date: ______________________________
Approved:

____
Michael L. Littman, Supervisor

____
Alan W. Biermann

____
Xiaobai Sun

____
Gregory B. Newby

____

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science in the Graduate School of Duke University

September 2000

Abstract

Information retrieval (IR) is the task of identifying information items (documents) that are relevant to a user query from a collection. The most popular IR research technique, the vector space model (VSM), is a word-matching approach: it uses the words in common between the query and the document as a basic way of determining their similarity. Because the same concept can be expressed in the query and document using very different vocabularies, synonymy can cause a document to be judged as irrelevant by VSM.

Query-expansion methods deal with this problem by automatically supplying the query with additional words that are related to those already in it. Latent semantic indexing (LSI) is a statistical method that derives term associations through a reduced dimensional singular value decomposition (SVD) of a matrix formed from the collection. LSI has equaled or outperformed VSM on many relatively small retrieval collections. But, with the growing size of modern information repositories, LSI has failed to demonstrate its advantage over traditional word-matching methods on some of these large corpora.

In this work, I provide evidence that LSI is not reaching its potential for large collections because existing SVD implementations are not able to compute a sufficiently large number of dimensions. I establish a unified framework of vector-based information retrieval called *dimension equalization.* Through this, I present *approximate dimension equalization* (ADE), a method that "extrapolates" the result of a high-dimensional SVD based on a relatively small number of computed dimensions. Experiments indicate that ADE improves retrieval performance over LSI and has great utility in cross-language applications.

I also investigate sampling approaches to reducing LSI computation, which use only a subset of the document collection to build LSI term-associations. My focus is on *local LSI*, a variation of the ever-popular local feedback approaches in the IR community. This method computes an SVD on a subset of documents that are related to the query. Experiments show that local LSI outperforms not only the global sampling methods but also the baseline VSM or LSI without sampling. I extend the existing local LSI approach to cross-language retrieval and present its high-quality results.

Acknowledgements

I could not ask for better support than what I have received from my advisor, Michael Littman, throughout the work of this thesis. Without his early guidance and encouragement, it would have taken me at least an extra year or two to find the right track to completing my degree work. Michael is also one of the easiest people I have ever worked with—I often felt as if he were a colleague rather than an advisor. Yet his casualness serves to be a constant reminder that doing research can be a lot of fun, which is one of the main reasons that I am where I am today. I do look forward to continuing having him as a "colleague."

I am very grateful to Xiaobai Sun, for inspiring me to the understanding and creation of the unified view of vector-based information retrieval methods. Her insightful comments from so far as the beginning of this thesis work have been an indispensable factor of my being able to complete this work.

Many thanks to Alan Biermann and Gregory Newby for serving on my committee. In particular, Professor Biermann has been giving me the type of encouragement and motivation that I can really feel in the heart. The many comments and suggestions from Professor Newby have been invaluable, and his work on the information space method has been a source of inspiration for me.

I would also like to thank Michael Berry for providing me many insights on the computation and application of SVD. Doug Oard has also helped me in a number of ways—I thank him especially for his comments and suggestions on how to prepare for the completion of a graduate career and for the start of a professional career.

I thank my parents, Dong Jiang and Songhe Liu, for helping me in many aspects of my daily life throughout my graduate years, so that I can be more focused on my thesis research than most other people on theirs. I am indebted to them for giving me loving and caring, not only during the work of this thesis, but also all through my lifetime.

And finally, I thank Grace, my fiancée, for her love, support, and patience. Our forthcoming marriage has been an extra motivation for me to complete my dissertation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the advent of the World Wide Web, there is virtually an unlimited amount of information that one can access on-line. What follows is the growing need for automated, large-scale information retrieval (IR) systems. Now, the task that used to fall solely on librarians has become the research and work of computer scientists. Our goal in this field is to develop methods and techniques for effective retrieval of information from large-scale text repositories.

1.1 Information Retrieval

Information retrieval (IR) is generally defined as the task of locating useful information items that are relevant to a user query from a free text corpus. The text collection can contain any type of mostly unstructured text, such as academic papers, bibliographic records, newspaper articles, or paragraphs in a manual. We usually call each information item, or the minimal retrievable element from a collection, a *document*, and hence "*document retrieval*" is often used in place of "information retrieval."

Research in information retrieval is concerned with developing models to represent contents of text documents and algorithms to retrieve the relevant items to a user's request for information. Even with the tremendous advance of computing technology and artificial intelligence in recent years, it is still impossible for a computer system to truly understand the ideas expressed in documents. Thus, most of the current retrieval systems employ some type of text or concept matching, with degree of relevance being determined by the quantity and/or quality of the matches.

There are two crucial aspects that people look for in their search engines: effectiveness and efficiency. Effectiveness refers to how well a system meets the users' information needs: whether they find what they are looking for in the first document(s) the systems returns. On the other hand, efficiency is about speed—how long a user has to wait for a return from the retrieval system may be as important a factor as effectiveness in determining the user's satisfaction with the systems. These two qualities often are at odds against each other: by doing some extra computing work (meaning the user waits longer), more relevant documents can be placed at the top of the return list. This dissertation is mostly concerned with the effectiveness problem, although the efficiency issue can not be ignored.

In recent years, research in information retrieval has been expanded to include tasks such as information filtering (sorting through large volumes of dynamically generated information and presenting to the user those that are likely to satisfy his or her information requirement) and text categorization (assigning keywords or categories to a text for easy organization and retrieval) [67]. In a sense, information retrieval is a special case of text categorization—we try to classify a collection of documents into two categories: the ones that are relevant to a user query, and the others that are not. As a result, phrases such as "document retrieval" and "*text retrieval*" have been used in place of "information retrieval" to distinguish them from the more general meaning of IR. In this dissertation, however, I still use "information retrieval" by its original definition: finding documents relevant to a user query.

When both the corpus and user queries are in a single language, this is known as *monolingual* information retrieval (MLIR). The counterpart of MLIR is CLIR, or *cross-language* information retrieval, where the queries are stated in one language and the text records are in another. There is no strict definition of multilingual information retrieval: sometimes it means information retrieval in a language other than English [54, 101]; but, it often means searching of texts that can be in any of a set of languages from a document repository, where it is possible that individual documents are in more than one language [62].

Cross-language information retrieval is an important problem in our increasingly global information economy. An automated CLIR system will benefit more than just people who speak two or more languages. People who read a second language may not be confident in their ability to express ideas in that language. They will likely find a cross-language text retrieval system particularly useful when they are looking for documents in that language. A CLIR system can also be helpful if the user is able to read only a single language. For example, when only a small portion of the document collection will ever be examined by the user, performing retrieval before translation (through a CLIR system) can be significantly more economical than performing translation before retrieval.

1.2 Present Research

One of the classical models of information retrieval is the vector space model (VSM), developed by Salton and his students at Cornell University in late 1960's and early 1970's [100, 94]. In this model, any given document is represented by a vector in a high-dimensional semantic space, with each dimension or direction of the space representing a particular indexing term in the collection. The proximity between two vectors in that space becomes a measurement of the semantic closeness of the corresponding two texts. The set of document vectors from a collection naturally forms a document matrix.

A fundamental weakness of VSM is its index-by-term scheme, under which no two documents will be recognized as relevant to each other unless they share some words in common. For example, imagine one does a keyword search for books on "computer graphics." A book titled "digital imaging" will certainly be missed by a retrieval system that only implements VSM. The problem here is that, in many cases, there are multiple ways to express a single concept, but VSM does not take this into account. This weakness of VSM also makes it useless in the area of cross-language retrieval, as words expressing the same idea are most often spelled differently across languages.

Techniques that address this problem use a form of thesaurus for *query expansion*, or adding words related to those in the query to the query, in hope of finding additional relevant documents that do not use the words in the query. The related words can come from an external source such as a standard published thesaurus, or can be identified directly from statistical analysis of the text corpus.

Latent semantic indexing (LSI) [32] is one such type of query-expansion method. It applies a linear algebra computation method called *singular value decomposition* (SVD) [49] on a matrix formed from the corpus of interest. Semantic associations among terms are found through this one-step analysis of their statistical usage in the collection, and they are implicitly stored in the *singular vectors* computed by SVD. A query represented in the latent semantic space can retrieve a relevant document even if they have no words in common. Given suitable training data, LSI can easily be extended for cross-language retrieval because building term associations across languages are just like creating translations.

Experimentally, LSI has been shown to perform better than both VSM on many, but relatively small corpora [39, 32, 36, 70, 28]. It also has achieved favorable results in cross-language information retrieval [65, 7, 26], where the conventional VSM cannot be directly applied without using machine translation techniques on queries or documents [92, 62, 42]. However, in recent experiments where the collection sizes exceed a hundred thousand, LSI has not performed well compared to other methods. The reason here is that the quality of term associations created by LSI depends on the number of singular vectors that can be computed via SVD, which in turn inversely depends on the size of the text collection.

In this dissertation, I investigate several techniques to deal with the scalability problem of LSI. I begin my study with an examination of the fundamental connections between LSI and a number of other vector-based IR methods. The similarity and differences among these methods, with observations of their behavior when applied to real text corpora, lead to the formulation of the concept of *dimension equalization.* Through this, I propose an approximation algorithm to LSI named *approximate dimension equalization* and compare its retrieval performance with other vector-based methods on various test collections. Then, I consider sampling approaches to reducing LSI computation and concentrate on the query-based *local LSI* method. I propose the extension of local LSI to cross-language retrieval and demonstrate its effectiveness on several collections.

1.3 Contributions

The main contributions of this research are

- **Dimension Equalization**: The comparison of various vector-based methods of information retrieval produces a unified view of these techniques.

- **Approximate Dimension Equalization**: Observations on the distribution of singular values of document matrices inspire me to develop this approximation algorithm, which mimics the performance of LSI with fewer computed singular vectors. From a different point of view, it makes cross-language VSM possible.

- **Cross-language Local LSI**: The well-known effectiveness of local feedback is re-applied with LSI on the top documents from an initial retrieval. A step further still is to employ it to cross-language applications. Different approaches are used and state of the art results are obtained.

Some other contributions I made during the work of this dissertation include

- The incorporation of current best indexing schemes into the LSI implementation module.

- The development and implementation of a new cross-language LSI retrieval formula.

- The implementation and evaluation of the generalized singular value decomposition approach to cross-language information retrieval.

- The implementation and evaluation of a length-based probabilistic sampling approach to SVD approximation.

1.4 Organization

The rest of this dissertation is organized as follows:

- Chapter 2 reviews the relevant methods and models of information retrieval, with a focus on the vector-based approaches.

- Chapter 3 shows the mathematics behind various vector-based IR methods including VSM, the generalized vector space model (GVSM), and LSI. I show how they are related to one another through the idea of dimension equalization.

- Chapter 4 introduces the approximate dimension equalization method and shows its retrieval performance on a number of test collections, with comparison to VSM, GVSM, and LSI.

- Chapter 5 begins with a discussion of the document sampling approach to SVD approximation. Then, I introduce the local LSI technique and propose its extension to cross-language applications. I demonstrate its utility on several retrieval collections later in the chapter.

- Chapter 6 concludes this work and points toward some future directions.

Chapter 2

Literature Review

In this chapter, I review the most popular models and algorithms developed in the field of information retrieval, with the emphasis on vector-based approaches. Chapter 3 gives an in-depth analysis of several of the vector-based methods that are pertinent to this dissertation. Before starting, I present brief background on how the effectiveness of an IR system is measured.

2.1 Evaluation of IR Systems

By definition, information retrieval is the process of matching some user query against text items from a collection and returning the most relevant ones to the user. In practice, a working IR system usually returns documents ranked according to their computed similarity to the query. To evaluate the effectiveness of an IR system, therefore, specific measure have been developed to determine how well a system ranks relevant documents ahead of non-relevant ones in the retrieved set. The "true" relevance of documents to queries is usually judged by domain experts.

Researchers have designed and built standard test collections of documents, along with user queries and associated relevance judgments. Different information retrieval systems can be directly compared on these identical collections. Typically, relevance assessments are binary (a document can only be relevant or irrelevant) and assumed to be exhaustive (documents that are not judged are irrelevant). The most straightforward way of building a test collection to evaluate IR systems is to let human experts determine for each query whether each document in the collection is relevant to it or not. This process is extremely time-consuming and is feasible only when the test collection is relatively small with a few thousand documents at most. Examples of such collections include the 1,400-document Cranfield collection with 225 queries (as mentioned in the overview of the first TREC [52]) and the CMU UNICEF multi-lingual (English and Spanish) collection of 1,121 documents and 30 queries [26, 121].

When the collection size gets large, a complete assessment of the relevance between every query-document pair becomes prohibitively expensive. An alternative way to build a test collection was proposed by Sparck Jones and van Rijsbergen (as mentioned in the overview of the first TREC [52]) in the case where there are many systems to be compared. Known as the *pooling method*, this approach lets human experts judge, for each query, only the top n documents returned by each retrieval system, thus limiting the number of relevance judgments that need to be made. The pooling method has been used in the Text REtrieval Conference to build some very large test collections [52, 53, 56].

Another approach to constructing a large test collection for IR evaluation has been introduced by Sheridan et al. [105], in which the collection must contain time-sensitive documents, like news articles. This approach relies on query topics to be related to unexpected events, and so all documents dated prior to the event of the query topic were automatically assumed to be irrelevant and discarded without being judged by human experts. For those documents dated on or after the event of a given query, the relevance judgments are made up to the last documents on the fourth day following the event. All documents dated thereafter are not included in the test collection of that query, which results in that each query has a different set of documents to be tested on. The advantage of this

	relevant	irrelevant	
retrieved	a	b	$a+b$
unretrieved	c	d	$c+d$
	$a+c$	$b+d$	$a+b+c+d$

Table 2.1: The relevance-retrieval contingency table: a, b, c, and d represent frequencies of occurrence of the four conjunctions.

method is further narrowing the number of relevance judgments made by human experts, while its disadvantage is its obvious limited applicability.

Since an ideal test collection is very hard to come by, the evaluation of systems has always been an issue in information retrieval research.

The retrieval results of an IR systems on a test collection with a set of test queries can be compared against the relevance judgments made by human experts. The most common measures of retrieval effectiveness consist of two quantities, namely *recall* and *precision.* Recall is defined to be the proportion of relevant documents retrieved by a system, and precision the proportion of the retrieved documents that are relevant. These two ratios are better understood by looking at Table 2.1 [112]. From the table, we see that the *recall ratio* is defined as

$$\frac{a}{a+c} = \Pr(\,\text{retrieved} \mid \text{relevant}\,), \tag{2.1}$$

or an estimate of the conditional probability that a document will be retrieved given that it is relevant, while the *precision ratio* is

$$\frac{a}{a+b} = \Pr(\,\text{relevant} \mid \text{retrieved}\,), \tag{2.2}$$

or an estimate of the conditional probability that a document will be relevant given that it is retrieved.

Typically, the two quantities are in conflict with each other: recall score can

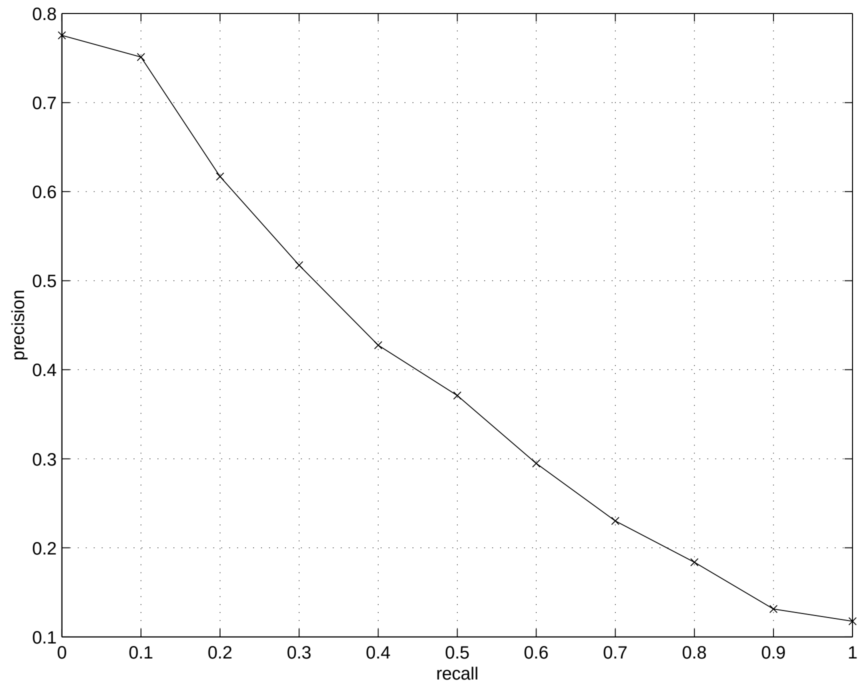

Figure 2.1: The interpolated recall-precision graph: precision at each of the 11 standard recall points are the highest achievable beyond that point.

be enhanced by returning more documents, but it is likely that more irrelevant documents will be added to the retrieval set, and hence reduce precision score. The opposite is also true: when precision goes up, the recall ratio tends to go down. Given the relevance judgment for a query against a set of documents, the precision scores for an IR system's retrieval runs can be computed at various levels of recall. The commonly used levels of recall are from 0% to 100% in increments of 10%, where the precision at a recall level is *interpolated*: we take the highest achievable precision score at or beyond this level of recall. We can then visualize the effectiveness of an IR system by plotting precision against recall at these levels. Figure 2.1 shows a typical precision-recall graph from a retrieval run by some IR system. Note the downward slope of the plot, which confirms the inverse relationship between these two quantities.

Information retrieval researchers have developed evaluation measures that combine precision and recall and also take into account document ranking. The precision numbers at the aforementioned 11 recall levels are averaged to get the *11-point average precision* value. If there is more than one query, the precision at each recall level is first averaged over all queries and then averaged over the 11 points. This way, there is a single score that summarizes the overall performance on a test collection.

In addition to the 11-point average precision, the *non-interpolated average precision* is also commonly used as an indicator of retrieval effectiveness: we compute precision at each point of the ranked return list where a relevant document is found, and then we take the average over all those precision values. See Chapter 15 of the book by Manning and Schütze [73] for an illuminative example of these measures of accuracy in information retrieval. The non-interpolated average precision was first introduced in the annual Text REtrieval Conference (TREC) (http://trec.nist.gov/) and is often called simply "average precision" by many IR researchers. TREC participants also use 11 recall-level and 9 document-level precision averages to report the performance of their systems. The latter quantities are the averaged precisions (over a set of queries) computed after a specific number (5, 10, 15, 20, 30, 100, 200, 500, and 1000) of documents is retrieved (see, for example, appendix of the TREC-2 proceedings [57]).

A deeper view of the precision and recall ratios in terms of statistical decision theory is given by Swets [112, 113]. In those papers, Swets argues that instead of using precision, we should be using a new measure called "false drop":

$$\frac{b}{b+d} = \Pr(\,\text{retrieved} \mid \text{irrelevant}\,), \tag{2.3}$$

which is an estimate of the conditional probability that an item will be retrieved given that it is irrelevant. He suggests that recall and precision cover only a column and a row of Table 2.1 and do not serve to specify the remainder of it. But, precision and recall prevailed and became the standard measurements of IR systems.

2.2 Exact Term Matching Systems

In information retrieval, *terms* refer to the individual indexable items in a document or a query. Usually, a term is just a single token, or *unigram*, which is a word in languages like English, or a character in languages like Chinese. A term may also be the combination of two or more consecutive tokens, or a *multi-gram*, in a document. When documents are scanned for n-gram terms, where $n \geq 1$, the previous term overlaps with the current one by $n - 1$ tokens, unless there is a punctuation mark at the end of the previous term, in which case the current term starts at the first token after the punctuation mark. For example, if we use *bigrams* (two consecutive words) indexing, the line “read my lips, no new taxes” will consist of the following tokens: “read my,” “my lips,” “no new,” and “new taxes.”

Today, most commercial applications of information retrieval such as those used by Yahoo! (www.yahoo.com) and eBay (www.ebay.com) are based on exact pattern matching of terms of queries with terms in text. These applications require queries that range from simple Boolean expressions using a few “ANDs” and “ORs” between terms, to complex pattern matching expressions using proximity operators, nested expressions, etc. For example, I often use the query “@1 promo CD single” at eBay’s search page to look for auction items that contain “promo CD,” “CD single,” or “promo single” in their titles (@1 means to pick two of the three words specified). To support rapid retrieval of the documents that contain any particular query word, a system typically includes an inverted index that indicates for each word in the collection the list of documents that contain it [11, 94].

Several indexing practices are commonly used to improve efficiency and retrieval performance in Boolean matching systems as well as the other retrieval systems discussed later. One of these is phrase and proper name indexing, in which any sequence of words that possesses a special, unique meaning is identified and grouped together as a single lexical item for indexing. Sophisticated algorithms and heuristics have been developed in the field of natural language processing to identify phrases, but a simple model of selecting the most frequent bigrams of words often works well [73]. In languages such as Chinese and Japanese, there are no obvious word boundaries. A sentence in one of these languages consists of a sequence of non-spaced ideographic characters that are meaningful and represent words, but can also be grouped together into multi-character words with special meanings. Whereas bigrams or multi-grams are the simplest schemes that people have tried for text retrieval in such languages, the topic of indexing Chinese or Japanese texts has grown into stand-alone research areas in IR. For example, in 1997 the Text REtrieval Conference (TREC) initiated Chinese retrieval as one of its several "tracks"—tasks that focus on particular subproblems of text retrieval [109].

Another common feature of IR systems is the removal of *stop words*, which are grammatical or function words that bear little of the content of the documents they appear in [73]. For instance, the SMART retrieval system [93] uses a default list of 571 English words that are to be removed from the documents during indexing. Examples of these stops words are "after," "enough," "such," and "what." Stop word lists for other languages such as German and French are not as widely available as for English.

Finally, *stemming*, the procedure of converting different morphological variants of the same word to a single basic form, is often used in the preprocessing step of many IR systems. People hope to use stemming to reduce the number of unique *keywords* being indexed as well as to match words that have similar morphological forms and semantics. For example, the words "organize" and "organization" will have their respective suffixes removed by the stemmer to become the same word "organ." Lovins and Porter stemmers are the mostly widely used for English over the years [72, 81], while stemmers for other languages have been created in recent years to meet the growing interest in multi-lingual information retrieval. For instance, the PRISE indexing and search engines from the National Institutes of Standards and Technology (NIST) include a Spanish stemmer in addition to Porter's stemmer for English [33]. A problem with the suffix- or prefix-removing stemmers is that words of different meanings can be stemmed to the same token. For example, "secret" and "secretary" would become the same word "secret."

A basic weakness of Boolean query systems is that they require user expertise to avoid all-or-nothing results. A user of the system often has to change her original formulation of the query to obtain a better result, and doing this well requires experience. More sophisticated retrieval models and methods have been developed over the years for measuring similarities between a query and a document. Some of the more prominent ones include: the vector space model [100] and related vector-based methods such as latent semantic indexing [32], probabilistic retrieval models [59, 86, 29, 44], and inference networks [114]. In this study, I investigate only the vector-based methods of information retrieval coupled with matrix computation techniques.

2.3 Term-Weighting Vector Space Model

In the vector space model of information retrieval developed by Salton [100, 94, 99], the index terms are viewed as basic orthogonal dimensions in a vector space, and each document is represented by a vector in that space. The frequency of occurrence of a term in a document contributes to the component of the document along the corresponding orthogonal dimension for the term. The vectors of all documents in a collection can be collected together to form a *term-by-document* matrix, which typically is the end result of automatically indexing an entire corpus.

The relevance of two text bodies is expected to be directly proportional to the spatial proximity of their vectors in the vector space, which is usually computed by a vector inner product. Hence, in VSM, two documents are considered to be more related if their vector representations match in more dimensions, i.e., they share more vocabulary. The appeal of VSM lies in this simple yet elegant way of representing and comparing textual information.

Salton and his students implemented a well-known information retrieval system named "SMART" based on the vector space model [93]. SMART has had a huge impact on IR research for the past thirty years, and many people have used it as the *de facto* standard against which they measure their IR research results.

One of the keys to the success of the vector space model is its term weighting schemes, which refer to the techniques of assigning real-number values to term components in a document vector [97, 106]. In other words, the value of each vector component in a VSM document vector represents the weight of the corresponding term in that document. In the simplest case, we can use a binary indicator, assigning a value of 1 to a vector component if the corresponding index term appears in the document and 0 otherwise. Generally, this approach will not be very useful since the vector space model is simply reduced to the exact pattern matching method discussed in Section 2.2. Many researchers have examined the subject of term weighting over the years and their experiments show that appropriately weighted terms improve the effectiveness of a retrieval system. Traditional factors used in term weighting can be divided into two types: local and global. A third factor, *document length normalization*, is a more recent development used to compensate for length differences among articles in a text collection.

2.3.1 Term Local Weighting

Aside from the binary weighting I have just mentioned, most commonly-seen local weighting schemes incorporate the *term frequency* (*tf*) factor. As the name suggests, term frequency is the occurrence count of a term in a document, which is intended to capture the importance of that term in the document. However, it is very unlikely that a query term appearing four times in an article can be four times as salient as a query term that appears only once, or a document with four occurrences of one query term is two times more relevant than another document with single usages of two different query terms. Thence, the term-frequency factor is usually dampened by a function like square root or logarithm [97, 35, 19, 106]. For example, one of the local weighting functions used by SMART is $1 + \ln(tf)$, while another of the form $1 + \ln(1 + \ln(tf))$ is used in recent TREC experiments by researchers at AT&T [108]. Both ensure that a term has at least a local weight of 1 if it appears in a document, while any additional appearances only increase the value slightly. On the other hand, no appearance would result in a weight of 0—a huge difference compared to any nonzero weights.

The *augmented tf* weighting is another popular approach for computing the term local weighting and is implemented in various retrieval systems including SMART and INQUIRY [97, 25, 17], the latter of which can be obtained from the University of Massachusetts. Also known as the maximum term frequency normalization, the basic form of this weighting scheme is [17, 106]:

$$d_t + (1 - d_t) \cdot \frac{f(tf)}{f(\max tf)},$$

where d_t is a constant representing the minimum local weighting value given to a term occurring in a document, $f(tf)$ a function of the frequency count of that term, and $f(\max tf)$ the same function applied to the frequency of most frequent term in the document. The value of d_t is usually set somewhere around 0.5 and never exceeding 1.0, so that the overall *tf* factor is restricted in the range of d_t and 1.0. This way, like the aforementioned *tf* factor involving logarithmic functions, a single appearance of a query term gets a default weight while additional occurrences cannot change this value too dramatically.

The augmented *tf* weighting is based on the observation that long documents usually use the same terms repeatedly so that the *tf* factors tend to be large in these documents. As a result, long documents are more likely to have higher similarities to a query than short documents. Normalizing the local weighting by the maximum term frequency in the document reduces this effect. Singhal et al. [107] believed that the maximum *tf* normalization is not ideal because a term with the same occurrence frequency in two documents will have the same value with augmented *tf* weighting. Their importance in the two documents can be very different depending on the frequencies of other terms in those documents. Hence, they used average term frequency in a document as the divisor in their normalization equation for all terms in the document:

$$\frac{1 + \log(tf)}{1 + \log(\text{average } tf)}, \tag{2.4}$$

so that a term that appears the average number of times of all terms in a document would have unit importance in that document. They obtain better experimental results using the average *tf* normalization than the maximum *tf* normalization.

Another local weighting factor that has been gaining recognition in recent years is the *tf* factor used in the Okapi information retrieval system [88, 87]. This *tf* factor is based on approximations to the combination of a two-Poisson model of term frequency [12, 58, 59] and Robertson and Sparck Jones' term relevance weighting [86]. The weighting function has the form

$$\frac{tf^c}{K^c + tf^c}$$

where K is a constant that is usually 1 or 2 and c is either 1 or greater than 1 depending on the shape of the actual function the factor intends to mimic. Since the two-Poisson model in effect assumes that all documents are of equal length, a document length normalization factor is usually included in the formulation of K, which I will further discuss in Section 2.3.3.

2.3.2 Term Global Weighting

Since the main problem in IR is to select a few relevant documents from a collection that includes many non-relevant ones, a term's local weighting alone does not seem to provide sufficient discriminating power to produce good retrieval performance. For example, a query term with high frequency counts in many documents tends to have all such documents retrieved as relevant [97]. Thus, researchers have developed global weighting schemes to differentiate content-bearing terms from others [110, 95].

The inverse document frequency (*idf*) factor, originally derived by Sparck Jones [110], is the most widely used global weighting scheme today. This factor is expressed as an inverse function of the number of documents a term appears in:

$$\log\left(\frac{n}{df}\right),$$

where n is the total number of documents in the collection and df is the number of documents in which the term being weighted appears in. The *idf* factor basically gives less weights to terms that occur in many documents. In addition, a theoretical justification of this *idf* weighting scheme was also presented by Wong and Yao [116], who proposed a goodness measure of a term in a document based on Shannon information theory and showed that *idf* is an approximation of that measure.

Another global weighting function that has the similar effect as *idf* is the entropy weighting [35]:

$$1 - \sum_j \frac{tf_j \cdot (\,\log(tf_j) - \log(gf)\,)}{gf \cdot \log(n)},$$

where tf_j is the term's frequency in the j-th document, gf is the term's *global frequency*, or the total occurrence count of the term in the collection, and n is, as before, the number of documents in the collection. Dumais [35] found that entropy weighting produced performance advantages over other weightings such as *idf* on several small test collections for latent semantic indexing, a retrieval method I introduce later in Section 2.4.3.

Robertson and Sparck Jones [86] derived a term weighting formula exploiting term distributions in relevant and non-relevant documents. A major assumption of their work is that some relevance information is available, so their model is good mostly for relevance feedback retrieval, a process in which the query is improved or recreated based on the documents that the user indicates are relevant. Croft and Harper [29] studied the problem of improving the initial search (i.e., the search before any relevance feedback) with a probabilistic model of document retrieval based upon Robertson and Sparck Jones' term relevance weight. Under some simplifying assumptions, they showed that their document ranking formula depends inversely on the probabilities of terms being present in a non-relevant document. Since in a large collection most documents are not relevant to a given query, this last probability can be estimated by the probability of occurrence for each term, which is just df/n via the maximum likelihood estimation, an approach for determining estimates of unknown parameters of a random sample in statistics (see, for example, the textbook by Ross [90]). Hence, Croft and Harper's weighting formula is similar to the Sparck Jones' *idf* factor.

Since most term weighting functions use the product of some function of term frequency and inverse document frequency, their combination is commonly known as the $tf \times idf$ weighting.

2.3.3 Length Normalization

A third weighting factor, the document length normalization factor, appears to be useful for collections with widely varying document lengths (term counts). Without this normalization factor, long documents would have a better chance of being retrieved as relevant to a query than short ones. This is because documents with more terms in them tend to have more distinct words that results in more matches at each vector position against the query vector. They are also likely to use the matched words more often, resulting in a high term frequency value.

Several schemes have been developed to diminish this unfair advantage that long documents have over short ones.

In *cosine normalization* [97, 96, 106], each term weight is reduced by a factor of the Euclidean norm (2-norm) of the document vector. Let $\vec{d} = (d_1, d_2, \ldots, d_m)^T$ be the vector representation of a document, where d_i, $1 \leq i \leq m$, are the *tf* $\times$ *idf* weight for the i-th term in the document. Then,

$$\|\vec{d}\| = \sqrt{d_1^2 + d_2^2 + \cdots + d_m^2}$$

is the cosine normalization factor for that document. The length of a document vector in the Euclidean space becomes unit after cosine normalization, and the inner product of two vectors measures the cosine of the angle between them. It should be noted that when comparing a query against a set of documents, whether we cosine-normalize the query vector does not affect the relative values of the products between the query and the documents. In other words, document ranking is not changed when we normalize the query vector. In addition, the cosine normalization scheme gives somewhat unfair advantage to short documents, as their vector components are "boosted" because they have smaller 2-norms to be divided by.

The Okapi system document length normalization is based on the "verbosity hypothesis" that a long document covers a similar content scope to a short document but simply uses more words [88, 87]. Because the Okapi *tf* factor is derived under the assumption that documents are all of equal length, all *tf* values of a document are to be normalized by the proportion of the document's length to the average document length. The final normalized Okapi term frequency weight in a document is (see Section 2.3.1 for a comparison)

$$\frac{tf^{\,c}}{(k_1((1-b)+b\frac{dl}{avdl}))^c + tf^{\,c}},$$

where k_1 and b are constants, dl is the byte length of this document, and $avdl$ is average byte length of documents in the collection.

Finally, the *pivoted document length normalization* is a normalization scheme intended to improve upon an existing normalization function [107]. This method is motivated by experimental observations that a normalization scheme that retrieves documents of all lengths with similar probabilities as their chances of being relevant outperforms another scheme that retrieves documents with probabilities different from their their chances of being relevant. The basic form of the pivoted document length normalization is

$$\frac{1}{(1.0 - slope) + slope \times \frac{old\ \ normalization}{average\ \ old\ \ normalization}}$$

where *slope* is some constant and *old normalization* is the existing normalization factor that the pivoted normalization scheme intends to improve upon. For example, researchers at AT&T used 0.2 for *slope* and document byte length (dl) as the *old normalization* factor in their TREC-7 retrieval experiments [108].

When applied to the traditional cosine normalization, the new pivoted scheme is called *pivoted cosine normalization*. Singhal et al. [107] observed that the cosine function behaves "weaker" than a linear function of the number of unique terms in a document, and suggested using the latter for document normalization. Thus, we have the *pivoted unique normalization*:

$$(1.0 - \textit{slope}) \times \textit{pivot} + \textit{slope} \times \textit{number of unique terms.}$$

In TREC-4 experiments, researchers at Cornell University fixed *slope* at 0.2, and set the pivot to the average number of unique terms across the entire collection [24]; they found these setting to be very effective for their retrieval tasks.

The SMART retrieval system implements various $tf \times idf$ weightings and normalization functions and denotes the combined schemes by triples of letters: the first letter in a triple represent the term frequency factor, the second inverse document factor, and the last the normalization factor. Table 2.2, part of which is taken from Singhal's thesis [106], summarizes the most commonly used functions in SMART term weighting. If different term weighting schemes are applied to queries and documents, two letter triplets are used for document and queries in that order. For example, "*ntc.ann*" would mean natural term occurrence weighting, inverse document frequency weighting, and cosine normalization combined for documents and augmented term occurrence weighting with no inverse document frequency or normalization for queries.

In this study, I used various term weighting scheme mentioned in this and previous sections for my experiments. Normally, I selected one that is known to work well on the collection of interest. In other cases, I tested several weighting schemes on a collection to find the best among them. I report the weighting scheme used in each case.

2.4 Vector-Based Query-Expansion Methods

The problem with the vector space model of information retrieval is that it is still a term-term matching scheme, even though terms are weighted by their importance

Term Frequency		Inverse Document Frequency		Normalization	
$f(tf)$		$f(\frac{1}{df})$		$f(length)$	
n	tf	n	1	n	1
l	$1 + \ln(tf)$			c	$\sqrt{d_1^2 + d_2^2 + \cdots + d_m^2}$
d	$1 + \ln(1 + \ln(tf))$	t	$\log\left(\frac{n+1}{df}\right)$	b	$0.8 + 0.2 \times \frac{dl}{avdl}$
a	$0.5 + 0.5 \times \frac{tf}{\max tf}$			u	$(1.0 - slope) \times pivot +$ $slope \times \#$ *of unique terms*
L	$\frac{1 + \log(tf)}{1 + \log(\text{avg } tf)}$				

Table 2.2: The retrieval system SMART uses a triplet of letters to denote its local, global, and normalization weighting schemes.

in reflecting document content. In the vector space being constructed, terms are treated as if they are pairwise orthogonal while each term weight is the component of the document vector along the direction of the term vector.

Two major problems with term-term matching schemes like VSM are synonymy and polysemy [47, 32]. Since there are always multiple ways to express a given concept verbally (synonymy), a user query may not use terms that match those in a relevant document. For instance, a "promotional CD" is also known as a "DJ CD" among collectors and disc jockeys; nonetheless, it is very unlikely that one will find the word "promotional" to be related to "DJ" in any standard thesaurus. On the other hand, most words have multiple meanings (polysemy), so terms in a user query can literally match terms in irrelevant documents. An example here is the query "Babyface Nelson"—when looked up on Yahoo!, it would match the biographies and stories of the infamous gangster in the 1930's, as well as pages containing links to pop music artists in the early 1990's (both Babyface and Nelson happen to be pop musicians in that period).

One method for easing the user's burden in choosing query words is for the retrieval system to automatically expand the query with terms that are related to those supplied by the user. The new terms can be either statistically related to the original query words or selected from lexical aids such as a thesaurus. The expanded query is likely to help alleviate the problem of synonymy and polysemy. In terms of our earlier examples, adding the word "DJ" to the query "promotional CD" would certainly help a novice collector looking for hard-to-find CD's on eBay, and including "crime" or "FBI" in the "Babyface Nelson" query would improve search results for a user looking for information about the criminal.

2.4.1 Local Feedback

A well-known interactive method for query expansion is *relevance feedback*, where users identify relevant documents in an initial list of retrieved documents, and the system creates an improved query based on those sample relevant documents [89, 86, 98, 51]. The assumptions underlying relevance feedback are that documents relevant to a query resemble each other, and that documents not relevant to the query are different from those that are relevant [1]. The vector representation of the query can then be modified to move it toward those of the relevant documents and away from those of non-relevant ones.

The procedure of relevance feedback can be fully automated wherein vector representations of the top-ranked documents from an initial query-document match are "added" to the query vector for further retrieval [3, 23, 119]. Since no real user feedback is used in this process, this query expansion scheme is known as *pseudo-relevance feedback* [121], *local feedback* [119], or *blind feedback* [115]. In this dissertation, we adopt the terminology "local feedback" to contrast it to the "global" analysis used by many query-expansion methods on an entire document collection. The number of feedback iterations can be more than one, so that a sequence of queries will be generated with the hope of converging to a near optimal query [89].

Early results of local feedback on some small collections did not appear to be promising. This is partially due to the fact that not many top retrieved documents were relevant, and so feedback would result in a negative effect by supplying useless terms. Researchers at Cornell University applied local feedback in their TREC experiments, however, and obtained favorable results. They believed that this is due to the combination of better initial retrieval and the large number of relevant documents per query in the TREC collections [23].

A more recent approach developed by Xu and Croft [119] involves locating relevant concepts (noun phrases) from passages (fixed-size text windows) inside top ranked documents to add to the original query. Known as *local context analysis*, this method has been shown to result in improved retrieval performance over the traditional local feedback. In addition, a study by Allan [2] focused on the case when the top retrieved documents are so large that they may cover several subject areas. Such documents are more likely to expand the queries with inappropriate terms that happen to co-occur with queries terms. He compared two approaches, discarding large documents and extracting passages from them, and found both to be "valuable" under certain conditions. He also proposed a hybrid system that performs passage feedback on large documents and uses the entire text of other smaller relevant documents. Finally, Fitzpatrick and Dent [41] examined the possibility of using past queries as an additional source of evidence to improve automatic query expansion and found encouraging results. In recent years, more and more IR researchers, especially participants of TREC, started to apply some form of local feedback on top of their initial retrieval to enhance performance.

2.4.2 Generalized Vector Space Model

An early method of deriving the term correlations from analysis of the corpus matrix is the generalized vector space model (GVSM) [118, 117]. The method was developed based on an observation of the limitations associated with the conventional VSM discussed earlier. Instead of using terms as orthogonal dimensions of the information vector space, GVSM identifies a set of new *fundamental concepts* and uses them as the basis vectors of the vector space of interest. Those concept vectors are derived from term usage in various sets of documents. In the simplest case, each document represents a fundamental concept, and so each rows of the term-by-document matrix is interpreted as the component of the corresponding term vector along the directions of the concept vectors. Term correlations can then be determined by the number of matches of the concept vectors between pairs of terms. Finally, in GVSM, semantic closeness between two texts is computed by first transforming them into the concept space, or a space whose orthogonal dimensions are documents, and then measuring their spatial proximity there as in VSM.

The similarity thesaurus approach proposed by Qiu and Frei [83, 82] has the same fundamental form as GVSM. It builds a term-by-document matrix where terms are indexed by documents, a similar interpretation to that in GVSM. The difference is that the values in the matrix are created by a new function so that they represent document weights in term vectors instead of term weights in document vectors. This approach adds soundness to the GVSM technique since it is not very clear how documents are represented by term vectors and terms by document vectors simultaneously in GVSM. In Section 3.3.2, I will explain the details of building term vectors from document weights and contrast it to GVSM.

The problem of GVSM is that it may create a "distorted" semantic space. For example, suppose there are two documents in the collection on economy-related topics and twenty on sports-related topics. Then, for any vector in the GVSM space, there are two dimensions representing economy-related concepts and ten times that many representing sports-related concepts. A query request on the topic of sports economy would make a GVSM system return mostly sports-related articles.

2.4.3 Latent Semantic Indexing

Latent semantic indexing (LSI) [46, 32] sees neither terms nor documents as the optimal choice for the orthogonal dimensions of a semantic space. Instead, it uses a basis computed from the term-document matrix of VSM for its vector dimensions. Berry et al. [9] summarize the basic idea behind of LSI as follows:

> [LSI] tries to overcome the problem of lexical matching by using statistically derived conceptual indices instead of individual words for retrieval. LSI assumes that there is some underlying or latent structure in word usage that is partially obscured by variability in word choice. A truncated singular value decomposition (SVD) is used to estimate the structure in word usage across documents. Retrieval is then performed using the database of singular values and vectors obtained from the truncated SVD.

Here, singular value decomposition [49] is a matrix computation technique that converts a matrix into a product of three parts: the left and right singular vector matrices and the singular value matrix. The retrieval documents are then projected by the singular vectors representing the most dominant orthogonal dimensions of the matrix into a reduced-dimension semantic space for comparison. Details of SVD and related matrix computations of LSI are discussed in Section 3.4.

Theoretically, LSI has been proven to succeed in capturing the underlying semantics of the corpus under the condition that the corpus is a reasonably focused collection of meaningfully correlated documents [79]. In addition, Zha et al. [122] proposed a systematic way of determining the optimal LSI dimension based on the minimum description length principle from the field of array signal processing. The prerequisites of this approach include knowing the spectrum (all singular values) of the term-document matrix, the computation of which requires enormous resources when the matrix becomes very large. Zha and Zhang [124] also noticed a special *low-rank-plus-shift structure* observed in every term-document matrix: let A be a term-document matrix, then

$$A^T A = \text{a low-rank matrix} + \text{a multiple of the identity matrix.}$$

In Section 3.5.3, I show how the distribution of the singular values of the term-document matrices are related to this structure. While Zha and Zhang suggested SVD computing and updating procedures that utilize the low-rank-plus-shift structure, I present a novel algorithm for LSI approximation that takes advantage of this special property in Chapter 4.

LSI has equaled or outperformed the conventional VSM in many monolingual retrieval experiments where the document collections are small to medium, perhaps because of its ability to make term association from matrix analysis [39, 32, 36, 37, 38, 70]. But, LSI reveals its weakness when the collection sizes become large. The computation time required by SVD is typically proportional to the product of the square of the number of terms (or documents, whichever is smaller) in the collection and the number of singular vectors or orthogonal dimensions desired [111]. Even if we need only a fraction of all the singular vectors of a corpus matrix for the retrieval to be effective, the actual number can be quite large given the growing size of the modern information repositories, and it becomes infeasible to compute that many singular vectors via existing SVD algorithms. Thus, when the collection size does become very large, the limited SVD vectors we can compute are just too few for achieving good retrieval performance.

Researchers have responded to the scalability of LSI using a variety of approximation techniques, sometimes applying problem-specific constraints or ad hoc algorithms. The scalability problem is one of the foci of this dissertation.

A similar technique to LSI was developed by Newby [75] that constructs an *information space* by selecting useful terms and performs a principle component analysis (PCA) on the term-by-term matrix of correlation scores. It differs from LSI in that terms are explicitly selected for the information space and that a statistical term-by-term correlation matrix is created and analyzed, whereas LSI does its work on the term-by-document matrix[1]. Section 3.4.2 makes a detailed comparison between LSI and the information space method.

2.5 Cross-Language Information Retrieval

As introduced in Chapter 1, cross-language information retrieval (CLIR) is the problem of using ad hoc queries in one language to retrieve documents in another language. Its importance has increased enormously in recent years because the information super-highway is reachable from virtually all over the world. This

[1]In the actual computation, the SVD of the term-by-document matrix is usually found by finding the eigenvalues and eigenvectors of the symmetric matrix formed by multiplying the term-by-document matrix by its transpose, exactly the process of PCA.

dissertation treats cross-language retrieval between two languages.

Schäuble and Sheridan [101] categorized the approaches to solving the CLIR problem into three types: translation of the queries into the language of the documents, translation of the documents into the language of the queries, and translation of both the queries and the documents into some intermediary or inter-lingual representation where they can be compared. Although translating the few words in a query into the language of the documents is considerably more efficient than translating all documents into the language of the query, the lengthiness of documents makes them more reliable indicators of content and less vulnerable to translation errors.

The translation can be accomplished by using one or a combination of the following resources: bilingual corpora, bilingual machine-readable dictionaries (MRD), and machine translation (MT) tools. Next, I describe some of the basic approaches in cross-language information retrieval.

2.5.1 Vector-Based Query-Expansion Methods

The classical vector space model of information retrieval cannot be applied directly to CLIR because similarity is based on the overlap of terms between queries and documents—this is typically zero in CLIR. The problem at hand is to retrieve documents containing expressions that do not exactly match those found in the query.

Vector-based query-expansion methods such as GVSM and LSI can easily be generalized for cross-language information retrieval as long as a bilingual *comparable* or *parallel* document collection is available for training [65, 104, 26, 121, 84, 69]. Comparable corpora usually are those texts that express the same idea or describe the same event, even though they are composed independently. For example, Rehder et al. [84] used a bootstrapping procedure to extract comparable documents from the TREC French and German collections of news articles (see Appendix A.1 for some sample documents). In contrast, parallel collections are more strictly aligned; they consist of texts that are translations of each other. An example of a parallel collection is the United Nations Parallel Corpus in English, French, and Spanish produced by the Linguistic Data Consortium (LDC) (http://www.ldc.upenn.edu/). Appendix A.2 shows a sample of this collection. Comparable or parallel, the connection between the texts in different languages can be on different levels of granularity: sentence, paragraph, or document; they are *aligned* if the correspondence is one-to-one. Thus, the most coarse type of alignment would be comparable documents, while the finest type would be parallel sentences.

With the expansion of the Internet in the past decade, there are a large number of international web sites created in multiple languages. Researchers have successfully used various heuristics to automatically extract parallel or comparable corpora from the web [85, 77]. For example, the HTML markup tags of the web documents can be used to both confirm the translational relationship between the texts and provide a finer level of granularity of parallelism.

Under a vector-based query-expansion method, words of similar meaning in different languages are treated in a similar fashion to synonyms in a single language, and translation is achieved through query expansions across languages. While in monolingual retrieval, query expansion is based on term-term co-occurrence in the same documents, in cross-language retrieval term-term similarity can be derived from their corresponding appearances in matching pairs of documents from the comparable (or parallel) corpus. Vector-based query-expansion methods on CLIR are usually fully automatic, can be used with much less knowledge of the syntax or semantics of the languages involved, and can be trained with documents in a specific domain for retrieval in the same domain. These query-expansion methods have been tested and shown to perform effectively on various collections [104, 26, 84, 69, 121].

2.5.2 Machine Translation and Dictionary Methods

Strict applications of machine translation on CLIR require sophisticated and expensive linguistic tools to process the queries (or documents). In a TREC CLIR track overview, Braschler et al. [15] observed that "CLIR is a difficult problem to solve based on MT alone: queries that users typically enter into a retrieval system are rarely complete sentences and provide little context for sense disambiguation." The availability of machine translation software for different language pairs is also limited, especially when neither of the two is English. For example, in their TREC-7 experiments, Gey et al. [48] experimented with three different MT software packages to translate queries in English, French, German, and Italian to retrieve documents in all four languages; yet for translation between two non-English languages, they had to use English as a bridge: queries in a non-English language were first translated into English and then into another non-English language. Their results show English queries do a great deal better than queries in other languages in cross-language retrieval performance. A recent development in MT-based CLIR has been the use of probabilistic translation models (PTM) to translate either the query [77] or the documents [42]. The parameters of PTM are usually estimated from a bilingual parallel corpus, which has to be aligned at passage or even sentence level in order to make the model robust. The retrieval performance of these models can be very effective, though. For instance, the translation model of Franz et al. [42] not only was "fast" in translating the documents into the language of the query, but it also produced very high performance results on TREC cross-language test collections.

In dictionary-based approaches to CLIR, on the other hand, word translations are looked up in a bilingual machine-readable dictionary [62, 4, 31], as such dictionaries are sometimes more easily found than parallel text corpora. The process can be as simple as word-by-word translations [4], more sophisticated phrasal translation when appropriate [5], or even mapping of the morphological roots of query words across languages [62, 80]. In the case of languages such as Chinese and Japanese, where the written texts contain no word boundaries, word segmentation strategies needs to be applied to identify words to be translated out of consecutive strings of characters [10, 27]. When a machine-readable dictionary is not readily available or lacks the coverage of domain-specific terminologies, bilingual term or phrase correspondences can be established through comparable or parallel corpora [80, 18, 27], as in vector-based query-expansion methods. The words in two languages are usually matched statistically with little concern for the syntactic structure of either language. Collocational statistics of terms in a parallel corpus can also be used for resolving translation ambiguities and reducing translation errors in dictionary-based approaches [6, 45].

Nie et al. [77] compared results of English-French CLIR using machine translation, bilingual dictionaries, probabilistic translation models estimated from different sources, and combinations of a PTM and a dictionary. They found that while a good MT systems may provide sound translation quality for CLIR, the PTM and combination of PTM and bilingual dictionary are more "flexible" and provide equally good results. A preliminary comparison between LSI and machine translation for CLIR was made by Littman et al. [69], in which they found both LSI and MT performed equally well on a test of locating the "mate" or translation counterpart of a document in a parallel corpus. Yang et al. [121] performed a comparative study among machine translation and query-expansion methods of cross-language information retrieval with a small subset of the UN corpus from LDC. Littman and Jiang studied the differences between LSI and GVSM using the same collection [70].

Chapter 3

A Mathematically Unified View

Vector-based, automatic information retrieval methods such as VSM, GVSM, and LSI represent both queries and documents by high-dimension vectors learned from analyzing a corpus of text. Each method has its own strengths and weaknesses. VSM scales well to large collections, but cannot represent term-term correlations, which prevents it from being used in cross-language retrieval. GVSM and LSI can represent term-term correlations, but do not give good retrieval performance on very large retrieval collections. In this chapter, I describe in detailed mathematical formulae how these three basic methods and their close cousins relate to each other. I present a novel unified view on vector-based information retrieval called *dimension equalization*.

3.1 Preliminary Concepts in Linear Algebra

To begin, I would like to refresh the readers' memories with some basic concepts in vector and matrix computation that will be used in this dissertation. Let $\vec{u} = (u_1, u_2, \ldots, u_n)^T$ and $\vec{v} = (v_1, v_2, \ldots, v_n)^T$ be two n-dimension real vectors, and let $A = [a_{ij}]$, $i = 1, \ldots, m$ and $j = 1, \ldots, n$ be an $m \times n$ matrix. Then,

- The *inner product* or *dot product* of the two vectors is the scalar number given by

$$\vec{u} \cdot \vec{v} = \vec{u}^T \, \vec{v} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n.$$

- The *(Euclidean) length* or *2-norm* of a vector $\vec{v}$, denoted by $\|\vec{v}\|$, is $\sqrt{(\vec{v})^T \vec{v}}$.

- The angle θ between two nonzero vectors $\vec{u}$ and $\vec{v}$ is given by

$$\cos(\theta) = \frac{\vec{u}^T \vec{v}}{||\vec{u}|| \, ||\vec{v}||}.$$

- Two vectors $\vec{u}$ and $\vec{v}$ are *orthogonal* if the angle θ between them is $90°$, or $\vec{u}^T \vec{v} = 0$. A set of nonzero vectors $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n\}$ is *orthogonal* if $\vec{v_i}^T \vec{v}_j = 0$ when $i \neq j$.

- The n-dimension vector $\vec{v}$ can be treated as an $n \times 1$ matrix. Conversely, each row of the matrix A can be viewed as an n-dimension vector and each column of A an m-dimension vector.

- The set of all $m \times n$ real matrices is denoted by $\Re^{m \times n}$.

- Matrix A is a *diagonal matrix* if $a_{ij} = 0$ for $i \neq j$. We write $A = \text{diag}(a_{11}, a_{22}, \ldots, a_{ss})$, where $s = \min(m, n)$. If $m = n$ and $a_{ii} = 1$, then the diagonal matrix A is known as an *identity matrix* and is usually denoted by I.

- A matrix $A \in \Re^{m \times m}$ is said to be *unitary* or *orthogonal* if $A^T A = I$.

- A matrix $A \in \Re^{m \times m}$ is *invertible* or *non-singular* if there exists $B \in \Re^{m \times m}$ such that $AB = BA = I$.

- The *Frobenius norm* of a matrix A is

$$||A||_F = \sqrt{\sum_{j=1}^{n} \sum_{i=1}^{m} |a_{ij}|^2}.$$

The readers are also expected to have some knowledge of the basic concepts in linear algebra such as vector space, subspace, and basis of a vector space. Please refer to a standard linear algebra textbook (for example, the one by Lang [66]) for these concepts. Briefly, we know that the set of all vectors of the same size is a *vector space.* The *dimension* of a vector space is the maximum number of independent, nonzero orthogonal vectors contained in that space. A vector space can also contain other vector spaces with smaller dimensions, and these are known as *subspaces* of that vector space. The smallest-dimension subspace that contains all the row vectors of a matrix A is known as the *row space* of A. We also say that these row vectors *span* the row space of A. Similarly, the subspace spanned by the columns of A is called the *column space* of A. The *rank* of A, denoted $\text{rank}(A)$, is the dimension of the column space of A.

In vector-based information retrieval, documents and queries are represented by vectors, and their semantic similarity is computed by vector inner product. In this thesis, I use the following equation as a general formula for vector-based information retrieval methods:

$$\text{Sim}_P(d, q) = (P^T\vec{d}) \cdot (P^T\vec{q}) = \vec{d}^T P P^T \vec{q}, \tag{3.1}$$

where $\text{Sim}_P(d, q)$ represents the similarity between a document and a query, $\vec{d}$ is the vector representation of a document d, $\vec{q}$ is the vector representation of a query q, and P is a projection matrix that transforms $\vec{d}$ and $\vec{q}$ into a new vector of a possibly different size.

The vector $\vec{d}$ in Equation 3.1 is usually obtained this way: a document collection can be represented by a term-document matrix $D = [d_{ij}] \in \Re^{m\times n}$, where m is the size of the vocabulary, and n is the number of unique documents in the collection. The entries of matrix D are term weights as described in Section 2.3, i.e., d_{ij} is the weight of term i in document j. Specifically, $\vec{d}$ can represent any column of the matrix D, which is an m-dimension vector. I also use $D_{(j)} = (d_{1j}, d_{2j}, \ldots, d_{mj})^T$ to denote the j-th column vector of the matrix D, which corresponds to the j-th document of the collection. The equivalent vector form for query q is then $\vec{q} = (q_1, q_2, \ldots, q_m)^T$.

Notation 3.1. *Define e_j be an $n \times 1$ matrix with all elements equal to 0 (zero) except for the j-th one, which is 1 (one) instead.*

Then, $D_{(j)}$ can be expressed as De_j, as a matrix product. While $\vec{d}$ will be used most of the time in this dissertation, occasionally $D_{(j)}$ and De_j are needed in a few mathematical derivations.

3.2 Vector Space Model

In the vector space model of information retrieval, the similarity between document d and query q is measured

$$\begin{aligned}
\mathrm{Sim}(d, q) &= \sum_{i=1}^{m}\sum_{k=1}^{m} d_{ij}\, q_k\, \vec{t_i} \cdot \vec{t_k} \\
&= \vec{d}^T \begin{bmatrix} \vec{t_1}\cdot\vec{t_1} & \vec{t_1}\cdot\vec{t_2} & \ldots & \vec{t_1}\cdot\vec{t_m} \\ \vec{t_2}\cdot\vec{t_1} & \vec{t_2}\cdot\vec{t_2} & \ldots & \vec{t_2}\cdot\vec{t_m} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \vec{t_m}\cdot\vec{t_1} & \vec{t_m}\cdot\vec{t_2} & \ldots & \vec{t_m}\cdot\vec{t_m} \end{bmatrix} \vec{q} \\
&= \vec{d}^T C\, \vec{q}, \qquad (3.2)
\end{aligned}$$

where $\vec{t_i}$ and $\vec{t_k}$ are unit term vectors in dimensions i and k, and C is an $m \times m$ *term correlation matrix*. Comparing Equation 3.2 to Equation 3.1, we see that $C = PP^T$.

The conventional vector space model (VSM) makes the term-independence assumption that any two index words are unrelated, or represent orthogonal dimensions in the semantic space. In the above equation, this is equivalent to

$$\vec{t}_i \cdot \vec{t}_k = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

In other words, the correlation matrix C is an identity matrix I. The VSM similarity measure becomes simply

$$\mathrm{Sim}_{VSM}(d,\, q) = \vec{d}^T\, I\, \vec{q} = (I^T \vec{d}\,) \cdot (I^T \vec{q}\,) = \sum_{i=1}^{m} d_{ij}\, q_i. \tag{3.4}$$

Thus, in the form of Equation 3.1, the projection matrix P in VSM is the $m \times m$ identity matrix.

A criticism of VSM is that it treats terms as unrelated in that they occupy orthogonal dimensions in the semantic space. A nonzero similarity score between two vectors will result only if they both have nonzero values in at least one dimension, i.e., their original documents contain at least one term in common. This is certainly far from ideal because, for example, for a query that contains the search word "computer" but not "digital," a document that uses exclusively the word "digital" will be deemed no more relevant than a document that talks about gardening. It is also obvious that the exact term matching scheme of VSM is not applicable to cross-language information retrieval.

Query-expansion techniques such as the generalized vector space model and local feedback have been invented to deal with the word mismatch problem in VSM [119]. These techniques often use a *training corpus* to derive term-term correlations, and related terms can be added to the original query to enrich retrieval. Ideally, the training corpus would simply be a thesaurus that lists for every word all the related words and how strong the relationships are between them. At the other extreme, this “thesaurus” would only contain word entries followed by no related words at all, and this is exactly analogous to the case in conventional VSM. What query-expansion methods can derive from a real training corpus is some statistical link between terms that appears together in the same documents.

In monolingual information retrieval, the training is usually done on the retrieval collection itself, but other possibilities also exist: a subset of the retrieval collection or even a completely different collection can be used. In cross-language retrieval, a bilingual corpus is needed for training, where corresponding documents in two languages are translations of each other or are on the same or related subjects. In this dissertation, I use symbols $A = [a_{ij}]$ (and $B = [b_{ij}]$ for cross-language) to denote the matrix for the training corpus, and $D = [d_{ij}]$ to represent the matrix for the retrieval collection. Unless specified, I generally do not assume $A = D$.

Next, I examine methods that compute the term correlation matrix C automatically from a training corpus.

3.3 Generalized Vector Space Model

The generalized vector space model (GVSM) [118, 117] of information retrieval measures term-term correlations based on the term-term co-occurrence information derived from a training corpus. Each term vector $\vec{t_i}$, $1 \leq i \leq m$, is expressed as the vector sum of a set of orthogonal basis vectors or minterms $\vec{m}_k$ that represent “fundamental concepts” [117]:

$$\vec{t_i} = \sum_{k=1} c_{ik}\vec{m}_k,\ 1 \leq i \leq m, \qquad (3.5)$$

where c_{ik} indicates how much term vector $\vec{t_i}$ weighs in the direction of $\vec{m}_k$. Since these basis vectors are derived from examining disjoint subsets of documents where terms co-occur, in the worst case each document in the training corpus represents a fundamental concept. So, the number of basis vectors will not exceed n, the total number of documents in the collection A.

In fact, the version of GVSM often seen does make this oversimplifying assumption—that documents express disjoint concepts and so their associated vectors are orthogonal. The term vectors can, thus, be rewritten as

$$\vec{t_i} = \sum_{k=1}^{n} a_{ik} \vec{f_k}, \tag{3.6}$$

where a_{ik}, is the weight of i-th term in the k-th document vector of A (the matrix of the training corpus), and $\vec{f_k}$ corresponds to the fundamental concept vector for the k-th document in the training collection. The assumption of orthogonality between documents is equivalent to the following property:

$$\vec{f_i} \cdot \vec{f_j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

In other words, the correlation between the i-th term and the j-th term is:

$$\vec{t_i} \cdot \vec{t_j} = \sum_{k=1}^{n} \sum_{l=1}^{n} a_{ik} a_{jl} \vec{f_k} \cdot \vec{f_l} = \sum_{k=1}^{n} a_{ik} a_{jk}. \tag{3.8}$$

In this case, the term correlation matrix becomes

$$\begin{aligned} C &= \begin{bmatrix} \vec{t}_1 \cdot \vec{t}_1 & \vec{t}_1 \cdot \vec{t}_2 & \dots & \vec{t}_1 \cdot \vec{t}_m \\ \vec{t}_2 \cdot \vec{t}_1 & \vec{t}_2 \cdot \vec{t}_2 & \dots & \vec{t}_2 \cdot \vec{t}_m \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \vec{t}_m \cdot \vec{t}_1 & \vec{t}_m \cdot \vec{t}_2 & \dots & \vec{t}_m \cdot \vec{t}_m \end{bmatrix} \\ &= \begin{bmatrix} \sum_{k=1}^n a_{1k}\, a_{1k} & \sum_{k=1}^n a_{1k}\, a_{2k} & \dots & \sum_{k=1}^n a_{1k}\, a_{mk} \\ \sum_{k=1}^n a_{2k}\, a_{1k} & \sum_{k=1}^n a_{2k}\, a_{2k} & \dots & \sum_{k=1}^n a_{2k}\, a_{mk} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \sum_{k=1}^n a_{mk}\, a_{1k} & \sum_{k=1}^n a_{mk}\, a_{2k} & \dots & \sum_{k=1}^n a_{mk}\, a_{mk} \end{bmatrix} \\ &= AA^T. \end{aligned} \tag{3.9}$$

The $m \times m$ matrix AA^T has a nonzero value in its row i and column j if and only if there is a document in A that contains both the i-th and j-th terms [28].

Thus, for GVSM, the similarity measure in Equation 3.2 can be rewritten as

$$\text{Sim}_{GVSM-ML}(d,\, q) = \vec{d}^T AA^T\, \vec{q} = (A^T \vec{d}\,) \cdot (A^T \vec{q}\,). \tag{3.10}$$

This tells us that GVSM uses the indexed term-document training matrix A as its projector P. Matching the vector components in $\vec{d}$ and rows of A by the terms they represent, $A^T \vec{d}$ transforms $\vec{d}$ into a new vector in the row space of A, and its components correspond to the n documents in the training corpus. The query vector $\vec{q}$ is also transformed by A in the same way, and so $\vec{d}^T AA^T\, \vec{q}$ compares the query and the document by the documents they are associated with.

The idea of using AA^T as the term-term similarity matrix has also been studied by Salton and McGill [99]. They set a threshold value on the elements of AA^T and cluster terms with a correlation value greater than the threshold into the same class. The resulting term classes can be organized in a hierarchy or a network. Removing small elements from the similarity matrix AA^T can reduce the effort of computing query expansions while leaving its structure mostly untouched.

Another technique of cutting the computational cost, called *sparsification*, is proposed by Yang [120]. This cost-cutting approach keeps only the most influential (largest) components of the projected vectors before comparing their similarity.

Notation 3.2. *Let $\vec{v}$ be an n-dimension vector. Define the* sparsification operator $\mathrm{Sp}_k(\vec{v})$, *$k \leq n$, to return a new vector that is the same as $\vec{v}$ except replacing its smallest $n - k$ elements by 0 (zero).*

Then, the term-document similarity is computed

$$\mathrm{Sim}_{GVSM-ML}(d, q) = \mathrm{Sp}_k(A^T \vec{d}) \cdot \mathrm{Sp}_k(A^T \vec{q}). \tag{3.11}$$

In other word, this *sparsification* method first does the transformations $A^T \vec{d}$ and $A^T \vec{q}$, then sets all but k largest-value elements of the resulting vectors to zero, and finally calculates the inner product of the two vectors. Yang et al. [26, 121] used 100 and 200 for the value of k in their experiments.

3.3.1 Cross-Language Retrieval

The extension of GVSM to cross-language retrieval was proposed by Yang et al. [26]. Assuming a bilingual corpus for training, two matrices $A = [a_{ij}] \in \Re^{m \times n}$ and $B = [b_{ij}] \in \Re^{p \times n}$ are formed, where A is a term-document matrix for the training documents in the language of documents being retrieved, and B is a term-document matrix for the training documents in the language of the query.

While the numbers of unique terms in the two languages are different (denoted by m and p above), the number of training documents are the same (n), with the corresponding columns of A and B representing the matching pairs of documents in the bilingual corpus.

As in the case of monolingual retrieval, a term vector in a multi-lingual corpus is the weighted sum of fundamental concept vectors that are represented by individual documents. Since each concept vector describes one pair of parallel documents in the two languages, terms in different languages can be associated with each other via the fundamental concepts (documents) they share. Specifically, the matrix

$$C = AB^T = \begin{bmatrix} \sum_{k=1}^{n} a_{1k}\, b_{1k} & \sum_{k=1}^{n} b_{1k}\, b_{2k} & \dots & \sum_{k=1}^{n} a_{1k}\, b_{mk} \\ \sum_{k=1}^{n} a_{2k}\, b_{1k} & \sum_{k=1}^{n} a_{2k}\, b_{2k} & \dots & \sum_{k=1}^{n} a_{2k}\, b_{mk} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \sum_{k=1}^{n} a_{mk}\, b_{1k} & \sum_{k=1}^{n} a_{mk}\, b_{2k} & \dots & \sum_{k=1}^{n} a_{mk}\, b_{mk} \end{bmatrix}$$

is the cross-language term similarity matrix. For a document $\vec{d}$ from the retrieval collection and a query $\vec{q}$ in a different language, the similarity comparison between them is defined as:

$$\text{Sim}_{GVSM-CL}(d,\, q) = \vec{d}^T AB^T \vec{q} = (A^T \vec{d}\,) \cdot (B^T \vec{q}\,). \tag{3.12}$$

3.3.2 Similarity Thesaurus

In GVSM, the elements of the training matrix A are obtained from one of the term weighting schemes discussed in Section 2.3. Those weighting schemes were designed to represent documents as vectors and terms as vector components. Thus, it is somewhat contradictory that terms in GVSM are represented by fundamental concept vectors, which turn out to be just the individual documents, while at the same time the indexing method being used assumes that documents are composed of term vectors.

The similarity thesaurus developed by Qiu and Frei [83, 82] at the Swiss Federal Institute of Technology (ETH) tries to avoid this problem by treating training documents as indexing features for retrievable terms. Also known as the *dual space* approach, this method creates a document-term matrix with an indexing formula that is different from the traditional *tf* × *idf* scheme [104]. In particular, the inverse document frequency of a term is replaced by the *inverse term frequency* of a document, which is an inverse function of the number of different terms that the document contains. The weight of the j-th document in the i-th term vector is

$$w(t_i, d_j) = \begin{cases} \left(0.5 + 0.5 \times \dfrac{tf(t_i,\, d_j)}{\max\limits_k\, tf(t_i,\, d_k)}\right) \times itf(d_j) & \text{if } tf(t_i,\, d_j) > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.13}$$

where $tf(t_i, d_j)$ is the usage frequency of term t_i in document d_j, and $itf(d_j) = \log(m/|d_j|)$ is the inverse term frequency of document j. Similar to the *idf* factor used in SMART, here m is the total number of indexing terms in the collection and $|d_j|$ is the number of different terms appearing in d_j.

Therefore, all terms in a collection can be represented by a document-term matrix $E = [e_{ji}]$ where $e_{ji} = w(t_i,\, d_j)$. The i-th column of E represents the i-th term vector. According to Qiu [82], each term vector is also cosine normalized. The term-term thesaurus matrix is then computed as

$$C = E^T E, \tag{3.14}$$

and the document-query similarity is simply

$$\text{Sim}_{ST-ML}(\vec{d}, \vec{q}\,) = \vec{d}^T E^T E\, \vec{q} = (E\vec{d}\,) \cdot (E\vec{q}\,). \tag{3.15}$$

Here, the transformation matrix is $P = E^T$. Comparing this to Equation 3.10, we see that the matrix B functions like the transpose of matrix A, the only difference being how they are indexed.

The dual space approach has been built into the SPIDER information retrieval system developed at ETH [104]. In the actual process of building the thesaurus, only "good" terms that do not have very high or low document frequencies are considered for expansion, and only pairs of terms that have high similarity scores are stored in the final data structure to aid retrieval. This approach is very similar to the threshold value used by Salton and McGill [99] on their similarity matrices.

The SPIDER system has also successfully extended the similarity thesaurus method to applications of cross-language information retrieval [104, 16, 14]. As in the case of monolingual retrieval, the basic equation is the same as that of GVSM, with the primary difference being the way indexing features of term vectors are constructed:

$$\text{Sim}_{ST-CL}(\vec{d}, \vec{q}\,) = \vec{d}^T\, E^T F\, \vec{q} = (E\vec{d}\,) \cdot (F\vec{q}\,),$$

where F is a document-term matrix similar to E but constructed from the training documents in the language of the query.

3.3.3 Local Feedback

Another flavor of the GVSM method is *local feedback*, which was first discussed by Attar and Fraenkel [3]. In their work, top returned documents from an initial retrieval run were assumed to be relevant and were used to expand the query. This is different from Rocchio's classic relevance feedback approach [89], where real relevance information from the user who submits the query is obtained and used. Hence, local feedback is sometimes also known as pseudo-relevance feedback [121]. Rocchio's formula for adding additional terms and reweighting the query, however, is frequently used in local feedback:

$$\vec{q}_{LF} = \alpha\,\vec{q} + \frac{\beta}{n_1}\sum_{i=1}^{n_1} R_i - \frac{\gamma}{n_2}\sum_{i=1}^{n_2} S_i, \tag{3.16}$$

where n_1 and n_2 are the numbers of relevant and non-relevant documents, respectively, R_i and S_i are vector representations of those relevant and non-relevant documents, respectively, and α, β, and γ are referred to as Rocchio weights. The optimal values of these weights are usually experimentally determined. In local feedback, there is no information or assumption about non-relevant documents, and so γ is assigned zero.

Now, compared to GVSM, we observe that the selection of top ranked documents for feedback is very similar to computing $A^T\vec{q} = D^T\vec{q}$ (assuming training on the retrieval collection) and performing sparsification (i.e., $\mathrm{Sp}_k(D^T\vec{q}\,)$). Here, the sparsification number k in Equation 3.11 is set to the number n_1 in Rocchio's formula. If we then left-multiply $\mathrm{Sp}_{n_1}(D^T\vec{q}\,)$ by D again, it amounts to computing the vector sum of the top ranked documents, but each document vector is weighted by the score of relevance between the query and that document. To compute $\sum_{i=1}^{n_1} R_i$ (summing up the vectors of the top-ranked documents without weighting them) as in Equation 3.16, I first introduce a new notation.

Notation 3.3. *Let $\vec{v}$ be a n-dimension vector. Then $\mathrm{Spn}_k(\vec{v})$, $k \leq n$, sets the k largest elements of $\vec{v}$ to 1 (one), replaces the other $n-k$ elements of $\vec{v}$ by 0* *(zero), and returns the new vector. Let D be an $n \times n$ diagonal matrix. Then* $\mathrm{Spn}_k(D)$ *does the same operation on the diagonal elements of D.*

Using this notation, Rocchio's formula can be expressed in a GVSM-like fashion:

$$\vec{q}_{LF-ML} = \alpha\,\vec{q} + \frac{\beta}{n_1}\,D\,\mathrm{Spn}_{n_1}(D^T\vec{q}\,). \tag{3.17}$$

When this newly expanded query vector is compared to a document vector again for relevance score, the formula is

$$\begin{aligned}
\mathrm{Sim}_{LF-ML}(d,\,q) &= (\vec{d}\,)\cdot(\vec{q}_{LF-ML}\,)\\
&= (\vec{d}\,)\cdot(\alpha\,\vec{q} + \frac{\beta}{n_1}\,D\,\mathrm{Spn}_{n_1}(D^T\vec{q}\,))\\
&= \alpha\,(\vec{d}^{\,T}\vec{q}\,) + \frac{\beta}{n_1}\,(\vec{d}^{\,T}D\,\mathrm{Spn}_{n_1}(D^T\vec{q}\,))\\
&= \alpha\,\mathrm{Sim}_{VSM}(d,\,q) + \frac{\beta}{n_1}\,(\vec{d}^{\,T}D\,\mathrm{Spn}_{n_1}(D^T\vec{q}\,)).
\end{aligned}$$

In words, the local feedback methods use Rocchio weights to combine the score of VSM and $\vec{d}^{\,T}D\,\mathrm{Spn}_{n_1}(D^T\vec{q}\,)$, which looks very much like the GVSM formula in Equation 3.10: $\vec{d}^{\,T}AA^T\vec{q} = \vec{d}^{\,T}DD^T\vec{q}$ when $A = D$.

Yang et al. [121] extended local feedback to cross-language retrieval as they did with GVSM. The idea is to find top relevant training documents in the language of the query and substitute them with their translations in the language of the retrieval collection. The cross-language feedback query is

$$\vec{q}_{LF-CL} = B\,\mathrm{Spn}_{n_1}(A^T\vec{q}\,),$$

where A and B are parallel training matrices as defined in Section 3.3.1. I use A (and B) instead of D here because the training collection is usually different from the retrieval collection in cross-language retrieval. The $\alpha\vec{q}$ part in Equation 3.17 is set to zero when we apply the original query cross-lingually. As a result, the weighting on the feedback part, β/n_1, is set to 1. Hence, the following is the cross-language similarity comparison formula for local feedback:

$$\mathrm{Sim}_{LF-CL}(d,\, q) = \vec{d}^T B\, \mathrm{Spn}_{n_1}(A^T\vec{q}\,) = (B^T\vec{d}\,) \cdot (\mathrm{Spn}_{n_1}(A^T\vec{q}\,)),$$

which closely resembles the cross-language GVSM formula (Equation 3.12). Davis and Dunning [30] experimented with a similar approach on the TREC English-Spanish retrieval: they set n_1 to be 100 and used only the 100 most frequent terms (after eliminating the 500 most frequent ones first) from those top documents (i.e. $B\,\mathrm{Spn}_{100}(A^T\vec{q}\,)$) to create the new query.

In general, local feedback leads to improved retrieval performance. Most of the groups that participate in TREC take advantage of this technique in one way or another. Some researchers make a distinction between local feedback and a method such as GVSM that derives the term correlation from the entire training corpus. They call the latter *global analysis* in contrast to analyzing the specific subset of documents returned from the initial query in local feedback. An excellent comparative evaluation of the global and local methods was given by Xu and Croft [119]. Another study of relevance feedback by Allan [2] was focused on finding optimal values for factors such as document or passage size and number of documents retrieved (i.e., the number n_1 above). The effectiveness of local feedback analysis can also be used together with the IR method I discuss next: latent semantic indexing.

3.4 Latent Semantic Indexing

Except for certain kind of corpora that are relatively small, the GVSM assumption that documents represent non-overlapping concepts is not generally valid. Latent semantic indexing (LSI) [32] is another vector-based query-expansion method that uses neither terms nor documents as the orthogonal basis of a semantic space. Instead, it computes the most significant orthogonal dimensions in the term-document matrix of the corpus, via singular value decomposition, and projects documents into the lower rank subspace thus found (known as the latent semantic space). LSI then computes semantic similarity by the proximity between projected vectors. LSI has been successfully applied to various document collections and has achieved favorable results, sometimes significantly outperforming VSM [32, 36, 37].

LSI uses SVD to factor the term-document training matrix A into three factors:

$$A = U\,\Sigma\,V^T = U\,\mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p)\,V^T, \tag{3.18}$$

where $U = (u_1, u_2, \ldots, u_m) \in \Re^{m\times m}$ and $V = (v_1, v_2, \ldots, v_n) \in \Re^{n\times n}$ are unitary matrices (i.e. $U^TU = I$ and $V^TV = I$) whose columns are the left and the right *singular vectors* of A, respectively, $\Sigma \in \Re^{m\times n}$ is a diagonal matrix whose diagonal elements are non-negative and arranged in descending order (i.e. $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$), and $p = \min(m, n)$. The values $\sigma_1, \sigma_2, \ldots, \sigma_p$ are known as the *singular values* of A, and are the square roots of the eigenvalues of A^TA and AA^T. An illustration of SVD is shown in Figure 3.1.

Suppose the rank of A is r. Then, $r \leq p$ and only $\sigma_1, \sigma_2, \ldots, \sigma_r$ are positive, while the remaining $(p - r)$, if $r < p$, singular values are zero. In LSI retrieval,

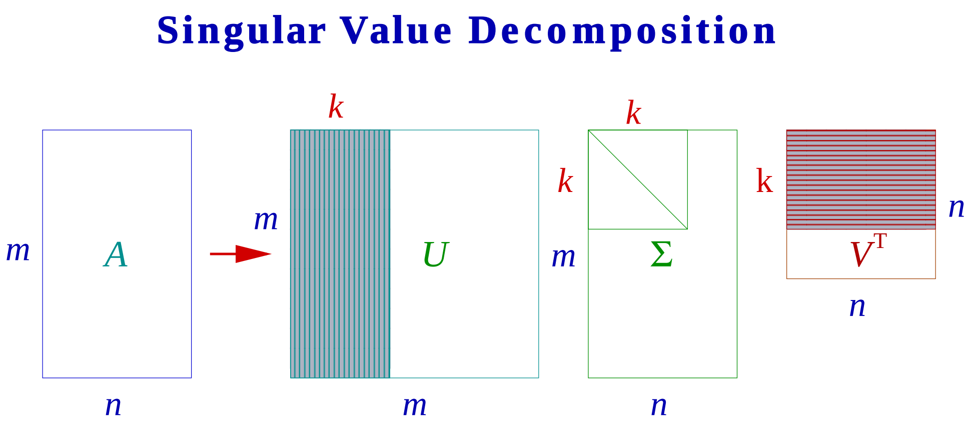

Figure 3.1: A matrix is decomposed into three matrices of singular values and vectors, and the most significant of these triplets can be used to form a low-dimension approximation to the original matrix.

researchers are only concerned with using the first r singular vectors of A, since the rest of them, if any, correspond to the zero singular values.

LSI uses the structure from SVD to obtain the reduced-dimension form of the training matrix A as its "latent semantic space."

Notation 3.4. *For $k \leq r$, define the* reduced-dimension *form of A to be*

$$A_k = U\,\Sigma_k\,V^T = U\,\mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k, 0, \ldots, 0)\,V^T.$$

That is, A_k is obtained by discarding the $r - k$ least significant singular values and the corresponding left and right singular vectors of A (since they are multiplied by zeros now). Then, the first k ($k \leq r$) columns of U that correspond to the k largest singular values of A together constitute the projection matrix for LSI:

$$\mathrm{Sim}_{LSI-ML}(d, q) = (I_k U^T \vec{d}\,) \cdot (I_k U^T \vec{q}\,) = \vec{d}^{\,T}\, U I_k U^T \vec{q}, \tag{3.19}$$

where I_k is like the identity matrix I expect that only its first k diagonal elements are nonzero (they are one). Here, each row of $I_k U$ is a k-dimension vector representing a term in the training collection. In analogy to VSM, the vector representation of a document is the weighted sum of the vector representation of its constituent terms. For a document vector $\vec{d}$ and a query vector $\vec{q}$, $I_k U^T \vec{d}$ and $I_k U^T \vec{q}$ are now the LSI vector representations of that document and query, respectively, in the reduced-dimension vector space. This process is known as "folding in" documents (or queries) into the training space [37, 9]. When $k < r$, we say that we performed *dimension reduction* on U (or A). Figure 3.1 shows the result of computing an SVD on A and using its k most dominant singular vectors and values for the dimension reduction.

An interesting observation is that according to Notation 3.3, $I_k = \mathrm{Spn}_k(\Sigma)$. Hence, we can rewrite Equation 3.19 as

$$\mathrm{Sim}_{LSI-ML}(d,\, q) = (I_k U^T \vec{d}\,) \cdot (I_k U^T \vec{q}\,) = (\mathrm{Spn}_k(\Sigma)\, U^T \vec{d}\,) \cdot (\mathrm{Spn}_k(\Sigma)\, U^T \vec{q}\,) \quad (3.20)$$

LSI's dimension reduction uses the same operator for sparsification the local feedback method, but on a very different object (matrix). Since the singular values in Σ are arranged in descending order, $\mathrm{Spn}_k(\Sigma)$ is the same as taking the following Heaviside unit step function (see, for example, the book by Bracewell [13]) on these values:

$$H_k(x) = \begin{cases} 1 & \text{if } x \geq \sigma_k \\ 0 & \text{otherwise} \end{cases}$$

Mathematically, we can also view the transformation of $\vec{d}$ and $\vec{q}$ by $I_k U$ as *orthogonal projections* onto the low-dimension space A_k. In the terminology of linear algebra, $U I_k U^T$ is the unique *orthogonal projection* onto the range of A_k [49], The projection of $\vec{d}$ and $\vec{q}$ under $U I_k U^T$ are thus $U I_k U^T \vec{d}$ and $U I_k U^T \vec{q}$.

Since

$$\begin{aligned}(UI_kU^T\vec{d})\cdot(UI_kU^T\vec{q}) &= \vec{d}^TUI_kU^TUI_kU^T\vec{q}\\ &= \vec{d}^TUI_kU^T\vec{q} \qquad \text{(since } U^TU = I \text{ and } I_kI_k = I_k\text{)}\\ &= \text{Sim}_{LSI-ML}(d,\, q), \qquad (3.21)\end{aligned}$$

we are indeed comparing the projected images of document and query vectors on the range space of A_k when we compute LSI similarities.

As in GVSM, monolingual LSI usually uses the original collection matrix D for training, i.e., $A = D$. Then, the projected vector of the j-th document on $A_k = D_k$ is

$$UI_kU^TD_{(j)} = UI_kU^TDe_j = UI_kU^TU\Sigma V^Te_j = UI_k\Sigma V^Te_j = U\Sigma_kV^Te_j = A_ke_j.$$

In words, the representation of the j-th document of A in the reduced-dimension space is simply the j-th column of A_k. The similarity comparison between the reduced-dimension document A_ke_j and the projected query $UI_kU^T\vec{q}$ is

$$\begin{aligned}(UI_kU^TD_{(j)})\cdot(UI_kU^T\vec{q}) &= (D_ke_j)^T\,UI_kU^T\vec{q}\\ &= e_j^T\,D_k^T\,UI_kU^T\vec{q}\\ &= e_j^T\,V\Sigma_kU^TUI_kU^T\vec{q}\\ &= e_j^T\,V_k\Sigma_kI_kU^T\vec{q}\\ &= e_j^T\,D_k^T\vec{q} \qquad \text{(since } \Sigma_kI_k = \Sigma_k\text{)}\\ &= (D_ke_j)\cdot(\vec{q}\,),\end{aligned}$$

which gives a simplified formula for LSI measurement when we train directly on the retrieval matrix D: Once we have the SVD dimension reduction of D, we simply compute $I_k U^T \vec{q}$ and compare it to the j-th column vector in $\Sigma_k V^T$.

Empirically, the computations of $I_k U^T$, $\Sigma_k V$, and others mentioned above do not require that we obtain the complete set of left and right singular vectors in U and V, which are very large matrices and impossible to obtain (with current resources) for a large term-document matrix. Instead, since the matrices I_k and Σ_k have nonzero elements only until the k-th diagonal element, it is adequate to just compute the first k columns (singular vectors) of U and V for retrieval purpose. I use U and V in the formulae of this dissertation merely for notational coherence.

The use of SVD for dimension reduction has the property that the most insignificant dimensions in the projection matrix are always discarded first. This is shown by the Eckart-Young theorem [40]:

Theorem 3.1. *Let $U\,\Sigma\,V^T$ be the SVD of matrix A, and $A_k = U\,\Sigma_k\,V^T$ be the reduced-dimension form of A, where $k \leq \operatorname{rank}(A)$. Then*

$$\min_{\operatorname{rank}(B)=k} \|A - B\|_F = \|A - A_k\|_F = \|A - U I_k U^T A\|_F.$$

That is, A_k is the closest to A (in terms of the Frobenius norm) out of all matrices of rank k. The intuition behind LSI is that the best k topics underlying the document collection are captured by the k dominant dimensions, with the dominance determined by the corresponding singular values. In this way, LSI learns the structure of the training materials, and uses this in making new judgments of query-document relatedness. In many experiments, as well as some theoretical analysis [79], keeping only the most significant dimensions and discarding the rest does result in improved retrieval performance.

3.4.1 LSI and Query Expansion

The forms of Equation 3.10 (or Equation 3.15) and Equation 3.19 look much alike. As the term similarity matrix used in the GVSM-based methods is AA^T, naturally we can think of the matrix UI_kU^T as the term-association matrix for LSI. Specifically, I_kU is an $m \times k$ matrix whose rows are vector representations of the m terms in the collection. The j-th value in a row vector represents how much the corresponding term weighs in the j-th most significant orthogonal dimension of the training space defined by A_k. UI_kU^T simply computes the dot product between every pair of reduced-dimension term vectors in U_k.

From a different point of view, we can compute the singular value decomposition on the similarity matrix AA^T of GVSM and find its latent semantic structure. Given the SVD of A as in Equation 3.18, the SVD of AA^T is

$$\begin{aligned} AA^T &= (U\,\Sigma\,V^T)\,(U\,\Sigma\,V^T)^T \\ &= U\,\Sigma\,V^T\,V\,\Sigma^T\,U^T \\ &= U\,\Sigma\,\Sigma^T\,U^T \qquad \text{(since } V^T\,V = I\text{)} \\ &= U\,\Sigma^2\,U^T \qquad \text{(since } \Sigma^T = \Sigma\text{).} \end{aligned} \tag{3.22}$$

Since U is a unitary matrix and Σ^2 is still diagonal, $U\,\Sigma^2\,U^T$ is exactly the singular value decomposition of the term-association matrix $A\,A^T$. In this new perspective, UI_kU^T is also the orthogonal projection onto the range of $A_kA_k^T$, the reduced-dimension form of the term-association matrix AA^T. Then, the LSI similarity comparison given by Equation 3.19 can be viewed as expanding the query $\vec{q}$ with UI_kU^T in the term-association subspace $A_k\,A_k^T$, and then comparing it to the document vector $\vec{d}$:

$$\text{Sim}_{LSI-ML}(d,\,q) = (I_kU^T\vec{d}\,)\cdot(I_kU^T\vec{q}\,) = \vec{d}^T\,UI_kU^T\vec{q} = (\vec{d}\,)\cdot(UI_kU^T\vec{q}\,). \tag{3.23}$$

Schütze's approach [102] to determining word senses is similar to the current view of LSI retrieval: He computed the SVD of a term-term co-occurrence matrix to find reduced-dimension vector representations of terms that reflect their closeness in meaning. One major difference between his approach and LSI is that he used a context window size of 1,000 characters to obtain term co-occurrence data.

As mentioned before, diagonal elements of Σ^2 are also known as the eigenvalues of AA^T, while the corresponding column vectors of U are called eigenvectors. Viewed this way, LSI is very closely related to the information space model that uses the principal component analysis on the term correlation matrix—I will discuss this shortly in the next section.

Since $(\vec{d}\,) \cdot (UI_kU^T\vec{q}\,) = (UI_kU^T\vec{d}\,) \cdot (\vec{q}\,)$, the LSI formula of Equation 3.23 can also been viewed as if it is performing document expansion before matching against the query. An interesting result arises when expansion is performed on both the documents and the query. If both $\vec{d}$ and $\vec{q}$ are multiplied by UI_kU^T at the same time and then their similarity are compared, the formula is exactly the same as expressed by Equation 3.21. Since $U^TU = I$, only one UI_kU^T is left in the end. This shows query expansion in LSI is *idempotent*: one UI_kU captures all levels of expansion. GVSM term-association matrix, on the other hand, only captures second-order term co-occurrence. To get higher order term co-occurrences, more copies of the transformation matrix A needs to be inserted before the query and document vectors. For example, the following equation captures third-order term co-occurrence:

$$(AA^T\vec{d}\,) \cdot (AA^T\vec{q}\,) = \vec{d}^TAA^TAA^T\vec{q} = \vec{d}^TU\Sigma^4U^T\vec{q}.$$

This formula establishes a positive relation between two terms if they both co-occur with a third term in some documents from the training collection. The problem with this is obvious: higher and higher powers of the singular values appear in the middle. I will come back to this point in Section 3.5.3.

3.4.2 Geometric Information Space

A method similar to LSI called Information Space (Ispace) was developed by Newby [74]. It operates by identifying eigenvectors and eigenvalues of a term-by-term correlation or covariance matrix. The covariance matrix is derived from term co-occurrence counts and the procedure of finding its eigenvalues and eigenvectors is known as principal components analysis (PCA) in statistics.

The information space is created and applied to document retrieval by the following steps:

(1) A set of specific terms is selected from the document collection for representing the information space.

(2) A term-by-term co-occurrence matrix is generated for those terms and their correlation data is derived.

(3) Principal components analysis is applied to the correlation matrix to form a reduced-dimension information space.

(4) Documents and queries are located at the geometric center of all the terms that occur in them and can be compared against each other in the information space.

(5) Similarity is measured by the geometric distance from each document to each query.

Among these steps, (4) and (5) are the same as in VSM or LSI, while (1) differs from indexing in LSI only in that LSI computes the relations among all terms in a corpus. The most significant contrast between LSI and information space lies in Step (2), and once that is clear, we will see that Step (3) is essentially the same process as in LSI.

The term co-occurrence matrix in Step (2) is a square matrix with each row or column representing a selected term. Let τ_x and τ_y denote the x-th and y-th selected terms, respectively. Then, the element at row x column y and at row y column x of the co-occurrence matrix indicates the number of times τ_x and τ_y co-occur in a document. Now, let two random variables O_x and O_y be the indicator variables for whether τ_x and τ_y occur in a document. Then, the correlation of any two terms can be computed using the following formula (see Appendix B.1 for the derivation):

$$\mathrm{Corr}(O_x,\, O_y) \approx \frac{n\, df(\tau_x \,\&\, \tau_y) - df(\tau_x)\, df(\tau_y)}{\sqrt{df(\tau_x)\,(n - df(\tau_y))\, df(\tau_y)\,(n - df(\tau_y))}}, \tag{3.24}$$

where n as before is the number of training documents, $df(\tau_x)$ and $df(\tau_y)$ are the document frequency of the terms τ_x and τ_y, respectively, and $df(\tau_x \,\&\, \tau_y)$ is the number of documents in which both τ_x and τ_y appear.

Then, for a set of m' selected terms, a term-term correlation matrix $C' = [c'_{ij}]$ can be constructed where the $c'_{ij} = \mathrm{Corr}(O_i,\, O_j)$. Except for the way that matrix elements are derived, this correlation matrix is the same as the term-association matrix $C = AA^T$ used in GVSM. In the information space method, PCA is then used to find the eigenvalue and eigenvectors of the matrix C':

$$(U')^T\, C'\, U' = \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{\mathrm{m'}}), \tag{3.25}$$

where Λ contains the eigenvalues of C', and U' is orthogonal and contains the eigenvectors of C'. The eigenvectors that correspond to the the first k eigenvalues

	LSI	Information Space
Term Selection	use all m terms	select $m' < m$ "good" terms
Term Association Matrix	$C = AA^T$	$C' = [\text{Corr}(O_i, O_j)]$
Computation	$A = U\Sigma V^T$ so that $U^T CU = U^T AA^T U = \Sigma^2$	$(U')^T C' U' = \Lambda$
Dimension Reduction	first k columns of U	first k columns of U'
Variance Captured	$(\sum_{i=1}^{k} \sigma_i^2)/(\sum_{i=1}^{r} \sigma_i^2)$	$(\sum_{i=1}^{k} \lambda_i)/(\sum_{i=1}^{m'} \lambda_i)$

Table 3.1: The information space method differs from LSI in only the selection of terms and creation of the term-association matrix.

are the first k *principle components* of C'. Furthermore, if we consider the m' selected terms as a random vector $T = (\tau_1, \tau_2, \ldots, \tau_{m'})$, then the proportion of total variance (of their occurrences in documents) due to the first k principal components is

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{m'} \lambda_i}.$$

Typically, k is selected where the above proportion reaches 90% or 99%. The corresponding principle components then form a k-dimension subspace of C', just like in LSI, where dimension reduction is performed via SVD on the matrix AA^T. In LSI, the same variance can be computed in terms of the singular values, which are just the square roots of the eigenvalues of AA^T:

$$\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2},$$

where r is the rank of A. From this, we see that the information space method is very closely related to LSI; the comparison is summarized in Table 3.1.

3.4.3 Cross-Language LSI

Like the generalized vector space model, latent semantic indexing can be applied to cross-language information retrieval, provided that a bilingual document-aligned corpus is available for training. The query-expansion mechanism in both methods can be viewed as a natural way to establish a cross-language semantic correspondence between terms in different languages that mimics the effect of real translations.

Cross-language LSI is very similar to monolingual LSI; it computes the orthogonal dimensions in the parallel training corpus and uses them as the projection matrix for the bilingual semantic space where queries and documents in different languages can be compared [65, 7]. Assuming A and B are the term-document matrices for the training documents in two languages, as in Section 3.3.1, the SVD of the following matrix is computed to build the bilingual space:

$$\begin{bmatrix} A \\ B \end{bmatrix} = U \, \Sigma \, V^T, \tag{3.26}$$

where matrix $[A^T B^T]^T$ is of dimension $(m+p) \times n$. Each column of this matrix combines a pair of parallel documents in two languages into a single document. Similar to the monolingual case, a term, regardless of the language it is in, gets a reduced-dimension vector representation in the rows of $I_k U$, which consist of the first k columns of U. Term similarities are computed by $U I_k U^T$, as in the monolingual case. Let $D'_{(j)}$ and $\vec{q}\,'$ be the vector extension of j-th document $D_{(j)}$ and query $\vec{q}$, respectively, so that they cover terms in both languages:

$$D'_{(j)} = \begin{bmatrix} D_{(j)} \\ 0 \end{bmatrix}, \quad \vec{q}\,' = \begin{bmatrix} 0 \\ \vec{q} \end{bmatrix}.$$

Then, the similarity comparison measure is essentially unchanged from monolingual LSI:

$$\text{Sim}_{LSI-CL} = (I_k U^T D'_{(j)}) \cdot (I_k U^T \vec{q}\,') = (D'_{(j)})^T U I_k U^T \vec{q}\,'. \tag{3.27}$$

Comparing the cross-language LSI formula to that of cross-language GVSM, the training matrix appears to be quite different from what one may anticipate. In GVSM, the cross-language term association matrix is AB^T, which, following the discussion in Section 3.4.1, we would imagine LSI should analyze via SVD. But, AB^T is a very large term-term matrix and also much denser than either A and B. On the other hand, the cross-language LSI formula in Equation 3.27 can be viewed as analyzing the following term association matrix:

$$\begin{bmatrix} A \\ B \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix}^T = \begin{bmatrix} A \\ B \end{bmatrix} [A^T B^T] = \begin{bmatrix} AA^T & AB^T \\ BA^T & BB^T \end{bmatrix};$$

with just one SVD, we are analyzing both the monolingual term-association matrices (AA^T and BB^T) and the cross-lingual ones (AB^T and BA^T). Of course, this is an approximation to a rigorous three-step process that incorporates both monolingual and cross-language term-associations (BB^T, BA^T, and then AA^T), which is the subject of next section.

3.4.4 Procrustes Analysis

The *Procrustes analysis* method developed by Littman et al. [71, 68] performs the "translation" from one language into another explicitly. They first find the SVD's of A and B to obtain the respective reduced-dimension LSI spaces A_k and

B_k:

$$\begin{aligned} A_k &= U\,\Sigma_k\,V^T \\ B_k &= W\,\Omega_k\,X^T. \end{aligned}$$

Since query $\vec{q}$ will be projected into the space of B_k, the translation would require a mapping from B_k to A_k so that $\vec{q}$ can have an appropriate representation in A_k. Applying the solution to the *Procrustes problem* [49], a rotation matrix can be found between the matrices $I_k U^T A = \Sigma_k\,V^T$ and $I_k W^T B = \Omega_k\,X^T$ [1], both of dimension $k \times n$ (n is the number of parallel documents). In particular, let the singular value decomposition of $\Omega_k\,X^T\,V\,\Sigma_k$ (a $k \times k$ matrix) be

$$\Omega_k\,X^T\,V\,\Sigma_k = U'\,\Sigma'\,(V')^T,$$

then $U'\,(V')^T$ is the orthogonal rotation matrix we are looking for (see Appendix B.2). This is equivalent to finding a transformation from B_k to A_k by computing the SVD of the matrix $B_k\,A_k^T$:

$$\begin{aligned} B_k\,A_k^T &= W\,\Omega_k\,X^T\,V\,\Sigma_k\,U^T \\ &= W I_k\,U'\,\Sigma'\,(V')^T I_k\,U^T \\ &= (W I_k\,U')\,\Sigma'\,(U I_k\,V')^T, \end{aligned}$$

where WI_kU' and UI_kV' are unitary matrices. So, $T = WI_k\,U'\,(V')^T\,I_k\,U^T$ is a rotation matrix from B_k to A_k. Now, the query vector $\vec{q}$ can be first projected (query expansion), then rotated (translation), and finally projected again

[1]An alternative approach is to find the rotation matrix between V and X, which is related to the new cross-language LSI formula that I present in Section 3.5.1.

as follows:

$$\begin{aligned}
\vec{q} &\rightarrow W I_k W^T \vec{q} \\
W I_k W^T \vec{q} &\rightarrow T^T W I_k W^T \vec{q} \\
T^T W I_k W^T \vec{q} &\rightarrow U I_k U^T T^T W I_k W^T \vec{q} = U I_k V' (U')^T I_k W^T \vec{q} = T^T \vec{q}.
\end{aligned}$$

In essence, the similarity comparison formula of the Procrustes method is

$$\begin{aligned}
\text{Sim}_{PA-CL}(d, q) &= \vec{d}^T U I_k V' (U')^T I_k W^T \vec{q} \\
&= ((V')^T I_k U^T \vec{d}) \cdot ((U')^T I_k W^T \vec{q}). \qquad (3.28)
\end{aligned}$$

The connection between the Procrustes analysis method and LSI is most evident when we apply the above derivation steps to the case of monolingual retrieval. Appendix B.3 shows the interesting result that monolingual Procrustes analysis is exactly the same as monolingual LSI.

Because the translation step of $B_k A_k^T$ requires an SVD of a probably very dense matrix (in contrast, recall that we only need to find the SVD of A—a sparse matrix—in order to compute the SVD of AA^T), such a three-step process has been carried over only on relatively small collections on the orders of several thousand dimensions [71, 68].

3.4.5 Generalized Singular Value Decomposition

Since cross-language LSI treats each training document pair (the aligned columns of A and B) as a single document, any query or document in only one language will thus be treated like a "half" document. To see this, let $A_{(j)}$ and $B_{(j)}$ be any pair of corresponding documents in the training corpus. Then, since they contain terms in only one language, their vector representations need to be "extended" with zeros to cover the terms in the other language. In fact, $I_k U^T \begin{bmatrix} A_{(j)} \\ 0 \end{bmatrix}$ and $I_k U^T \begin{bmatrix} 0 \\ B_{(j)} \end{bmatrix}$ are the vector representations of $A_{(j)}$ and $B_{(j)}$ in the latent semantic space.

One way to address this issue is to force the two documents to have the same vector representation after SVD. This can be achieved through the generalized version of singular value decomposition (GSVD) on a pair of matrices [49]:

$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} U\Sigma \\ W\Omega \end{bmatrix} V^{-1}, \tag{3.29}$$

where $U \in \Re^{m\times m}$ and $W \in \Re^{p\times p}$ are unitary matrices, $V \in \Re^{n\times n}$ is invertible, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_\text{n})$, and $\Omega = \text{diag}(\omega_1, \omega_2, \ldots, \omega_\text{t})$, where $t = \min(p, n)$.

The GSVD of A and B give rise to a new LSI-like formula for cross-language document comparison:

$$\text{Sim}(d,\, q)_{GSVD-CL} = \vec{d}^T\, U\, \Sigma^{-1}\, \Omega^{-1}\, W^T\, \vec{q} = (\Sigma^{-1} U^T \vec{d})^T\, (\Omega^{-1} W^T \vec{q}). \tag{3.30}$$

Here, the common part (V^{-1}) of the two matrices is extracted, and, in U and W, two sets of orthogonal vectors are constructed to represent terms in the two different languages.

Now, for any pair of training documents $A_{(j)}$ and $B_{(j)}$, their projected images in the GSVD space are

$$\begin{array}{ccccccc} \Sigma^{-1}\, U^T A_{(j)} & = & \Sigma^{-1}\, U^T A e_j & = & \Sigma^{-1}\, U^T U \Sigma V^{-1} e_j & = & V^{-1} e_j \\ \Omega^{-1}\, W^T B_{(j)} & = & \Omega^{-1}\, W^T B e_j & = & \Omega^{-1}\, W^T W \Omega V^{-1} e_j & = & V^{-1} e_j. \end{array}$$

Thus, their mapped images are indeed identical.

Whereas whether we really need to treat parallel documents in different languages as identical remains as an issue, the retrieval performance of GSVD appears to be not much different from that of LSI, which I will briefly present in Section 4.3.2. Like Procrustes analysis, the GSVD algorithm is currently feasible only on small collections because there is no software package for large-scale sparse matrices. I had to use the general LAPACK procedure for GSVD to accomplish the task described in Section 4.3.2.

3.5 Dimension Equalization

All the methods I have discussed so far fall into three categories: the basic term-matching approach of VSM, the methods with a simple utilization of the term-term association matrix, and the others that use SVD to analyze the association matrix. Hence, we have the three basic forms of vector-based method of information retrieval: VSM, GVSM, and LSI. They all have their advantages and disadvantages in terms of computational complexity, storage requirements, and retrieval performance. In this section, I discuss their mathematical relationships to solidify a unified view of those methods and lay the groundwork for the approximation algorithm presented in Chapter 4.

From the previous sections, we see that all three methods do in one way or another conform to Equation 3.1. Their formulae look quite different, however, as Table 3.2 shows.

We can obtain the VSM and LSI similarity formulae from a different perspective, in terms of a special form of the training matrix A. Before I continue, I introduce some new notation.

First, I define the operation of *dimension equalization* as re-composing A with U and V, but giving all columns unit equal weight. This process is shown

	Monolingual	Cross-language
VSM	$(\boldsymbol{I^T\vec{d}})\cdot(\boldsymbol{I^T\vec{q}}) = (\boldsymbol{\vec{d}})\cdot(\boldsymbol{\vec{q}})$	–
GVSM	$(\boldsymbol{A^T\vec{d}})\cdot(\boldsymbol{A^T\vec{q}})$	$(\boldsymbol{A^T\vec{d}})\cdot(\boldsymbol{B^T\vec{q}})$
GVSM with Sparsification	$\mathrm{Sp}_k(A^T\vec{d})\cdot\mathrm{Sp}_k(A^T\vec{q})$	$\mathrm{Sp}_k(A^T\vec{d})\cdot\mathrm{Sp}_k(B^T\vec{q})$
Similarity Thesaurus	$(E\,\vec{d})\cdot(E\,\vec{q})$ where E is like A^T	$(E\,\vec{d})\cdot(F\,\vec{q})$ and F is like B^T
Local Feedback	$\frac{\beta}{n_1}(D^T\vec{d})\cdot(\mathrm{Spn}_{n_1}(D^T\vec{q}))$ $+\alpha(\vec{d})\cdot(\vec{q})$ where $D = A$	$(B^T\vec{d})\cdot(\mathrm{Spn}_{n_1}(A^T\vec{q}))$
LSI	$(\boldsymbol{I_kU^T\vec{d}})\cdot(\boldsymbol{I_kU^T\vec{q}})$ **where** $\boldsymbol{A = U\Sigma V^T}$	$(\boldsymbol{I_kU^T\vec{d}})\cdot(\boldsymbol{I_kU^T\vec{q}})$ **where** $\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{B}\end{bmatrix} = \boldsymbol{U\Sigma V^T}$
Information Space	$(I_k(U')^T\vec{d})\cdot(I_k(U')^T\vec{q})$ where $(U')^TC\,U' = \Sigma$ and C is like AA^T	–
Procrustes Analysis	$(I_kU^T\vec{d})\cdot(I_kU^T\vec{q})$ where $A = U\,\Sigma\,V^T$	$((V')^TI_k\,U^T\vec{d})\cdot((U')^TI_k\,W^T\,\vec{q})$ where $A = U\,\Sigma\,V^T$, $B = W\,\Omega\,X^T$, and $\Omega_k\,X^T\,V\,\Sigma_k = U'\,\Sigma'\,(V')^T$
GSVD	–	$(\Sigma^{-1}U^T\vec{d})\cdot(\Omega^{-1}W^T\vec{q})$ where $\begin{bmatrix}A\\ B\end{bmatrix} = \begin{bmatrix}U\Sigma\\ W\Omega\end{bmatrix}V^{-1}$

Table 3.2: The three conventional vector-based information retrieval methods (shown in boldface) and their close relatives all use a projection matrix, but they look wildly different from the surface.

in Figure 3.2 and a formal definition is given below.

Notation 3.5. *Suppose* $A = U\,\Sigma\,V^T$. *Then the* equalized-dimension *form of* A *is*

$$\bar{A} = U\ \mathrm{diag}(1, 1, \ldots, 1)\ V^T = U I\, V^T = UV^T.$$

In addition, recall that the reduced-dimension form of A is

$$A_k = U\,\Sigma_k\,V^T.$$

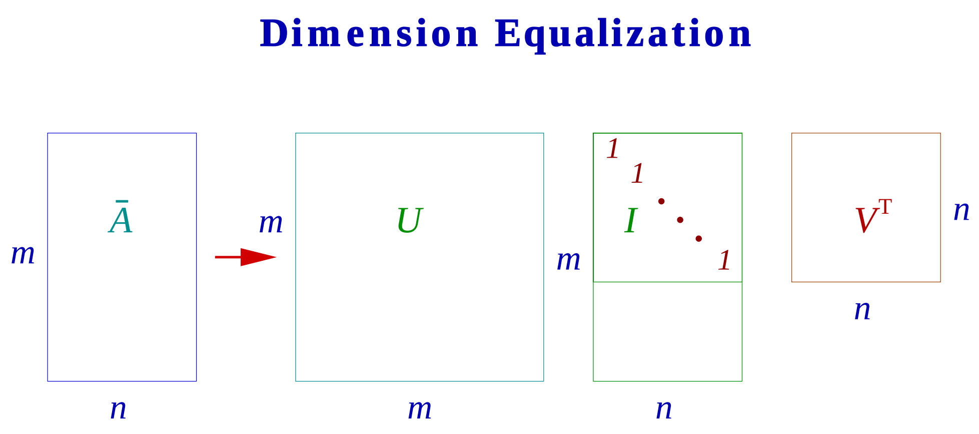

Figure 3.2: The dimension equalization of a matrix A involves replacing the singular value matrix Σ by the identity matrix I.

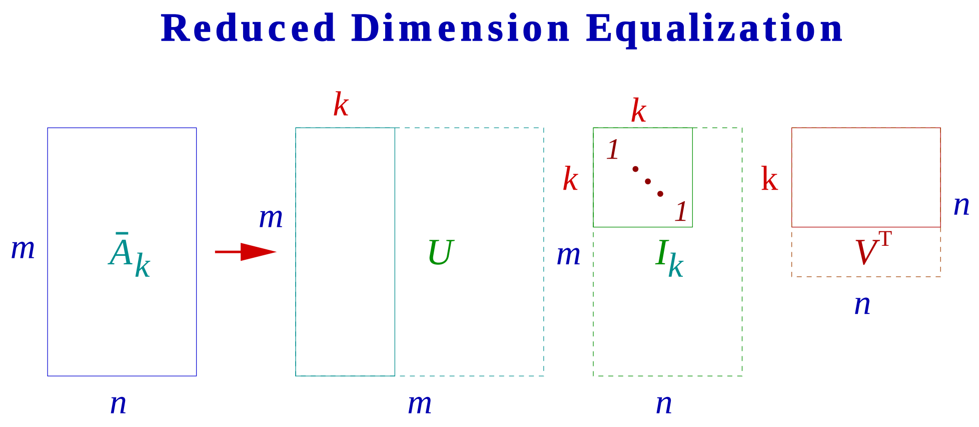

Figure 3.3: The reduced form of a dimension-equalized matrix $\bar{A}$ involves replacing the identity matrix I by I_k, where $k \leq r$ (the rank of the original matrix A).

Then, the reduced-dimension form of $\bar{A}$ is

$$\bar{A}_k = U\,\mathrm{Spn}_k(\Sigma)\,V^T = U I_k V^T = U\,\mathrm{diag}(1, 1, \ldots, 1, 0, 0, \ldots, 0)\,V^T.$$

This is illustrated in Figure 3.3.

Now, we are ready to see formulations of LSI and VSM that look similar to that of GVSM.

Formulation	Equation	Interpretation
$(I_k U^T \vec{d}) \cdot (I_k U^T \vec{q})$	3.19	fold in the document and the query in the latent semantic space
$(U I_k U^T \vec{d}) \cdot (U I_k U^T \vec{q})$	3.21	projecting the document and the query onto the range of reduced-dimension space A_k
$(\vec{d}) \cdot (U I_k U^T \vec{q})$	3.23	expand the query using the first k dimensions of A before comparing to the document
$(\bar{A}_k^T \vec{d}) \cdot (\bar{A}_k^T \vec{q})$	3.31	using the reduced dimension equalization form of A as the transformation matrix

Table 3.3: Similarity comparison of LSI can be formulated in four different ways with different interpretations.

3.5.1 LSI Revisited

Through the following derivation, we see that $\bar{A}_k$ is also a transformation matrix for LSI:

$$\begin{aligned}(\bar{A}_k^T \vec{d}) \cdot (\bar{A}_k^T \vec{q}) &= (V I_k U^T \vec{d}) \cdot (V I_k U^T \vec{q}) \\ &= \vec{d}^T U I_k V^T V I_k U^T \vec{q} \\ &= \vec{d}^T U I_k U^T \vec{q} \\ &= \text{Sim}_{LSI-ML}(d, q). \qquad (3.31)\end{aligned}$$

So far, I have given four different formulations of LSI similarity comparison. They are summarized in Table 3.3.

The new LSI formula of Equation 3.31 looks very much like the GVSM one in Equation 3.10, except for what is in the middle: while the GVSM transformation matrix A has its natural diagonal matrix intact, the LSI matrix $\bar{A}_k$ replaces it with an identity matrix up to k dimensions. It should be noted that the difference is the squares of diagonal elements and the number of dimensions:

$$\begin{aligned}
\mathrm{Sim}_{GVSM-ML}(d,\, q) &= \vec{d}^T A A^T \vec{q} \\
&= \vec{d}^T (U\,\Sigma\, V^T)(U\,\Sigma\, V^T)^T \vec{q} \\
&= \vec{d}^T U \Sigma\, V^T V \Sigma\, U^T \vec{q} \\
&= \vec{d}^T U \Sigma^2\, U^T \vec{q} \\
&= (\Sigma U \vec{d}\,) \cdot (\Sigma U \vec{q}\,).
\end{aligned}$$

Comparing this last formula to the similarity measure of LSI in Equation 3.19, we see that GVSM is a weighted (by the squares of singular values) version of full-rank LSI.

Now, for cross-language retrieval, recall that GVSM uses the following formula (Equation 3.10):

$$\mathrm{Sim}_{GVSM-CL}(d,\, q) = (A^T \vec{d}\,) \cdot (B^T \vec{q}\,),$$

with A being replaced by its translational counterpart B on the query side. For cross-language LSI, we could do a similar thing, simply substituting $\bar{B}_k$ for $\bar{A}_k$ on the query side:

$$\mathrm{Sim}_{LSI-CL}(d,\, q) = (\bar{A}_k^T \vec{d}\,) \cdot (\bar{B}_k^T \vec{q}\,). \tag{3.32}$$

This is quite different from the original cross-language LSI formulation by Landauer and Littman [65] as described in Section 3.4.3. Compared to this traditional cross-language LSI, my new approach computes two separate SVD's of smaller matrices instead. This is certainly more useful when the combined matrix becomes too large to analyze via SVD.

When $k = n$, the number of documents in the training collection, for k in

Equation 3.32, this new cross-language LSI is a special case of the Procrustes method discussed in Section 3.4.4 (see Appendix B.4 for a derivation).

3.5.2 VSM Revisited

As shown in Equations 3.10, 3.12, 3.31, and 3.32, GVSM and LSI use forms of the training matrices A and B for projection. VSM can be expressed in a similar fashion as well.

To see this, I first examine the relationship between VSM and LSI. In monolingual LSI retrieval, training is usually done directly on the retrieval collection. This means $A = D$, where A is the matrix for the training collection and D the matrix for the retrieval collection. In addition, if all the orthogonal dimensions of D are used in LSI ($k = r$, the rank of D), then Equation 3.19 becomes

$$\begin{aligned}
\mathrm{Sim}_{LSI-ML}(d, q) &= (I_k U^T D_{(j)}) \cdot (I_k U^T \vec{q}) && \text{(since } A = D\text{)} \\
&= (U^T D e_j) \cdot (U^T \vec{q}) && \text{(since } k = r\text{)} \\
&= (U^T U \Sigma V^T e_j) \cdot (U^T \vec{q}) \\
&= (\Sigma V^T e_j) \cdot (U^T \vec{q}) \\
&= e_j^T V \Sigma U^T \vec{q} \\
&= D_{(j)}^T \vec{q} \\
&= (D_{(j)}) \cdot (\vec{q}) \\
&= \mathrm{Sim}_{VSM}(d, q). && (3.33)
\end{aligned}$$

In words, if the full rank of the retrieval matrix is used for training, monolingual LSI turns into VSM.

It then follows that since LSI uses $\bar{A}_k$ as the transformation matrix, VSM uses

	Monolingual	Cross-language
GVSM	$(A^T\vec{d})\cdot(A^T\vec{q})$	$(A^T\vec{d})\cdot(B^T\vec{q})$
VSM	$(\bar{A}^T\vec{d})\cdot(\bar{A}^T\vec{q})$	$(\bar{A}^T\vec{d})\cdot(\bar{B}^T\vec{q})$
LSI	$(\bar{A}_k^T\vec{d})\cdot(\bar{A}_k^T\vec{q})$	$(\bar{A}_k^T\vec{d})\cdot(\bar{B}_k^T\vec{q})$

Table 3.4: VSM, GVSM, and LSI expressed in terms of the different forms of the training matrices A and B.

$\bar{A} = \bar{D}$ with no dimension reduction at all. Hence, we have

$$\begin{aligned}
(\bar{A}^T D_{(j)}) \cdot (\bar{A}^T\vec{q}) &= (VU^T D_{(j)}) \cdot (VU^T\vec{q}) \\
&= (VU^T U\,\Sigma\,V^T e_j) \cdot (VU^T\vec{q}) \\
&= (V\,\Sigma\,V^T e_j) \cdot (VU^T\vec{q}) \\
&= e_j^T\,V\,\Sigma\,V^T V U^T\vec{q} \\
&= e_j^T\,V\,\Sigma\,U^T\vec{q} \\
&= D_{(j)}^T\,\vec{q}
\end{aligned}$$

In other words, in VSM the projection matrix $P = \bar{A}$. Thus, all three methods can be expressed in a normal or special form of A, as summarized in Table 3.4.

Note that in the cell for cross-language VSM, the formula $(\bar{A}^T\vec{d})\cdot(\bar{B}^T\vec{q})$ is not exactly VSM but full dimensional LSI. The requirements for LSI to be VSM include not only full dimensionality, but also training on the retrieval collection. If we did have a parallel corpus for retrieval, then a trivial extension of the monolingual VSM would do the cross-language retrieval job: do monolingual retrieval on the documents that are in the language of the query, and then return their aligned counterparts in the other language.

3.5.3 Comparison of Singular Values

If the singular values of the training matrix A were all the same, then the formulae for GVSM and VSM in Table 3.4 would give the same scores for the same query against a document collection (i.e. same ranking). Similarly, LSI and VSM are the same if the dimensionality k is chosen equal to the rank r of A. Therefore, the differences we observe between VSM, GVSM, and LSI in practice depend on the distribution of the singular values of the training matrix.

Figure 3.4 shows the singular values of the small 1,121 UNICEF English test documents used by Carbonell et al. [26]. The collection is indexed with the SMART *ntc* weighting scheme and a complete spectrum of 1,103 singular values is easily found from the matrix. The characteristic of this plot is that its corresponding matrix possesses the low-rank-plus-shift structure, reported by Zha and Zhang in their study of LSI [124, 123] (see Section 2.4.3): the singular values are relatively large but decrease sharply at the beginning, level off noticeably for the most part in the middle, and dip again at the end (due to rank-deficiency [124]). Since the singular values indicate how important their corresponding singular vectors (dimensions) are in the term-document space, the special shape of their distribution plot does show that the vector space has a few very dominant dimensions (hence the "low rank"), plus a wide range of mostly identical dimensions (like "shifting" or multiplying the identity matrix by some constant value).

I have plotted the singular values of a number of corpora, varying language, size, and indexing scheme, and all seem to have this special property. Figure 3.5 shows four such plots of some relatively small document collections (with at most 10,000 documents) on which a complete SVD can be computed using current hardware and software. The detailed information of these four corpora are listed

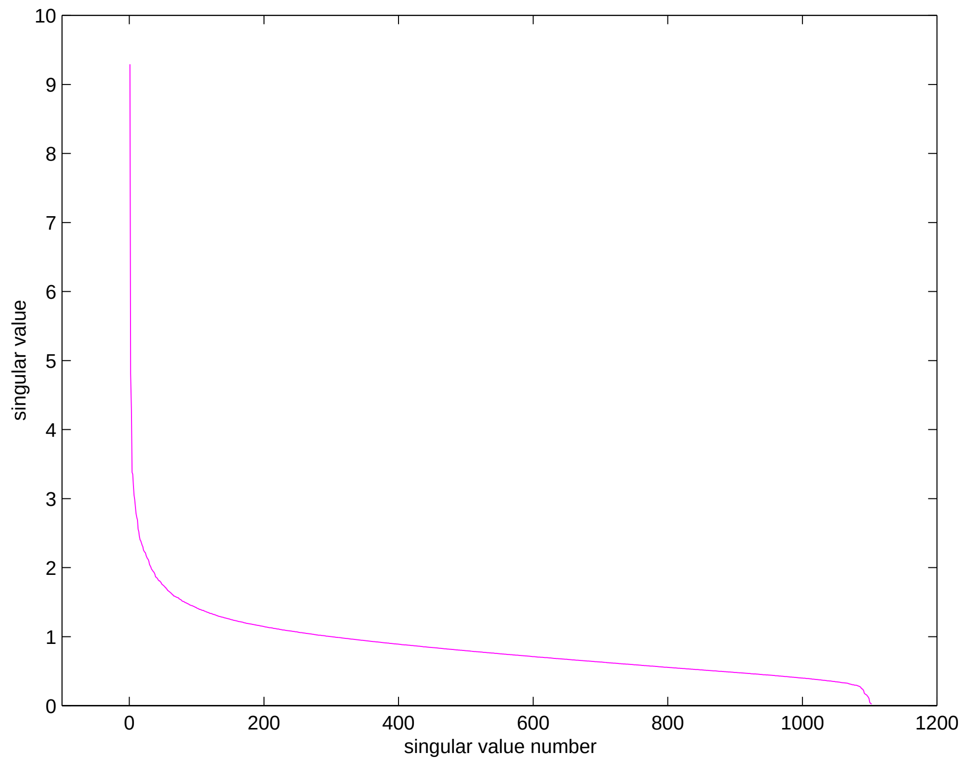

Figure 3.4: Singular value distribution of the matrix of the CMU UNICEF English test collection shows the its low-rank-plus-shift structure.

in Table 3.5 on the same page as the plots.

When the matrix size gets too large to compute the complete spectrum using current computing resources, we still see the trend of initial dropping and leveling off of singular values. Plotted on Figure 3.6 are the first 480 singular values of the TREC French collection of 141,643 documents with Okapi term weighting. More plots of other large corpora are shown in Figure 3.7, and their information is listed in Table 3.7.

What does this imply for the behavior of the various vector-based IR methods? All three projection matrices (A, $\bar{A}$, and $\bar{A}_k$) have the same left and right singular vectors (U and V) and only differ in how they weight these vectors (Σ, I, and I_k). So, for all three methods, $P = U\,\Phi\,V^T$ using Equation 3.1, where Φ is a diagonal

Collection	Language	Subject	Documents	Terms	Weighting
Medlars	English	Medicine	1,033	7,014	SMART *ntc*
Cranfield	English	Aerodynamics	1,400	3,763	SMART *ntc*
CMU UNICEF	Spanish	Politics	1,134	15,343	SMART *ntc*
TREC AP 1990	English	News	10,000	47,887	SMART *ntc*

Table 3.5: Characteristics of four small document collections show that they are very diverse.

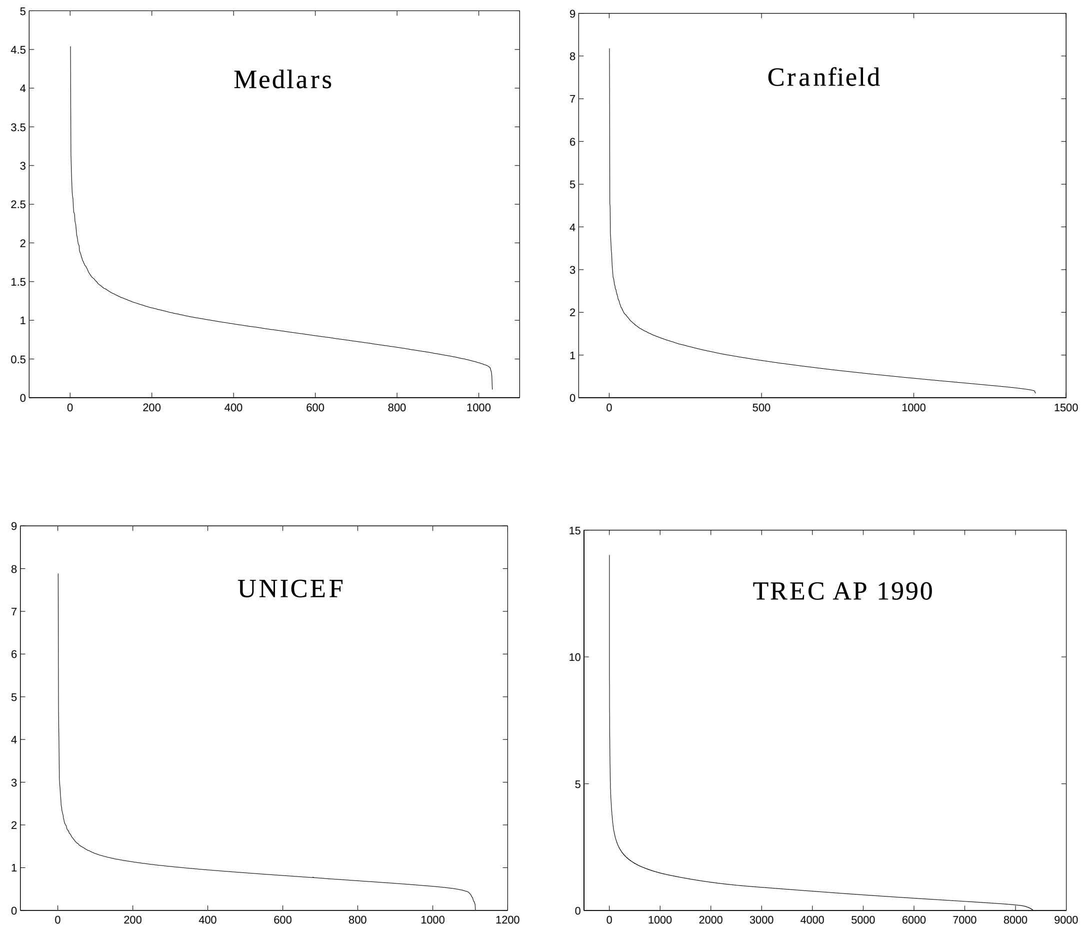

Figure 3.5: Singular value distributions of the matrices of the Medlars, Cranfield, UNICEF, and TREC AP 1990 collections all show the low-rank-plus-shift structure.

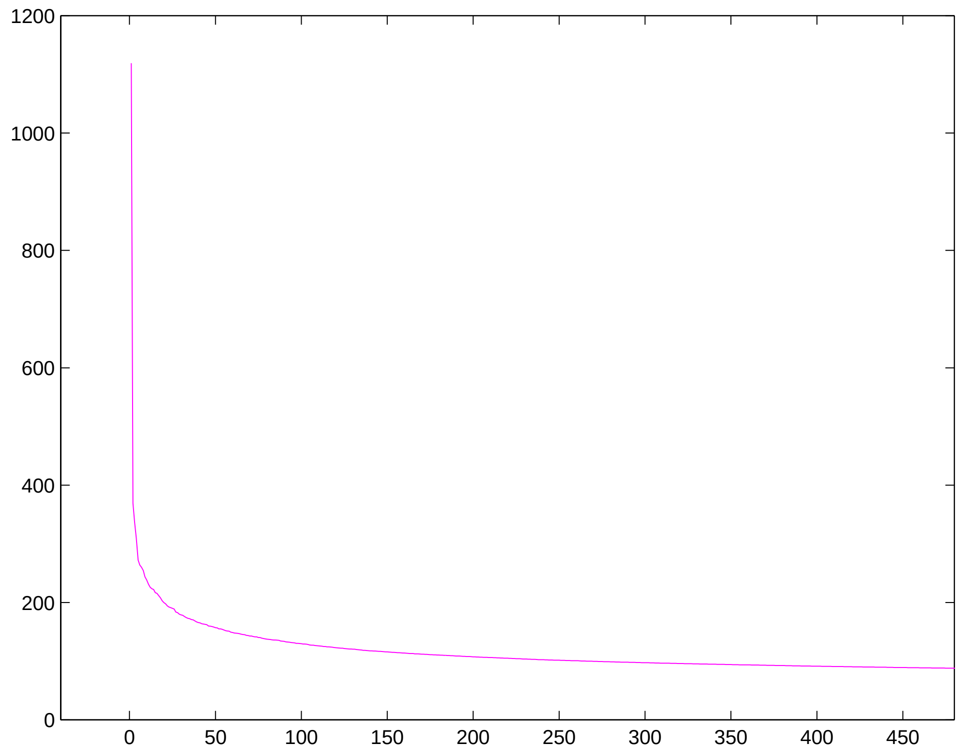

Figure 3.6: Distribution of the first 480 singular values of the TREC French collection matrix still show the initial part of its low-rank-plus=shift structure.

matrix of *dimension weights*. The difference between the three methods in terms of Φ is summarized in Table 3.7. According to the distributions of real singular values, we see that the projection P in GVSM is skewed by Σ, while the one in LSI is filtered by a Heaviside function. The relative dimension weights or normalized singular value distributions of the three vector-based methods are illustrated by the solid lines in Figure 3.8 (the dashed lines will be explained shortly). While VSM and LSI equally apply the orthogonal dimensions they use, GVSM seems to give overwhelming emphasis to the initial ones. This makes GVSM look like a very low-dimension LSI, since the effect of all dimensions except the first few is negligible. VSM is the other extreme since it is a full dimensional version of LSI.

A natural question to ask at this point is, what is the "right" value of Φ to

Collection	Language	Subject/Type	Documents	Terms	Weighting
TASA	English	Highschool Readings	37,651	74,729	Similarity Thesaurus
NACSIS	Japanese	Academic Paper Abstracts	45,372	33,553	Similarity Thesaurus
TREC	French	News	141,643	196,878	SMART *ntc*
Hansard	French	Canadian Parliament Proceedings	227,344	102,078	SMART *ntc*

Table 3.6: Characteristics of four large document collections show that they are quite different.

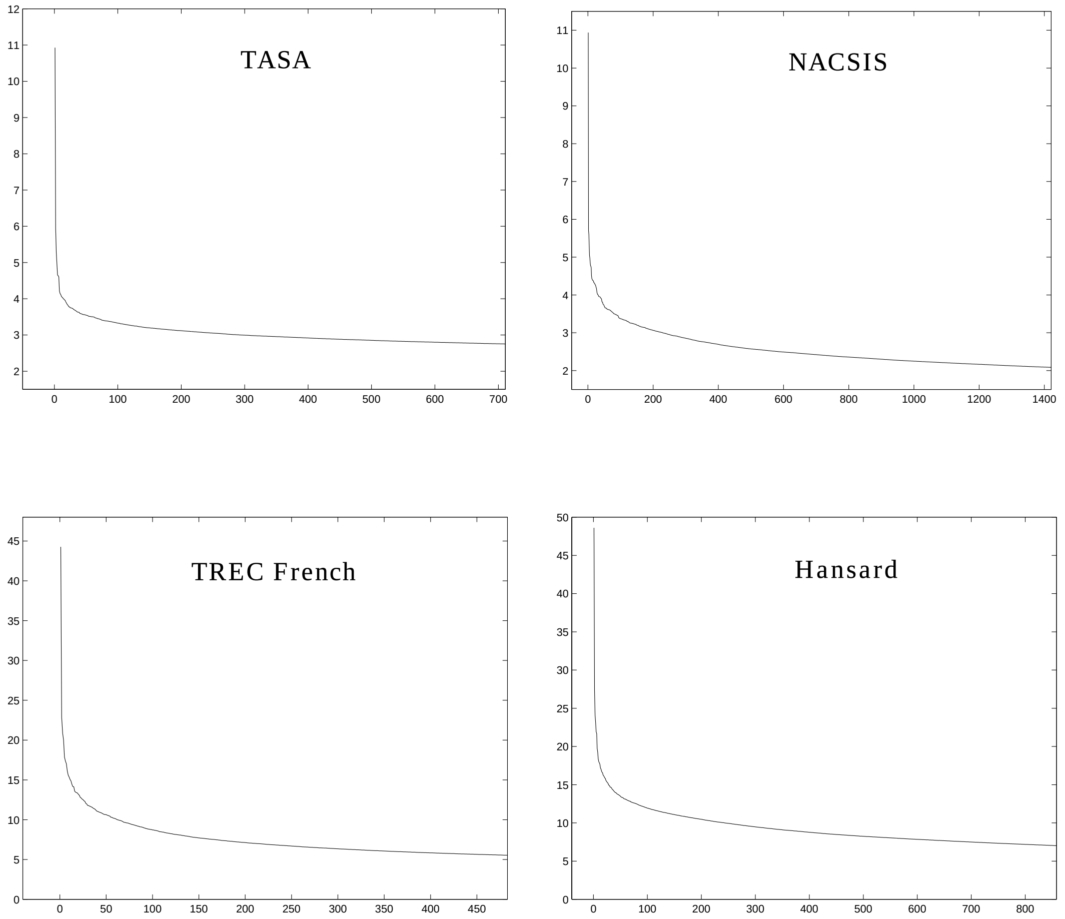

Figure 3.7: Initial singular value distributions of the TASA, NACSIS, TREC French, and Hansard collection matrices show they all possess the common low-rank-plus-shift structure.

	Monolingual	Cross-language
	$A = U\,\Sigma\,V^T$ $P = U\,\Phi\,V^T$ $\text{Sim}(d,\,q) = (P^T\vec{d})\cdot(P^T\vec{q})$	$A = U\,\Sigma\,V^T$, $B = W\,\Omega\,X^T$ $P = U\,\Phi\,V^T$, $Q = W\,\Psi\,X^T$ $\text{Sim}(d,\,q) = (P^T\vec{d})\cdot(Q^T\vec{q})$
GVSM	$\Phi = \Sigma$	$\Phi = \Sigma$, $\Psi = \Omega$
VSM	$\Phi = I$	$\Phi = I$, $\Psi = I$
LSI	$\Phi = I_k$	$\Phi = I_k$, $\Psi = I_k$

Table 3.7: VSM, GVSM, and LSI differ only in the diagonal elements of their respective projection matrices.

use? In some sense, the answer is whichever gives the best empirical results. But, we can make a motivated prediction based on linear algebra. Ideally, the transformation matrix P should project queries and documents into the row space of the training matrix, since this is our source of information on term-term relatedness. However, the least significant dimensions of the row space (those corresponding to the smallest singular values) are likely to be modeling "noise"—variability in term usage that does not correspond to a significant change in meaning. Thus, we might want to project documents and queries into all but the least significant dimensions of the row space. This truncation is achieved by setting $\Phi = I_{k^*}$, where k^* is the optimal number of dimensions to include.

Empirically, studies using smaller collections have shown that best retrieval performance can indeed be achieved with reduced but equally weighted dimensions of the training matrix [39, 36, 70]. With larger collections, however, the small number of dimensions that can be calculated does not appear to be enough to demonstrate the utility of dimension reduction with equalization. In other words, by extrapolating from studies on the smaller collections, I predict the ideal k^* is lying there somewhere beyond what can be computed using current computing resources. This is depicted as a dashed line in Figure 3.8.

While none of VSM, GVSM, and LSI match the singular values of the idealized

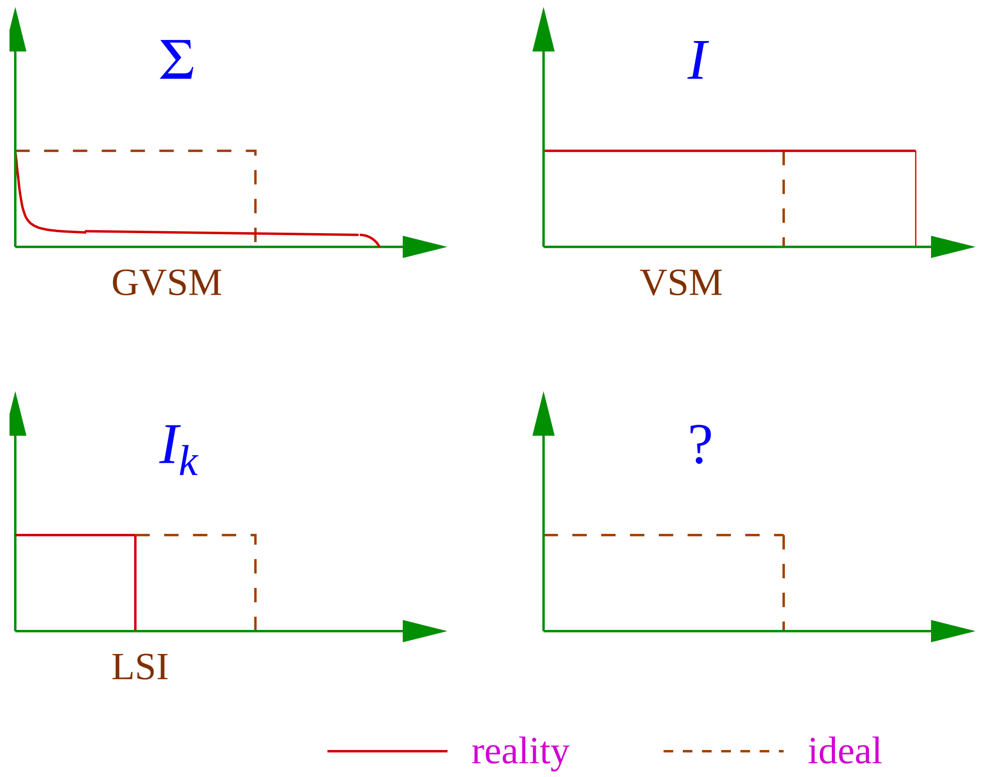

Figure 3.8: The three basic vector-based methods apply quite different relative weights to the orthogonal dimensions of the training matrices that they use for projection.

transformation matrix, the following lemma is useful in judging which gives the best approximation.

Lemma 3.1. *Let A be a monolingual training corpus with singular value decomposition $U\Sigma V^T$. Let $P = U\Phi V^T$ and $Q = U\Psi V^T$ be two transformation matrices that share left and right singular vectors with A. Compare the matrices of similarities obtained by comparing all pairs of documents in A to each other using* Sim_P *and* Sim_Q*. The Frobenius norm of* $\mathrm{Sim}_P(A, A) - \mathrm{Sim}_Q(A, A)$ *is equal to the Frobenius norm of* $\Sigma^2(\Phi^2 - \Psi^2)^2$*.*

A proof of this lemma can be found in Appendix B.5. What it tells us is that the closer two transformation matrices are in their singular values, the closer are the similarities they produce. Further, the dimensions that matter the most are the ones that correspond to the largest singular values in the training matrix A.

Based on this, we would expect VSM to produce the most accurate similarity scores, followed by LSI, which is accurate for the dimensions with the highest singular values. However, VSM cannot be used directly for cross-language retrieval. In the next chapter, I develop a novel approximation to the idealized transformation matrix, which will fill in the question mark in Figure 3.8.

Chapter 4

Approximate Dimension Equalization

As I have described in last chapter, VSM, GVSM, and LSI are closely related vector-based IR methods each with its own unique strengths. VSM is simple, scales extremely well, and gives excellent performance on large text collections, but it cannot be used for cross-language retrieval without applying language-specific knowledge or machine translation techniques [20, 119]. LSI extends VSM by obtaining a reduced-dimension representation that models term-term associations, thus allowing a query to have a positive similarity to a document with which it shares no terms. This is especially important in cross-language IR applications. While LSI has shown impressive performance on some text collections, its performance on extremely large and diverse text collections has lagged behind that of VSM. I have presented some evidence that this is because of the large number of dimensions needed from the SVD for large text collections. Although SVD is used as a preprocessing step in retrieval by LSI, the large size of modern information collections has made full SVD computations less and less feasible. GVSM also models term-term associations, and scales much more easily than LSI. Unfortunately, its performance also lags substantially behind that of VSM and often LSI. I argued that this is because, although GVSM uses more dimensions than LSI, it puts tremendous weight on the largest ones, resulting in an effective dimensionality that is substantially smaller.

In this chapter, I show how to combine ideas from VSM, GVSM, and LSI to obtain a vector representation that approximates the effect of dimension equalization. This approximation algorithm is based on the consistent pattern we observe from the distribution plots of singular values of many text collections, which I discuss in detail in Section 4.1. Then, in the sections that follow, I will formally introduce the algorithmic procedure of this approximation method and show experimental results applying the approximation technique on a variety of retrieval collections.

4.1 Motivation

In Sections 3.5.1 and 3.5.2, I explained the analytical differences between VSM, GVSM, and LSI in terms of dimension equalization of the projection matrix P. All three methods use the same left and right singular vectors from the training matrix but with different weights in the middle. Then, in Section 3.5.3, I provided the singular value distributions of a number of document collections that consistently demonstrated the low-rank-plus-shift structures of the corresponding matrices. This helps to clarify the difference: GVSM is very low dimensional (approximately), LSI reduced dimensional, and VSM full dimensional.

Retrieval performance of these methods has been compared by researchers between VSM and GVSM [118, 117], between VSM and LSI [39, 32, 36, 37], and among VSM, GVSM, and LSI [26, 121, 70]. A mixture of results has been reported, with GVSM and LSI outperforming VSM on smaller collection and LSI almost always having a better retrieval result than GVSM on the same corpus. A notable exception is Yang et al. [121], who found GVSM to be more effective, especially in cross-language retrieval, than LSI. Our attempt to replicate these results were not successful [70].

In recent years, however, LSI has exhibited second-rate performance in mono-lingual retrieval on large collections compared to VSM. The crucial reason for this under-achievement of LSI is that, as the modern text collections grow larger, the large-dimensional SVD has become more difficult to compute. For corpora that consist of hundreds of thousands of documents, only about one or two percent of the orthogonal dimensions of their matrices can be calculated with present resources. The hypothesis is that although dimension reduction is a fundamental property of LSI, abandoning 95% or more of the dimensions (see, for example, experiments described in Section 4.3) oversimplifies the vast, complex structure of the semantic space represented by the large number of documents. As we see from the results presented later in Section 4.3.2, where I am able to obtain all dimensions of the space of the relatively small UNICEF collection, the best LSI performance is reached at 55% of the full dimension of the training matrix. This also explains why GVSM works as well as other methods on small collections. When collection size is small, the small number of dimensions that dominate in the GVSM transformation matrix would comprise a large portion of the complete dimensionality, making it closer to LSI than it is on larger corpora.

From these observations, I conclude that we need a certain fraction of the orthogonal dimensions of the training matrix for productive retrieval. We see that too few dimensions do not capture enough information while too many, in the case of VSM, can provide good retrieval performance. But, in the latter case, we lose the ability of deriving term-term associations. Hence, getting this proportion of dimensions, which varies over corpora, as well as using them equally is a key to effective retrieval. What we need is an approximation method that achieves this with only limited computing power. The current resources are the SVD package from LSI, which can compute a limited number of singular vectors and values, and GVSM, which essentially uses all the singular vectors, but in a way that over-emphasizes the first few singular vectors.

Upon re-examining the plots in Figures 3.4, 3.5, 3.6, and 3.7, I realized I can take advantage of the special low-rank-plus-shift structure of the singular values. From these figures, I speculate that

1. the relatively flat singular values in the middle (the "shift" part) can probably still be used to weight the corresponding singular vectors (as in the GVSM case) without too much deviation from the situation in dimension equalization, where 1 is the weight on all available singular vectors,

2. some work needs to be done on the large singular values at the beginning, which are computable using the current resources, to decrease the importance level of their corresponding vectors to that of the middle vectors, and

3. the singular values at the end that correspond to the "dip" should not concern us because we are dropping the last singular vectors in dimension reduction anyway.

These points combined would result in a method that utilizes the computations used in both LSI and GVSM. The key to this algorithm lies in the fact that the middle singular vectors in the original matrix are weighted close to evenly with small differences between the corresponding consecutive singular values. This gradual downgrading of singular vectors with singular values may not be too detrimental at all, since each next dimension may be less important than the current one. On the other hand, as we are still giving weights to the middle orthogonal dimensions, lowering down the importance of initial singular vectors would mostly mean assigning them weights close to the middle singular values.

4.2 Algorithm

Recall from Chapter 3, A_k is the reduced-dimension form of the training matrix A:

$$A_k = U\,\mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k, 0, \ldots, 0)\,V^T = U\,\Sigma_k\,V^T,$$

$\bar{A}$ is the equalized dimensional form of A:

$$\bar{A} = U\,\mathrm{diag}(1, 1, \ldots, 1)\,V^T = UI\,V^T = UV^T,$$

and $\bar{A}_k$ is the reduced-dimension form of $\bar{A}$:

$$\bar{A}_k = U\,\mathrm{diag}(1, 1, \ldots, 1, 0, 0, \ldots, 0)\,V^T = UI_k\,V^T.$$

Now, define

$$\tilde{A}_k = \bar{A}_k + \frac{1}{\sigma_k}A - \frac{1}{\sigma_k}A_k.$$

$\tilde{A}_k$ has the same left and right singular vectors of A, and consists of the first k equalized dimensions of A and the remaining $r-k$ dimensions of A normalized by σ_k. In terms of the projection matrix $P = U\,\Phi\,V^T$ as discussed in Section 3.5.3, the dimension weights Φ is

$$\tilde{I}_k = I_k + \frac{1}{\sigma_k}\Sigma - \frac{1}{\sigma_k}\Sigma_k,$$

where $\tilde{A}_k = U\,\tilde{I}_k\,V^T$ as matrix multiplication is associative. The process of creating $\tilde{I}_k$ is illustrated in Figure 4.1. The normalization $1/\sigma_k$ is there to down-weight the rest of the singular values (σ_{k+1}, ... , σ_r) because the first k singular values (σ_1, ... , σ_k) are changed to the value of one.

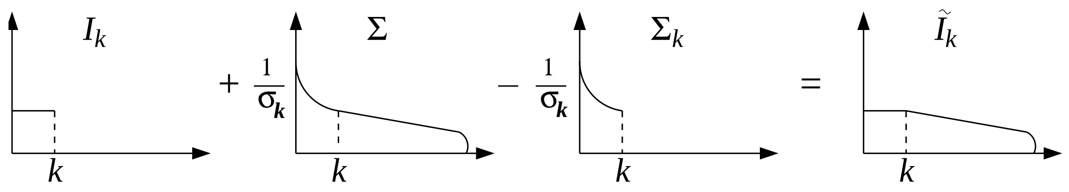

Figure 4.1: Approximate dimension equalization: using the dimension weights of $\bar{A}_k$ (I_k), A (Σ), and $\bar{A}$ (Σ_k) to form those of $\tilde{A}_k$ ($\tilde{I}_k$).

This new matrix $\tilde{A}_k$ takes advantage of the special shape of the singular value plots of all the term-documents we have seen; it flattens out the first k very large singular values, and attaches the rest of the real singular values, which are a relatively level and long middle portion with a small dipping tail. This way, we obtain relatively equalized dimensions of the training matrix until close to the end of all dimensions. Now, for monolingual retrieval, we simply calculate

$$\text{Sim}_{ADE-ML}(d,q) = (\tilde{A}_k^T \vec{d}) \cdot (\tilde{A}_k^T \vec{q}),$$

and for cross-language retrieval, we replace the $\tilde{A}_k$ in front of $\vec{q}$ by $\tilde{B}_k$:

$$\text{Sim}_{ADE-CL}(d,q) = (\tilde{A}_k^T \vec{d}) \cdot (\tilde{B}_k^T \vec{q}).$$

I call this approach *approximate dimension equalization* (*ADE*). The big question mark in Figure 3.8 can now be replaced by the dimension weights of the ADE method. The new drawing is shown in Figure 4.2, and the analytical difference between the four method is compared in Table 4.1.

It is clear that ADE approximates the ideal singular values better than either LSI or GVSM. From one perspective, ADE is trying to extend the limited ability of LSI to compute the singular vectors and values of a large training matrix by implicitly adding additional ones with relatively equal weights. From another, ADE makes cross-language VSM possible, obtaining most if not all the dimensions of both training matrices with equalization. From a third, it uses GVSM's approach

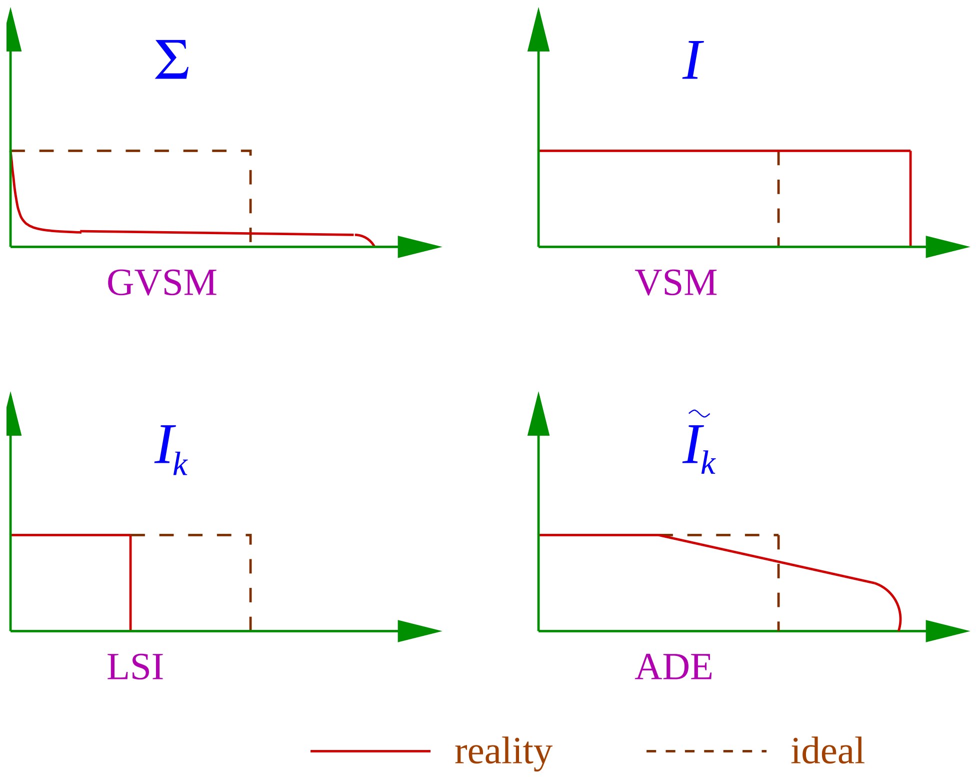

Figure 4.2: Weights applied to the orthogonal dimensions of their respective projection matrices by VSM, GVSM, LSI, and ADE.

to scalably capturing term-term correlations, modified to prevent overemphasizing the first handful of dimensions. In a sense, the projection used by ADE is better than LSI on large matrices because in LSI, truncating dimensions at the largest number we can compute is completely unjustified by the data. ADE, on the other hand, uses all the available dimensions in a way that is supported by the data. The four vector-based methods, VSM, GVSM, LSI, and ADE, expressed in a normal or special form of A are summarized in Table 4.2.

Computationally, ADE only needs to compute the first k dimensions of the training matrix, much like a k-dimension LSI. Moreover, in practical applications, the ADE similarity formula is simpler to compute than one would imagine. Notice

	Monolingual	Cross-language
	$A = U\,\Sigma\,V^T$ $P = U\,\Phi\,V^T$ $\mathrm{Sim}(d,\,q) = (P^T\vec{d}\,)\cdot(P^T\vec{q}\,)$ $\tilde{I}_k = I_k + \frac{1}{\sigma_k}\Sigma - \frac{1}{\sigma_k}\Sigma_k$	$A = U\,\Sigma\,V^T$, $B = W\,\Omega\,X^T$ $P = U\,\Phi\,V^T$, $Q = W\,\Psi\,X^T$ $\mathrm{Sim}(d,\,q) = (P^T\vec{d}\,)\cdot(Q^T\vec{q}\,)$ $\tilde{I}_k^A = I_k + \frac{1}{\sigma_k}\Sigma - \frac{1}{\sigma_k}\Sigma_k$, $\tilde{I}_k^B = I_k + \frac{1}{\omega_k}\Omega - \frac{1}{\omega_k}\Omega_k$
GVSM	$\Phi = \Sigma$	$\Phi = \Sigma$, $\Psi = \Omega$
VSM	$\Phi = I$	$\Phi = I$, $\Psi = I$
LSI	$\Phi = I_k$	$\Phi = I_k$, $\Psi = I_k$
ADE	$\Phi = \tilde{I}_k$	$\Phi = \tilde{I}_k^A$, $\Psi = \tilde{I}_k^B$

Table 4.1: VSM, GVSM, LSI, and ADE differ in the diagonal elements of their respective projection matrices.

	Monolingual	Cross-language
GVSM	$(A^T\vec{d})\cdot(A^T\vec{q})$	$(A^T\vec{d})\cdot(B^T\vec{q})$
VSM	$(\bar{A}^T\vec{d})\cdot(\bar{A}^T\vec{q})$	$(\bar{A}^T\vec{d})\cdot(\bar{B}^T\vec{q})$
LSI	$(\bar{A}_k^T\vec{d})\cdot(\bar{A}_k^T\vec{q})$	$(\bar{A}_k^T\vec{d})\cdot(\bar{B}_k^T\vec{q})$
ADE	$(\tilde{A}_k^T\vec{d})\cdot(\tilde{A}_k^T\vec{q})$	$(\tilde{A}_k^T\vec{d})\cdot(\tilde{B}_k^T\vec{q})$

Table 4.2: VSM, GVSM, LSI, and ADE expressed in terms of the different forms of the training matrices A and B.

that

$$\mathrm{Sim}_{ADE-ML}(d,q) = (\tilde{A}_k^T\vec{d}\,)\cdot(\tilde{A}_k^T\vec{q}\,) = \vec{d}^T\tilde{A}_k\tilde{A}_k^T\vec{q},$$

and the matrix $\tilde{A}_k\tilde{A}_k^T$ actually is not a complicated product once items cancel out:

$$\begin{aligned}
\tilde{A}_k \tilde{A}_k^T &= (\bar{A}_k + \frac{1}{\sigma_k} A - \frac{1}{\sigma_k} A_k)\,(\bar{A}_k + \frac{1}{\sigma_k} A - \frac{1}{\sigma_k} A_k)^T \\
&= (U I_k V^T + \frac{1}{\sigma_k} U \Sigma V^T - \frac{1}{\sigma_k} U \Sigma_k V^T)\,(V I_k U^T + \frac{1}{\sigma_k} V \Sigma U^T - \frac{1}{\sigma_k} V \Sigma_k U^T) \\
&= U I_k U^T + \frac{1}{\sigma_k} U \Sigma_k U^T - \frac{1}{\sigma_k} U \Sigma_k U^T + \frac{1}{\sigma_k} U \Sigma_k U^T + \frac{1}{\sigma_k^2} U \Sigma^2 U^T \\
&\quad - \frac{1}{\sigma_k^2} U \Sigma_k^2 U^T - \frac{1}{\sigma_k} U \Sigma_k U^T - \frac{1}{\sigma_k^2} U \Sigma_k^2 U^T + \frac{1}{\sigma_k^2} U \Sigma_k^2 U^T \\
&= U I_k U^T + \frac{1}{\sigma_k^2} U \Sigma^2 U^T - \frac{1}{\sigma_k^2} U \Sigma_k^2 U^T.
\end{aligned}$$

Now the ADE retrieval becomes quite straightforward:

$$\begin{aligned}
\text{Sim}_{ADE-ML}(d,q) &= \vec{d}^T \tilde{A}_k \tilde{A}_k^T \vec{q} \\
&= \vec{d}^T U I_k U^T \vec{q} + \frac{1}{\sigma_k^2} \vec{d}^T U \Sigma^2 U^T \vec{q} - \frac{1}{\sigma_k^2} \vec{d}^T U \Sigma_k^2 U^T \vec{q} \\
&= \text{Sim}_{LSI-ML}(d,q) + \frac{1}{\sigma_k^2}\, \text{Sim}_{GVSM-ML}(d,q) - \frac{1}{\sigma_k^2}\, \vec{d}^T U \Sigma_k^2\, U^T \vec{q}
\end{aligned}$$

Note that at full dimensionality, $\vec{d}^T U\, \Sigma_k^2\, U^T \vec{q} = \text{Sim}_{GVSM-ML}(d,q)$. Therefore, full-dimension ADE becomes full-dimension LSI, which is just VSM when training on the retrieval corpus. On the other hand, when $k = 0$, the above formula would have only the $(1/\sigma_k^2)\,\text{Sim}_{GVSM-ML}(d,q)$ part left. So, ADE is just GVSM with no SVD involved.

In cross-language retrieval, even though we cannot do the same tricks as in the monolingual case, the dense matrix $\tilde{B}_k$ is still never directly formed. Instead, we compute $\tilde{B}_k^T \vec{q}$ by forming a transformed query vector at each step (assuming

$B = W\,\Omega\,X^T$ as in Section 3.4.4):

$$\begin{aligned}\tilde{B}_k^T\vec{q} &= \bar{B}_k^T\vec{q} + \frac{1}{\omega_k}B^T\vec{q} - \frac{1}{\omega_k}B_k^T\vec{q} \\ &= X I_k W^T\vec{q} + \frac{1}{\omega_k}B^T\vec{q} - \frac{1}{\omega_k}X\,\Omega_k W^T\vec{q}.\end{aligned}$$

Note that there is no intermediate result (matrix) bigger than the size of the number of queries by the number of documents (or terms, whichever is bigger). Even if we process queries in batches of tens or hundreds, those matrices are still not very large. Then, since

$$\mathrm{Sim}_{ADE-TL}(d,q) = (\tilde{A}_k^T\vec{d}\,)\cdot(\tilde{B}_k^T\vec{q}\,) = \vec{d}^T\tilde{A}_k\tilde{B}_k^T\vec{q},$$

$\tilde{A}_k\tilde{B}_k^T\vec{q}$ can be computed in a similar way.

An experimental comparison between LSI and the ADE method in terms of the relative weights given to the orthogonal dimensions of the training matrix is shown in Figure 4.3. This matrix is created from the 1,400-document Cranfield corpus, detailed results on which will be discussed in Section 4.3.1. To show a real-data graph similar to Figure 4.2, I plotted Figure 4.3 based on the results in Table 4.4. According to that table, LSI achieves its best performance with 300 orthogonal dimensions, all evenly weighted, and the rest are discarded. The relative weights given to these dimensions by LSI is shown in dashed line and labeled "LSI Ideal" in Figure 4.3. Now, suppose in reality I were only able to compute the first 75 SVD dimensions. The relative weights used by LSI in this case is shown in solid line and labeled "LSI Reality" in the graph. With the ADE method, I implicitly attach the remaining 1,325 dimensions slightly down-weighted, the dimension weights of which is plotted as the dash-dot line in the figure. As we can see, the ADE graph covers not only the entire area under the

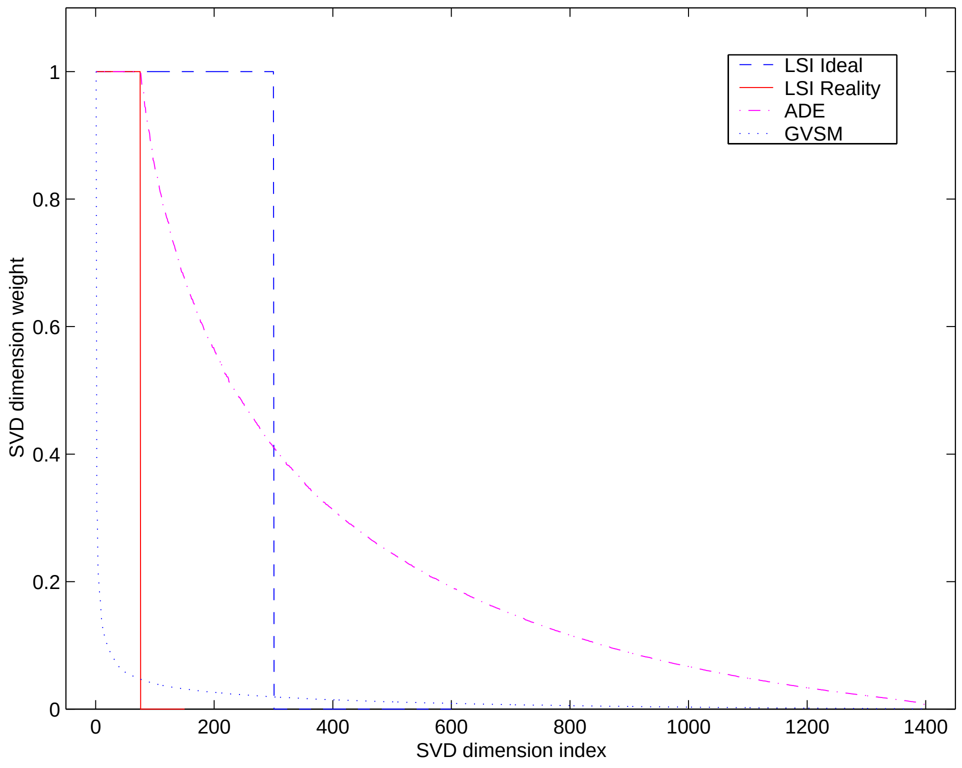

Figure 4.3: Relative dimension weights used by GVSM, LSI, and ADE on the retrieval of the Cranfield collection: ADE simulates the effect of an ideal LSI retrieval by computing fewer initial dimensions and attaching the rest implicitly.

"LSI Reality" plot, but also a large portion of the area under the "LSI Ideal" line. In terms of performance, with 75 computed dimensions, ADE did nearly as well as LSI with 300! For comparison, I also include the dimension-weight plot of GVSM in the same graph. Since in the retrieval process we rank documents by their similarity scores to the query, what is important is the relative instead of the absolute weight of each orthogonal dimension. Thus, the plotted GVSM weights are normalized by dividing all the singular values by the first one. Since the first few singular values are relatively large compared to the rest of them (the low-rank-plus-shift structure), we see the very sharp drop at the beginning of the GVSM plot. Performance-wise, the average precision of GVSM on this collection is approximately the same as that of LSI with only 25 dimensions.

4.3 Experimental Results

In this section, I present some results of applying the four vector-based methods—VSM, GVSM, LSI, and ADE—on collections of different sizes and in different languages. I have used both the traditional *tf* $\times$ *idf* and the more recent Okapi weighting schemes to build the training matrices, and have presented my results with whichever weighting scheme that works better with the conventional VSM.

Except for some of the rather small collections, most of my experiments were run on a SGI machine named "dragon" in the Department of Computer Science at Duke University. This computer has four MIPS R10000 2.5 processors and 2 gigabytes of RAM. I will refer to this machine simply as "dragon" in the experiment descriptions where I mention the actual time spent on SVD computation.

Also at a few places in the descriptions of experiments, I report the variance captured by the computed SVD dimensions. As discussed in Section 3.4.2, for a training matrix $A = U\,\Sigma\,V^T$, the total variance due to the first k singular values is

$$\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}.$$

where $r = \text{rank}(A)$. Since $\|A\|_F^2 = \|\Sigma\|_F^2 = \sum_{i=1}^{r} \sigma_i^2$, the divisor in the above formula can be easily computed through the Frobenius norm of A.

The variety of the test collections that I used makes their individual details difficult to keep track of. Therefore, before starting the discussion on the experiments, I give a summary of the processing details of these collections in Table 4.3, so that the readers can locate relevant information quickly. Note that since in all the experiments, stemming and stop-word removal are always applied together if

Collection	Set	Docs	Stem & SW	Weighting	Dim
Cranfield	Documents	1,400	Yes	SMART *ntc*	1,400
	Queries	225	Yes	SMART *ntc*	–
Medlars	Documents	1,033	Yes	SMART *ntc*	1,033
	Queries	30	Yes	SMART *ntc*	–
CMU UNICEF	Training	1,134	Yes	SMART *ntc*	1,117
	Test	1,121	Yes	SMART *ntc*	1,103
	Queries	30	Yes	SMART *ntc*	–
TREC	AP 1990	78,321	Yes	SMART *ntc* Okapi	800 800
	TREC 4 Topics	50	Yes	SMART *ntc* binary	–
	TREC 6 & 7 Topics	53	Yes	SMART *ntc* binary	–
TREC	French	141,643	No	SMART *ntc*	–
	German	185,082	No	SMART *ntc*	–
	Fre Training	40,000	No	SMART *ntc*	1,700
	Ger Training	40,000	No	SMART *ntc*	1,700
	TREC 6 Topics	25	No	SMART *ntc*	–
	TREC 7 Topics	28	No	SMART *ntc*	–
TREC	English (AP)	242,918	Yes	SMART *Lnu*	–
	French	141,643	Yes	SMART *Lnu*	–
	Eng Training	30,024	Yes	SMART *ntc*	2,200
	Fre Training	30,024	Yes	SMART *ntc*	2,000
	TREC 6 Topics	25	Yes	SMART *ltn*	–
	TREC 7 Topics	28	Yes	SMART *ltn*	–
NTCIR-1	Japanese	332,918	No	SMART *Lnu*	–
	English	187,080	Yes	SMART *Lnu*	–
	Jap Training	45,372	No	SMART *ntc*	1,400
	Eng Training	45,372	Yes	SMART *ntc*	1,200
	Training Topics	21	No	SMART *ltn*	–
	Test Topics	39	No	SMART *ltn*	–

Table 4.3: Information on how the test collections were processed is given in terms of the number of documents, whether stemming and stop-word removal are used, the weighting scheme being applied, and the number of SVD dimensions computed.

they are used, Table 4.3 has only one column for these two steps.

4.3.1 Cranfield and Medlars Collections

I begin with two English monolingual retrieval collections, Cranfield and Medlars, both can be downloaded from Cornell University computer science department's web site (ftp://www.cs.cornell.edu/pub/smart/). The Cranfield corpus consists of 1,400 documents on aerodynamics and 225 test queries, while Medlars consists of 1,033 medical abstracts and 30 queries. These collections are very small but were used extensively in the past by IR researchers. I use them to demonstrate LSI's ability to derive term-term correlations via the reduced-dimension semantic space.

Both collections were first stemmed and stop-word removed with the SMART software available at Cornell's ftp site. I then proceeded to create the term-document matrix using the *ntc* weighting scheme as shown in Table 2.2.

For all four methods, I developed software tools to do matrix addition and multiplication. For LSI computation, I used SVDPACK, a sparse SVD package developed by Michael Berry at the University of Tennessee at Knoxville [8]. A complete SVD of the Cranfield document matrix takes about 20 minutes to compute on an AlphaStation with one 266Mhz processor. For GVSM, I computed the vector dot product between the transformed query and documents vectors as shown in Equation 3.10, which is replicated here:

$$\mathrm{Sim}_{GVSM-ML}(d,\, q) = \vec{d}^T A A^T \,\vec{q} = (A^T \vec{d}\,) \cdot (A^T \vec{q}\,).$$

Now, if $A^T \vec{d}$ is computed for every document in a collection D, the resulting set of vectors $(A^T D)$ comprise a very dense matrix to be stored on disk. This may not seem to be a lot for small collections such as Cranfield, but $A^T D$ would require a huge amount of storage space if D represents a very large collection. Thus, when running GVSM, I usually compute $A^T\vec{q}$ and then $AA^T\vec{q}$ before comparing the resulting vector to $\vec{d}$. As the number of test queries are small, so are the sizes of intermediate results—$A^T\vec{q}$ and $AA^T\vec{q}$ for a small number of $\vec{q}$.

The retrieval results on Cranfield shows the positive effect of query expansion via LSI dimension reduction While the full rank of the Cranfield matrix is 1,400, LSI achieves its best average precision of 0.4196 at 300 dimensions, which account for 22% of the total dimensionality and 67% of the total variance. Similarly, ADE's best performance of 0.4115 is obtained at the smaller dimensionality of 75 (only 5% of the total). Here, by dimensionality, I mean the cut-off points before which the initial singular values are "leveled," even though in theory ADE assigns nonzero weights to all the dimensions of the training matrix. The fact that both scores are much higher than that of VSM confirms the ability of LSI to derive term-term correlations at the reduced dimensions of the semantic space. To examine the effects of dimension reduction by LSI and ADE, I plotted the average precision against SVD dimensionality from 50 to 1,400 in Figure 4.4. The average precision of VSM is included in the same graph for comparison.

There are several things to be noticed in this graph. First and most importantly, the plot of ADE looks like a shift-backward image of that of LSI, confirming that it approximates LSI with fewer dimensions, "extrapolating" the results of high-dimension SVD's based on the values computed. The shift is approximately 200 dimensions. Secondly, at the full dimension of 1,400, the LSI performance score equals 0.3800, the average precision of VSM, an empirical proof of one of my theoretical conclusions: VSM is just full dimensional LSI training on the retrieval collection itself (which I did for this collection). ADE's result is also the same, since full dimensional ADE is just full dimensional LSI. Finally, the score of GVSM is too low to fit in the range of Figure 4.4, but by observing the performance tendency of LSI at low dimensions, we can imagine that LSI and GVSM will match their performance at some point—around 25 dimensions as mentioned in Section 4.2. This is also in accordance with my early prediction that GVSM is like a low-dimension LSI.

Because GVSM produced a dismal average precision of 0.2173, I tried a normalized version of Equation 3.10. Namely, instead of dot product of transformed vectors, I used the cosine measure for similarity comparisons:

$$\cos(A^T\vec{d},\, A^T\vec{q}\,) = \frac{\vec{d}^{\,T} A A^T \vec{q}}{\|A^T\vec{d}\,\| \cdot \|A^T\vec{q}\,\|}. \tag{4.1}$$

With this normalization step, GVSM produced a much higher average precision of 0.4145. This extra computation step is not practical for large collections because it now requires the actual vector transformation of $A^T\vec{d}$. As I mentioned earlier in this section, for an entire collection this becomes a very dense matrix $A^T D$, which would require enormous storage space if both A and D are large in size. But, since the Cranfield collection is fairly small, the cosine comparison is still feasible. In a similar way, I produced new sets of results for LSI and ADE using the cosine measure as well: $\cos(\bar{A}_k^T\vec{d},\, \bar{A}_k^T\vec{q}\,)$ and $\cos(\tilde{A}_k^T\vec{d},\, \tilde{A}_k^T\vec{q}\,)$. This extra step also improved the results of these two methods on this collection.

Similar to Figure 4.4, Figure 4.5 shows the effects of computed dimensions used by LSI and ADE with the cosine comparison. For VSM, since I used the SMART *ntc* weighting scheme, which already includes document length cosine normalization, its result does not change in the new figure. The new plots appear to be less "regular" than the ones without cosine normalization, but they still show ADE approximating LSI with approximately 200 fewer computed dimen-

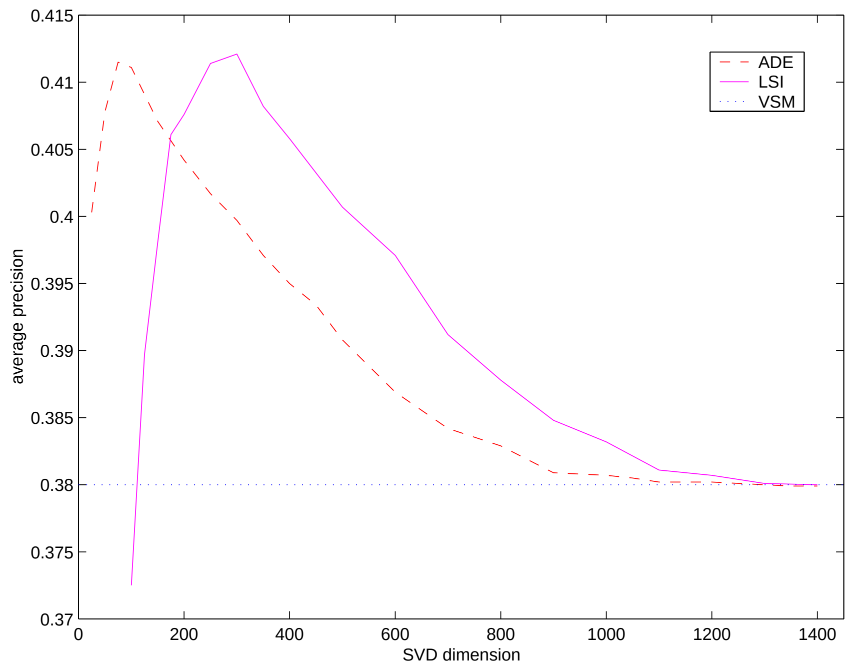

Figure 4.4: LSI and ADE average precisions rise and fall with the number of computed dimensions they use on the Cranfield collection; ADE appears to be a shifted image of LSI.

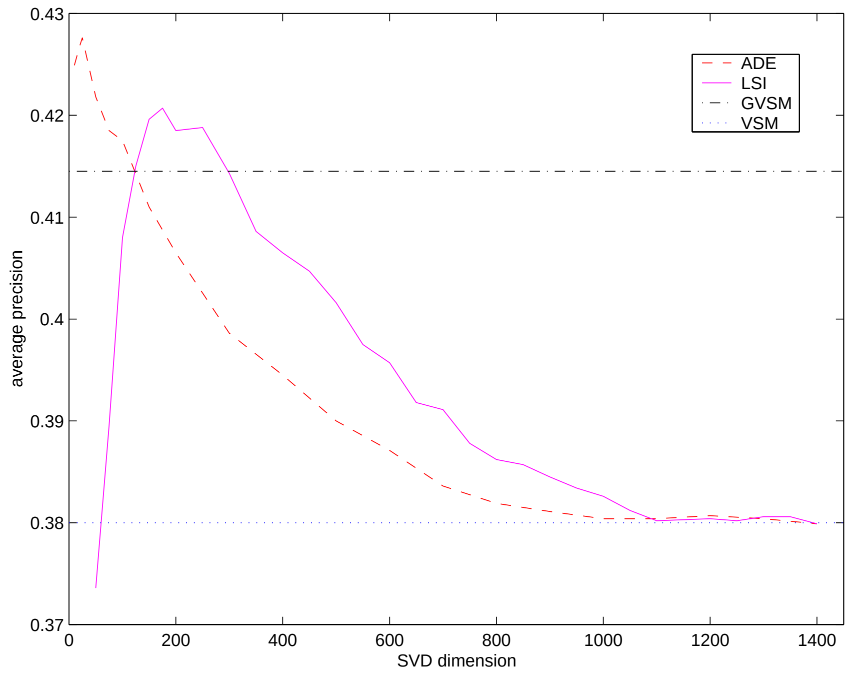

Figure 4.5: LSI and ADE average precisions vary with the number of computed dimensions they use on Cranfield retrieval; the cosine similarity measure brings up the performance of GVSM, LSI, and ADE.

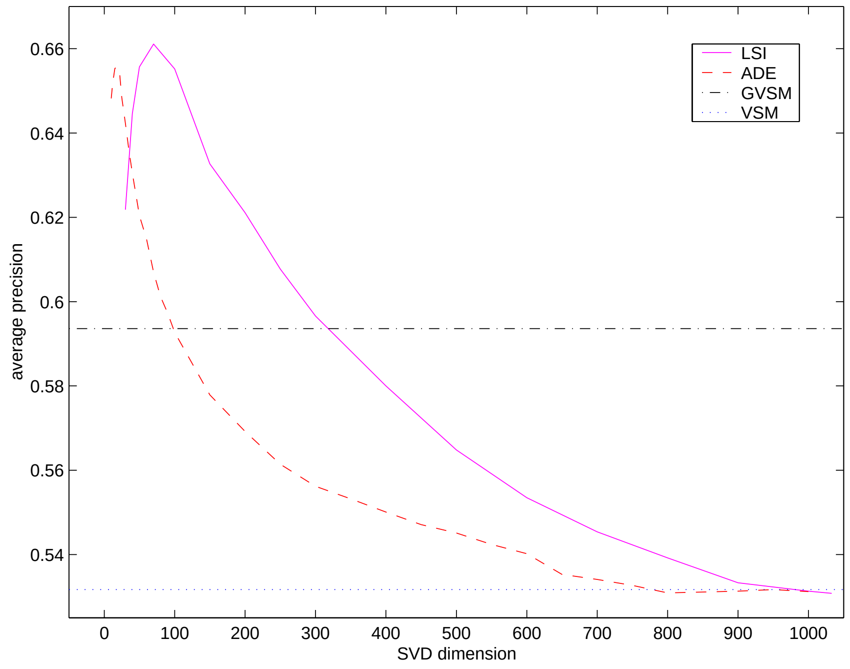

Figure 4.6: ADE again appears to be a shift-backward image of LSI on the Medlars collection.

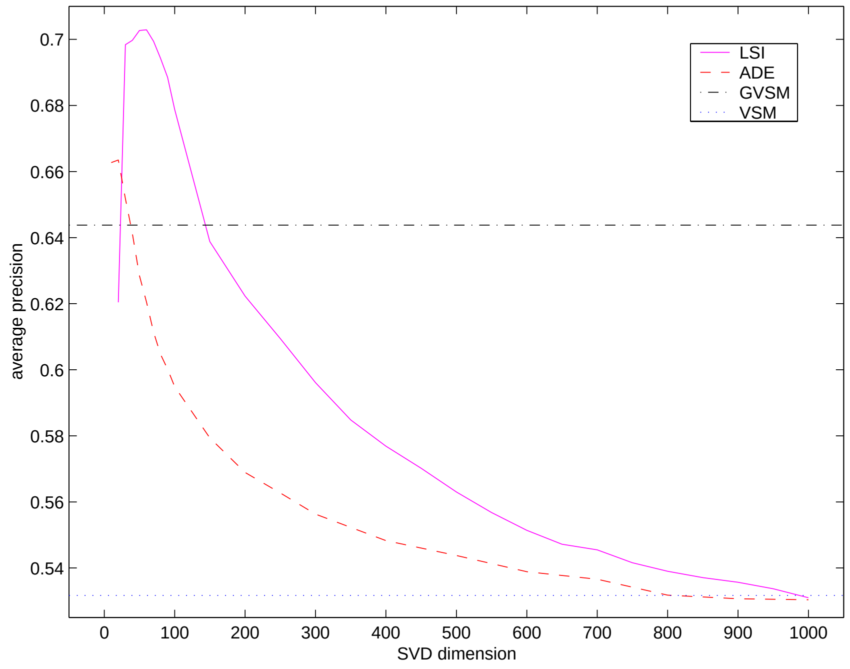

Figure 4.7: Cosine measure greatly enhances LSI performance but only improves those of GVSM and ADE slightly.

	Cranfield			Medlars		
	AvgP	P10	Dim	AvgP	P10	Dim
VSM	0.3800	0.2862	1,400	0.5317	0.6367	1,033
GVSM	0.2173	0.1924	–	0.5936	0.6900	–
GVSM cosine	0.4145	0.3098	–	0.6438	0.7067	–
LSI	0.4121	0.3080	300	0.6611	0.7233	70
LSI cosine	0.4207	**0.3191**	175	**0.7029**	**0.7467**	60
ADE	0.4115	0.3062	75	0.6546	0.7167	20
ADE cosine	**0.4276**	0.3169	25	0.6635	0.7133	20

Table 4.4: Results on the Cranfield and Medlars collections are shown in terms of average precision, precision at document cutoff of 10, and SVD dimensionality at which the corresponding precision scores are obtained. Boldface indicates the best score in a column.

sions.

For the Medlars collection, the average-precision-versus-dimension plots of LSI and ADE are shown in Figure 4.6 and 4.7, with Figure 4.7 showing the results of using cosine normalization for comparison. The shape of the plots are very similar to those of the Cranfield collection, making the same compelling point that term-term correlations are in effect and that ADE approximates LSI with fewer computed singular vectors. Another notable thing is that the cosine measurement gives LSI performance a much greater boost than it gives to GVSM and ADE, while on the Cranfield collection it is the opposite. This shows the cosine normalization is quite erratic in affecting retrieval results in addition to its unscalability.

The best results from each method on both collections are summarized in Table 4.4. It contains average precisions as well as precisions at document cutoff value of 10, both standard measures in the information retrieval field. A third column for each collection shows the equalized dimensions used by the respective methods.

4.3.2 CMU UNICEF Collection

My next empirical evaluation of the four vector-based methods is on the UNICEF document collection created by Carbonell et al. [26]. This collection consists of 1,134 training documents and 1,121 test documents, each in both English and Spanish. A set of 30 English queries is also provided along with exhaustive relevance judgments for these queries over all 1,121 test documents. The relevance judgments were made between the English queries and the English test documents, and this is what I use for my monolingual retrieval evaluation. Since the Spanish test documents are translationally equivalent to the English ones, the relevance judgments between the English queries and the Spanish test documents are assumed to be the same as in the monolingual English case. This cross-language relevance is needed for the evaluation of cross-language information retrieval methods. See Appendix A.2 for examples of the actual documents and queries in the UNICEF collection.

Like the Cranfield and Medlars collections, I processed the CMU UNICEF English collection and queries with stemming and stop-word removal using the SMART software. I then indexed the documents and the queries using the SMART *ntc.ntc* weighting scheme, which gives the best baseline VSM performance compared to other schemes on this collection [121]. After indexing, the matrix of the English retrieval corpus contains 7,542 unique terms and the matrix of the English training corpus has 7,739 terms. I performed VSM retrieval directly through the SMART system and also with my own VSM program—the two results came out identically: 0.4562 for non-interpolated average precision.

The trainings in GVSM, LSI and ADE were first done directly on the matrix of the retrieval documents, and then on the matrix of the training documents to compare the results. I computed complete SVD's on the two matrices; both take less than 20 minutes on the SGI machine "dragon." Apparently, the matrices are rank deficient: the SVD software package finds 1,103 of the 1,122 possible nonzero singular values of the retrieval matrix, and 1,117 of the 1,134 singular values of the training matrix.

I then ran LSI and ADE at various dimensions to observe the effect of rank reduction as in the Cranfield and Medlars experiments. Again, I plotted the average precision scores computed at SVD dimensions from 20 to the full rank of 1,103 in Figure 4.8. Also included in the same graph is the plot for GVSM with varying sparsification numbers. Recall that sparsification is a technique used by Yang [120] that set all but the k largest elements in a vector to zero (see Section 3.3). The vector in the case of GVSM is the transformed document or query vector, and I varied the number k also from 20 to 1,121 (i.e. no sparsification) to study its influence on retrieval performance. For this collection, I found that cosine normalization still improves the performance of GVSM dramatically but not so much for LSI and ADE. However, the results shown in the graph are still from using cosine normalization after projection for all three methods.

Even though the horizontal axes of the LSI and ADE plots and the GVSM plot represent different variables, the graph still reveals to us that ADE and LSI perform much better than GVSM on this collection. The horizontal dotted line across the upper half of the graph represents the VSM average precision. It is very encouraging to see that once again the scores of ADE and LSI climb higher than that of VSM and peak at a range of very low dimensions. As in Cranfield and Medlars, ADE appears to approximate LSI with fewer computed dimensions.

As I mentioned earlier, typically people train GVSM and LSI directly on the collection they perform retrieval on in monolingual retrieval. But, since the

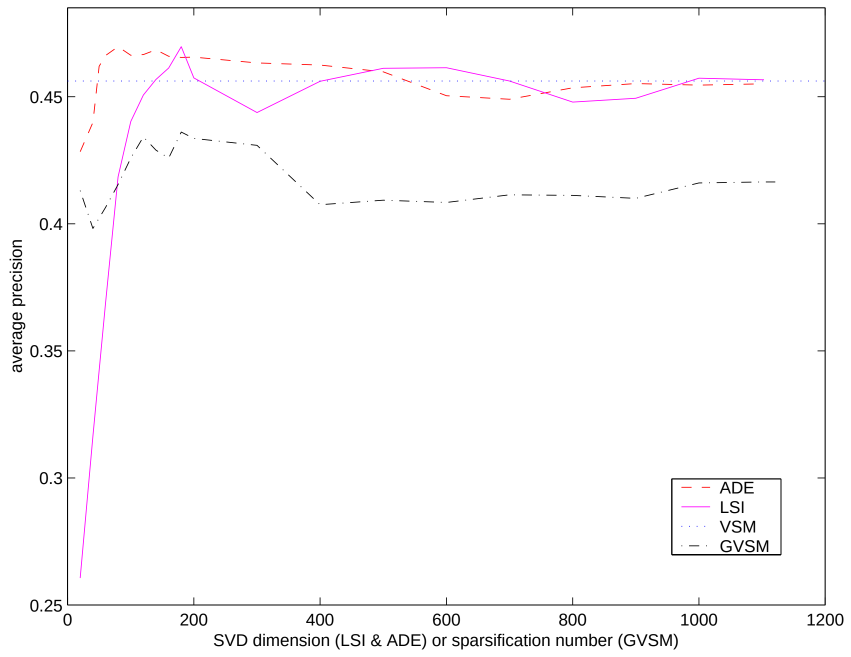

Figure 4.8: ADE outperforms VSM and LSI at low dimensions, and GVSM with sparsification at all dimensions in MLIR on the CMU UNICEF collection.

UNICEF data includes a separate set of training documents, I also ran the three query-expansion methods with this training set. The plots of average precision against dimension or sparsification number are shown in Figure 4.9. Here, as expected, we see an overall performance decline for all three methods compared to when training them directly on the retrieval collection. However, ADE still comes out at the top and the shape of its plot mirrors that of LSI with a smaller number of dimensions. In both Figure 4.8 and 4.9, the curves of LSI and ADE once again converge at high dimensionalities.

For cross-language LSI retrieval, I used the new formula (Equation 3.32) that I proposed in Section 3.5.1, instead of the traditional approach by Landauer and Littman [65] (see Section 3.4.3). Since my formula requires separate SVD's being computed for the two parallel corpora, I indexed the 1,134 Spanish training documents also with the SMART *ntc* weighting; the resulting matrix has 15,343

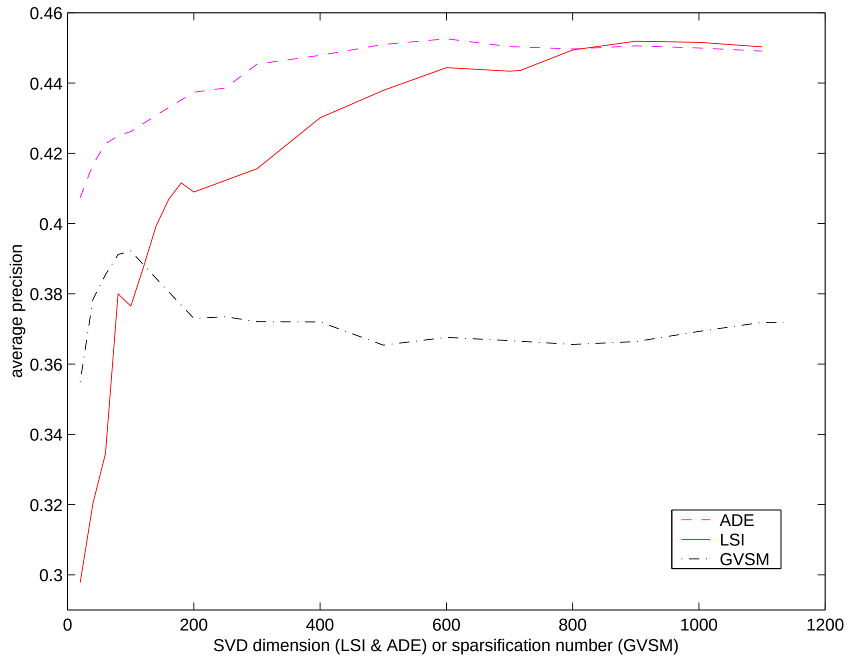

Figure 4.9: ADE outperforms LSI and GVSM using a separate training corpus in MLIR on the CMU UNICEF collection

unique terms and 1,117 dimensions. A performance comparison of LSI, ADE, and GVSM are shown in Figure 4.10. The shape and relative position of the three plots appear to be very similar to their monolingual counterparts in Figure 4.9. Once more, ADE achieves impressive average precision scores at low dimensions.

Since the CMU UNICEF collection is relatively small, I was able to compute the generalized singular value decomposition on the training matrix as discussed in Section 3.4.5. Without a sparse-matrix package for GSVD, I used the LAPACK routine "dgssvd" that can handle a pair of $m \times n$ and $p \times n$ general matrices, where in practice m, n, and p depend on the size of the RAM on a particular computer. The average-precision-versus-dimension plot of the GSVD method is also included in Figure 4.10. Here, we see its performance mimics that of LSI on this collection. Unless we have more RAM or a better software package to handle GSVD, the size of the CMU collection is already pushing the limit of the current

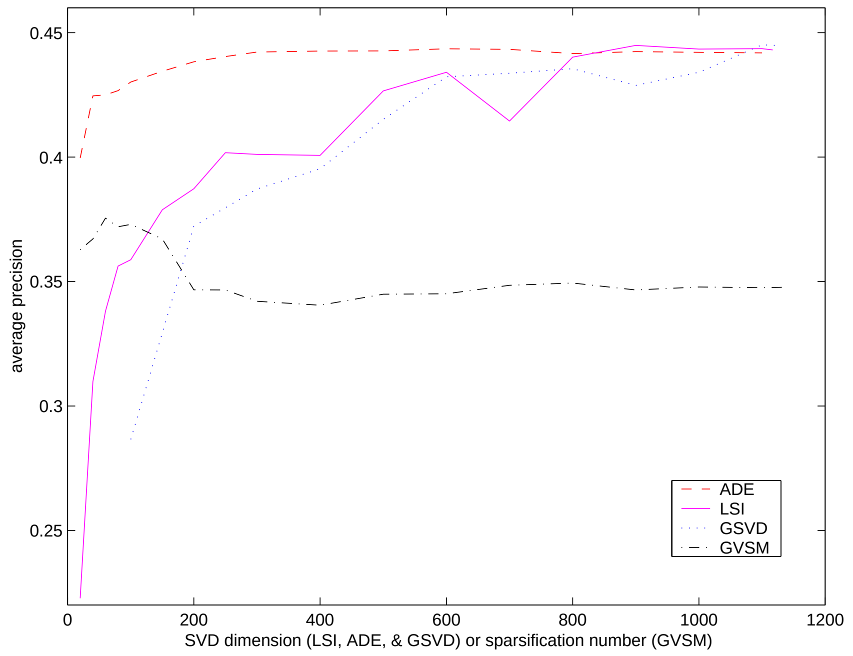

Figure 4.10: At low dimensions, ADE outperforms LSI, GSVD, and GVSM in CLIR on the CMU UNICEF collection; the curve of GSVD is very similar to that of LSI.

computing resources.

The best monolingual results for each method are summarized in Table 4.5. Notice that while LSI achieves its best score with 180 dimensions, or approximately 17% of the full dimensionality (and 56% of total variance), ADE accomplishes the same feat with only 7% of the rank. However, as we will see in later sections, it becomes impossible even to obtain 5% of the total dimensionality on very large matrices with the current computing resource. In Table 4.6, we see that although the best results of LSI, ADE, and GSVD in cross-language retrieval are nearly identical, ADE accomplishes it with the fewest computed dimensions. In fact, ADE's average precision reaches the 0.4400-level with only 250 dimensions! Moreover, it is very intriguing to see that with the same training collection, the best results of LSI and ADE were obtained at exactly the same dimensionalities (900 and 600, respectively) in both monolingual and cross-language runs.

	$A = D$				$A \neq D$			
	AvgP	P10	%VSM	D/S	AvgP	P10	%VSM	D/S
VSM	0.4562	0.3931	100	1,121	–			
GVSM	0.4361	0.4276	95.6	180	0.3922	0.4172	86.0	100
LSI	**0.4697**	**0.4276**	103.0	180	0.4519	0.4207	99.1	900
ADE	**0.4697**	0.4207	103.0	80	0.4526	0.4172	99.2	600

Table 4.5: Monolingual results on the CMU UNICEF collection: LSI achieves the best scores and ADE is close to the best using fewer computed dimensions. The last two columns for each type of run show the averages precision as a percentage of the baseline VSM performance and dimensionality or sparsification number at which the scores in each row are obtained.

	AvgP	P10	%Monolingual	%VSM	D/S
GVSM	0.3755	0.3966	95.7	82.3	60
LSI	0.4449	0.4034	98.5	97.5	900
ADE	0.4435	0.4000	98.0	97.2	600
GSVD	**0.4451**	**0.4042**	–	97.6	1100

Table 4.6: Cross-language results on the CMU UNICEF collection: LSI, ADE, and GSVD achieve similar scores with ADE using the fewest computed dimensions. The "%Monolingual" column shows the cross-language average precision as a percentage of the monolingual one for each method.

4.3.3 TREC Associated Press Collection 1990

My next test corpus is a set of Associated Press (AP) news articles from the TREC-6 CLIR collections [101]. While the entire English AP collection consists of 242,918 news items from years 1988 to 1990, I used only 78,321 documents from year 1990. I use this sub-collection because I wanted the corpus to be large enough to demonstrate the scalability problem of LSI, and yet small enough to be able to compute an SVD. I will refer to this sub-collection as the AP 1990 collection; sample documents from it can be found in Appendix A.3.

Again, since this collection is in English, I used SMART for stemming and stop-word removal; this resulted in 114,698 unique terms in the processed documents. I then indexed the collection with the SMART *ntc* weighting scheme. I used two sets of queries—TREC-4 ad-hoc topics 201–250 and TREC-6 and

TREC-7 cross-language topics 1–53, because they have the corresponding relevance judgments made against the AP 1990 documents (see Appendix A.3 for some examples of the topics). The TREC-6 and TREC-7 topics are designated "cross-language" because they were used in the cross-language track of the Text REtrieval Conference. They have both monolingual and cross-language relevance judgments made for collections in their own languages and other languages, respectively. In this experiment, I used the TREC cross-language topics in English along with their relevance judgments on the AP collection for my monolingual experiments. I extracted the "long" version of these topics that consists of the title, description, and narrative fields of the original topics. I indexed the queries with the *ntc* weighting and ran the baseline VSM retrieval. The average precisions are 0.2132 for the 50 ad-hoc queries and 0.4056 for the 53 cross-language queries.

As discussed in Sections 2.3.1 and 2.3.3, in recent years the Okapi term-weighting scheme [88] has been very popular for experiments on TREC data. Therefore, I also indexed the AP documents with the Okapi formula used by Franz et al. [42]:

$$\frac{tf}{c_1 + c_2 \times \frac{dl}{avdl} + tf} \times \log\left(\frac{n - k + 0.5}{k + 0.5}\right),$$

where tf is, as usual, the occurrence frequency of a term in the document, dl the document length, $avdl$ the average length of the documents in the collection, n the total number of documents in the collection, and k the number of documents containing this term. The weights c_1 and c_2 are set to 0.5 and 1.5, respectively. The queries are indexed with binary weighting: a term's position in the query vector contains a value of 1 if the term is in the query and 0 otherwise. As a result, I obtained somewhat surprising results: an improved average precision of 0.2964 for the ad-hoc queries, but a declined average precision of 0.3732 for the cross-language ones. I have also observed better performances with the SMART *ntc* weighting for cross-language queries on other TREC collections such as the French collection (see Section 4.3.4). The reason behind this remains to be investigated but is beyond the scope of this dissertation.

I then proceeded to compute an SVD on each matrix. The computation took 15 and 21 hours on "dragon"; the reason for the time difference may be the load of the system during the experiment. Because of computer memory limitations, I was only able to calculate a little more than 800 singular vectors and values. This is merely about 1% of the rank of the matrices and 32% of the total variance. As a result, LSI scored substantially lower than VSM in retrieval performance. But, ADE partially bridges the gap between LSI and VSM, achieving results close to VSM with the same number of computed dimensions as LSI. The results are listed in Tables 4.7 and 4.8, with extra columns showing the number of individual queries on which VSM is outperformed by other methods. We do see an encouraging sign that on about one-fourth of the queries, LSI query-expansion actually works with only a very limited number of dimensions; ADE is even better, improving performance on nearly half of them.

Because of the size of the collection, I was not able to compute LSI, ADE, and GVSM in the special normalization-after-transformation way as for Cranfield and other small collections. As a result, GVSM gives extremely poor performances with both query sets and both indexing schemes; again, it behaves like a very low-dimension LSI as predicted.

I also included the average-precision-versus-dimension plot for this collection. Figure 4.11 reveals the effect of dimension on performance with TREC-4 ad-hoc queries and Okapi weighting, while Figure 4.12 illustrates a similar effect with

	TREC 4 ad-hoc (50 queries)			TREC 6 and 7 cross-language (53 queries)		
	AvgP	P10	# ≥ VSM	AvgP	P10	# ≥ VSM
VSM	**0.2132**	**0.2479**	–	**0.4056**	**0.4044**	–
GVSM	0.0240	0.0333	0	0.0534	0.0844	1
LSI	0.1322	0.2021	16	0.2318	0.2578	10
ADE	0.2049	0.2354	25	0.3418	0.3600	15

Table 4.7: Results on the TREC AP 1990 collection with SMART *ntc* indexing: ADE not only bridges the gap between LSI and VSM, but also outperforms VSM on a number of queries.

	TREC 4 ad-hoc (50 queries)			TREC 6 and 7 cross-language (53 queries)		
	AvgP	P10	# ≥ VSM	AvgP	P10	# ≥ VSM
VSM	**0.2964**	**0.3437**	–	**0.3732**	**0.3956**	–
GVSM	0.0060	0.0125	0	0.0153	0.0356	0
LSI	0.2328	0.2724	17	0.2149	0.2711	10
ADE	0.2841	0.3292	27	0.3499	0.3600	20

Table 4.8: Results on the TREC AP 1990 collection with Okapi indexing: they are very similar to the results on the same collection with SMART *ntc* weighting.

TREC-6 and TREC-7 cross-language queries and SMART *ntc* indexing. In both figures, we see that the results for LSI and ADE are strictly increasing, as they are still trying to "catch up" with the performance of VSM using a small number of singular vectors. ADE achieves the effect of approximately 700 additional dimensions. Note that while selecting the optimal dimension is a tricky issue in the small collections, it does not appear to be important for large collections—more dimensions are consistently better than fewer [79].

4.3.4 TREC French and German Collections

The next set of results is on a pair of much larger collections: the French and German collections used in the cross-language track of the Sixth Text REtrieval Conference [101]. The French collection consists of 141,643 news articles from the

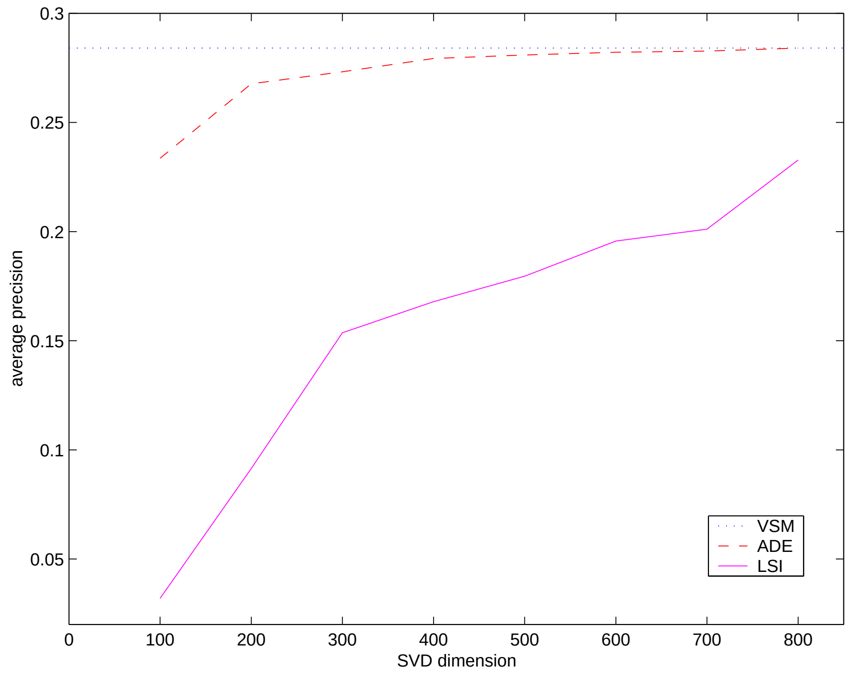

Figure 4.11: ADE outperforms LSI and approaches the performance level of VSM on the AP 1990 collection with TREC 4 ad-hoc queries and the Okapi term weighting.

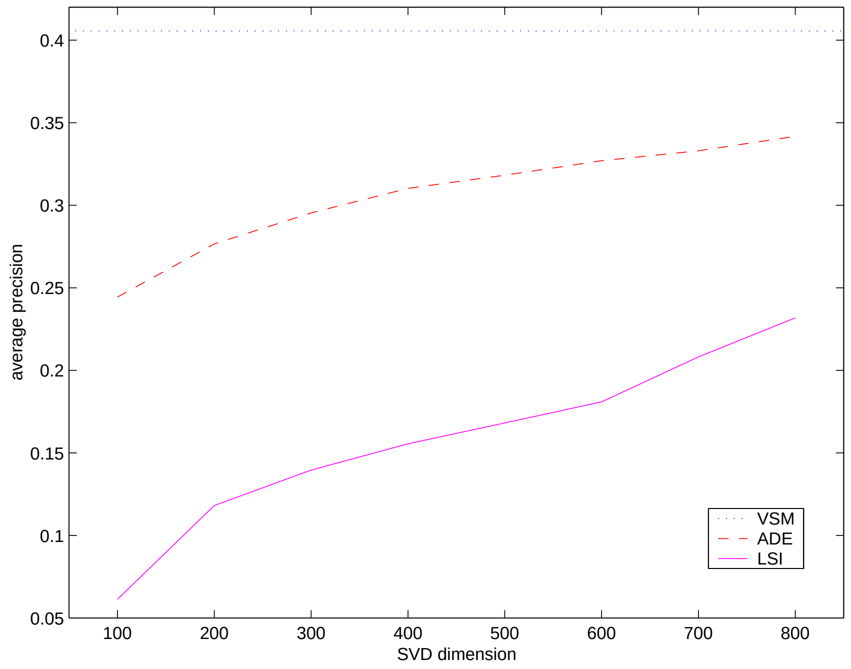

Figure 4.12: ADE out-distances LSI and approaches the performance level of VSM on the AP 1990 collection with TREC-6 and TREC-7 cross-language queries and the SMART *ntc* term weighting.

Swiss news agency (Schweizerische Depeschen Agentur or SDA), and the German corpus contains 185,082 documents from the same agency. Like the TREC AP collections, these news articles are specially selected to cover the same time span (1988 to 1990). The same test queries as the 53 English ones I used on the AP 1990 collection are available in both French and German. They are the ones that I used in experiments in this section. Since there are two query sets, and both can be used for monolingual and cross-language retrieval, I will adopt the convention of referring to a particular retrieval run in terms of query-language-document-language from now on. For example, by "French-German" retrieval, I mean using French queries to retrieve German documents in a cross-language run. For cross-language training, I used a subset of 40,000 pairs of documents aligned by Rehder et al. [84]. Each pair of documents are only comparable in content: they are on the same topic and from the same time period, but are not direct translations (see Appendix A.1 for a sample pair).

Similar to the AP 1990 collection, I first indexed the French collection with both SMART *ntc* and Okapi weighting schemes. I did not use any stemming or stop-word removal on the collection, but I did remove accent mark from all words (for example, "décembre" was replaced by "decembre"), since it was discovered that a large minority of the words in the French collection that should have had accents did not [20]. The resulting matrices have 196,878 rows for unique terms. After running the baseline VSM, I obtained the following non-interpolated average precisions for the first 25 TREC-6 topics: 0.1786 for Okapi indexing and 0.3162 for SMART *ntc*. (I separated the topics by conference, TREC-6 and TREC-7, because there exist published results for these query subsets with which I can compare my results.) The fact that the cross-language topics work better with SMART *ntc* indexing appears consistent with the results I obtained for the

AP 1990 collection. So, I decided to use *ntc* weighting in subsequent experiments.

I created matrices for the 40,000 French and German training documents the same way I did for the complete French collection. The French training corpus contains 91,634 terms while the German corpus has 221,366 terms. I created 1,700-dimensional (only a little more than 4% of the full rank) subspaces from the two matrices using the SVD package, which took approximately 32 hours for the French matrix and 39 hours for the German one on "dragon." The 1,700 singular values account for 49% and 38% of the total variance in the French and German training matrices, respectively.

I tested GVSM, VSM, and ADE with monolingual and cross-language retrieval runs; results are shown in Tables 4.9 and 4.10. Again, I found both LSI and ADE lagging behind VSM in both monolingual French and monolingual German retrieval. But, while VSM cannot be applied to cross-language retrieval, ADE obtains very strong results in that territory. French-German retrieval with TREC-6 queries had an average precision of 0.2177, or about 77% of the German monolingual performance. This compares favorably with one of the best results obtained for this collection by Franz el al. [42], who used passage-aligned text to train a statistical machine translation system and produced an average precision of 0.2361 (68% of monolingual performance). ADE's score was lower in absolute value compared to theirs, but they achieved a much higher monolingual performance (0.3478) than I was able to, even though I implemented their Okapi formula to index the documents.

Noticeable from the tables is the significant performance decline of retrieving on the German collection from TREC-6 to TREC-7 queries. One reason could be that while there are approximately 41.3 relevant documents per TREC-6 query in the German collection, there are only 27.6 relevant documents per TREC-7

		Monolingual		Cross-language	
		French	German	Ger ⇒ Fre	Fre ⇒ Ger
VSM	AvgP	**0.3162**	**0.2958**	–	–
	P10	**0.4714**	**0.4043**	–	–
GVSM	AvgP	0.0509	0.0101	0.0206	0.0187
	P10	0.1333	0.0174	0.0429	0.0435
LSI	AvgP	0.1711	0.1025	0.1175	0.1495
	P10	0.2190	0.1609	0.1810	0.2217
ADE	AvgP	0.2064	0.1635	**0.1671**	**0.2177**
	P10	0.3095	0.2304	**0.2810**	**0.2957**

Table 4.9: Monolingual and cross-language results on the TREC French and German corpora with TREC-6 cross-language topics and SMART *ntc* indexing.

		Monolingual		Cross-language	
		French	German	Ger ⇒ Fre	Fre ⇒ Ger
VSM	AvgP	0.2796	**0.2541**	–	–
	P10	**0.4357**	**0.3259**	–	–
GVSM	AvgP	0.0529	0.0004	0.0189	0.0217
	P10	0.0929	0.0000	0.0250	0.0407
LSI	AvgP	0.2070	0.0651	0.0947	0.0751
	P10	0.3036	0.0852	0.1250	0.1000
ADE	AvgP	**0.2929**	0.1527	**0.1933**	**0.1572**
	P10	0.4000	0.2148	**0.2607**	**0.1963**

Table 4.10: Monolingual and cross-language results on the TREC French and German corpora with TREC-7 cross-language topics and SMART *ntc* indexing.

	Monolingual		Cross-language	
	French	German	Ger ⇒ Fre	Fre ⇒ Ger
VSM	24	30	–	–
GVSM	1	1	1	2
LSI	6	3	4	1
ADE	18	16	44	47

Table 4.11: Number of TREC 6 and 7 topics on which each method achieves the best performance on the TREC French and German corpora.

query. Fewer relevant documents in the top returned list means their slots are filled by irrelevant documents, thus bringing the average precision down. But, this still does not explain the decline in precision at the 10-document cutoff level. Since the overall German retrieval results are weaker for TREC-7 topics, I indexed the German collection with Okapi weighting and ran monolingual VSM with those topics. The average precision from this run is only 0.1543, much lower compared to 0.2273 for the SMART *ntc* weighting. However, since the objective of these experiments is not necessarily to obtain the best possible performance on a specific collection, I chose not to investigate the source of this performance decline any further.

These tables do reveal the utility of ADE for improving LSI performance in both monolingual and cross-language situations. Table 4.11 breaks down the results by topics by showing the number of topics for which each method obtained the best average precision. The numbers in each column do not add up to 53 because several queries do not have any relevant documents in the collections. Again, we see the effect of query-expansion by LSI and ADE on a number of queries in monolingual retrieval with such a small portion of the full dimensionality. In addition, ADE outperforms VSM 24 and 19 times on French and German monolingual retrieval, respectively. For cross-language retrieval, ADE's performance dominates the best of LSI and GVSM. An interesting observation is that the same query, the very first one in the set, puts GVSM on top in all four cases.

4.3.5 TREC English and French Collections

To further demonstrate the utility of ADE for cross-language retrieval, I ran it for retrieval across a pair of languages that are most commonly used in cross-language retrieval tests: English and French. The superset of the TREC AP 1990 collection from Section 4.3.3, the entire TREC AP collection, is the English collection that I used. This corpus consists of 242,918 documents and 210,580 terms after stemming with SMART. For French, I used the same collection as in Section 4.3.4; but, to match the pre-processing of English documents, I did stop-word removal (from a list of 355 words) and some elementary stemming. My French stemmer is implemented in the same way as described by Buckley et al. [20]. The resulting French collection has 156,875 unique terms, or 20% less than when no stemming or stop-word removal is used. Finally, I also de-accented all the words in both collections.

For training, I used two sources: the Hansard Corpus and the United Nations (UN) Parallel Text Corpus [50]. The Hansard Corpus contains the official records of the proceedings of the Canadian Parliament in English and Canadian French and is available on a CD-ROM produced by the Linguistic Data Consortium (LDC). I used the "set b" of the collection that was originally obtained by Bell Communications Research and spans from 1986 to 1988. The UN Corpus is the original source of the CMU UNICEF collection described in Section 4.3.2. I used Version 1.0 of the data CD-ROM, also produced and published by LDC. The documents in this CD come from the Office of Conference Services at the UN in New York and span the period between 1988 and 1993, and a large number of them are available in English, French, and Spanish versions.

I extracted parallel passages from the corpora by taking advantage of the SGML tags in the documents. In the Hansard collection, there are more than 220,000 parallel passages with no document boundaries. Had I treated each passage as a single document, the number of documents in the training collection would have been too large for the SVD package. To reduce this number, I indexed each passage as a single document and ran VSM similarity comparisons between consecutive documents. Then, I combined English passages that had a similarity score greater than 0.16 (a somewhat arbitrary threshold) and combined French passages according to their English parallels. I also omitted all passages that have fewer than twenty words in either the French or the English version. The resulting parallel corpora had more than 100,000 documents, about one-sixth of which was randomly selected for training. On the other hand, the documents in the UN collection can be so large that they contain 30–50 paragraphs. Thus, I only selected segments of the UN documents—from 2 to 20 consecutive paragraphs—to form my training documents. This process is the similar to that of Yang et al. [121] in their preparation of the CMU UNICEF collection. The resulting parallel collections had approximately 23,000 documents and I again randomly selected half of them for this experiment. The combined UN and Hansard parallel collections have 30,024 documents in each and they are stemmed and stop-word removed as described earlier.

I ran the baseline VSM retrieval on English and French collections with the SMART *Lnu.ltn* indexing scheme (see Table 2.2). The *Lnu* weighting on documents was developed by Singhal [106] and used by Buckley et al. [24, 20] and Braschler et al. [16] in their recent TREC experiments:

$$\frac{\frac{1+\log(tf)}{1+\log(\text{average } tf)}}{(1.0 - slope) \times pivot + slope \times number\ of\ unique\ terms}.$$

Here, I set *slope* to be 0.2 and the pivot to the average number of unique terms in a document for the collections, the same as in the experiments of Buckley et al. [24]. For the queries, I adopted the *ltn* scheme by Braschler et al. instead of the *ltu* scheme by Buckley et al. [24], because it is not exactly clear how the normalization factor u is to be used with a set of queries[1].

For the training documents, I used the SMART *ntc* indexing, which has worked well with TREC cross-language topics in my previous experiments. The 30,024 English documents contain 26,436 unique terms while their French parallels have 46,174. Notice that because of the smaller size of the training collections compared to the retrieval collections, the latter have far more terms than the former. However, most of the words in the TREC-6 and TREC-7 topics were common enough that they are found in the training documents. Using "dragon," I was able to find a little more than 2,200 dominant dimensions (8% of the full rank and 82% of the total variance) from the English and 2,000 (7% of the full rank and 65% of the full rank) from the French training corpora. The results from LSI and ADE with these computed dimensions are shown in Tables 4.12 and 4.13. I did not run GVSM here because previous experiments have shown it to be ineffective on large collections.

My TREC-6 cross-language results compare favorably with those by Buckley et al. [20], who matched English and French words directly if they are very close to each other in spelling. They also used local feedback after the initial cross-language run to achieve an average precision of 0.2408 for 13 of the 25 TREC-6 English queries on French documents. ADE already accomplishes this level of average precision without taking advantage of the closeness in spelling of the two languages and additional feedback enhancement (I did run this with a feedback method—local LSI—as described in Chapter 5, though). My monolingual baseline VSM runs had lower average precisions than theirs because I did not use their

[1]Recall from Section 2.3.3, the pivoted normalization factor "u" is created to match the observed relationship between retrieval documents' length and their chance of being relevant. This does not seem to have anything to do with the queries.

		Monolingual		Cross-language	
		English	French	Fre ⇒ Eng	Eng ⇒ Fre
VSM	AvgP	**0.3597**	**0.3661**	–	–
	P10	**0.5095**	**0.5048**	–	–
LSI	AvgP	0.2308	0.2402	0.1936	0.1965
	P10	0.3514	0.3619	0.2857	0.2952
ADE	AvgP	0.2999	0.2931	**0.2656**	**0.2582**
	%VSM	–	–	74%	71%
	P10	0.4524	0.4333	**0.4048**	**0.3571**
Montreal	AvgP	0.2895	0.3686	0.2560	0.3053
	%VSM	–	–	88%	83%

Table 4.12: Monolingual and cross-language results on the TREC English and French corpora with TREC-6 cross-language topics and SMART *Lnu* indexing; included for comparison are the results by researchers at the University of Montreal.

		Monolingual		Cross-language	
		English	French	Fre ⇒ Eng	Eng ⇒ Fre
VSM	AvgP	**0.4882**	**0.4056**	–	–
	P10	**0.6385**	**0.5250**	–	–
LSI	AvgP	0.3845	0.2975	0.3027	0.2667
	P10	0.5123	0.4286	0.4546	0.3786
ADE	AvgP	0.4216	0.3185	**0.3482**	**0.2978**
	%VSM	–	–	71%	73%
	P10	0.5815	0.4500	**0.4783**	**0.4179**
Montreal	AvgP	0.3202	0.2764	0.3245	0.2649
	%VSM	–	–	101%	96%

Table 4.13: Monolingual and cross-language results on the TREC English and French corpora with TREC-7 cross-language topics and SMART *Lnu* indexing; included for comparison are the results by researchers at the University of Montreal.

"SuperConcepts" approach, or an additional processing step after local feedback in these experiments.

For both TREC-6 and TREC-7 topics, several sets of comparable results can be found in the report by Nie et al. [77] at the University of Montreal in Canada. One of their best approaches involves the combination of a probabilistic translation model estimated from the Hansard collection and a large bilingual dictionary of over one million entries. They obtained average precisions of 0.2560 and 0.3245 on TREC-6 and TREC-7 French-English runs, respectively, and 0.3053 and 0.2649 on TREC-6 and TREC-7 English-French runs, respectively. These numbers are all listed in Tables 4.12 and 4.13 in the rows labeled "Montreal." While ADE's average precisions are higher than theirs in absolute value in three out of the four cases, they are lower in terms of the percentage of the monolingual VSM performance (also shown in the two tables). However, their low baseline results may just be one reason behind the high percentages.

4.3.6 NACSIS Test Collection

My last test collection for this chapter is the NACSIS Test Collection for Information Retrieval Systems 1 (NTCIR-1) [64], where NACSIS is the National Center for Science Information Systems in Japan. This collection consists of approximately 330,000 documents in either Japanese, English, or a combination of both. These documents are abstracts of academic papers presented at meetings hosted by 65 Japanese academic societies and cover a variety of topics such as chemistry, computer science, and linguistics. A document has several fields enclosed by SGML tags; these fields typically include ID, title, author, abstract, keyword, and the name of the academic society, each of which can be in both Japanese and English. The NTCIR-1 data has three parts: JE collection, which is also denoted “ntc1-je1” and contains the entire collection of mixed documents; J collection, which is also denoted “ntc1-j1” and contains 332,918 Japanese-only documents; and E collection, which is denoted by “ntc1-e1” and contains 187,080 English-only documents. I used only J and E collections for my cross-language experiments.

The retrieval topics for NTCIR-1 consist of 30 (topic0001–topic0030) used for training and 53 (topic0031–topic0083) used for testing in the First NTCIR Workshop held in Tokyo, Japan, in 1999. These topics are all in Japanese only and contain the following fields: title, description, narrative, concept, and field (i.e. academic field). Among the topics, 21 of the first 30 and 39 of the last 53 topics are usable for cross-language retrieval; the others were discarded because they have fewer than five relevant documents in the collection. In my experiments, I ran separate tests for the two topic sets for the purpose of simplifying comparisons with published results.

Relevance assessments are available for both topic sets against both ntc1-j1 (monolingual) and ntc1-e1 (cross-language). The judgments were carried out by the pooling method as described in Section 2.1 and had three grade levels: relevant (grade A), partially relevant (grade B), and non-relevant (grade C). As a result, there are two types of relevance judgment files available: In “rel1” files, only the grade-A documents are treated as relevant, while in “rel2” files, both grade-A and grade-B documents are taken as relevant. For my results, I used both types of relevance judgment files, and they seem to produce results that are consistent with each other.

In processing the documents in both languages, I extracted the text from the title, abstract, and keyword fields from them. The English documents were stemmed and stop-word removed through the SMART system, the same way I did for the TREC AP documents. The resulting English collection has 208,276 unique terms. The Japanese text is coded in EUC, or Extended UNIX Code, which represents each character in Japanese by two bytes. In order to be able to process the Japanese documents with my programs that handle only ASCII characters, I created another program that converts each two-byte Japanese character into a four-character "word" in ASCII. With no stemming or stop-word removal, the converted Japanese collection has 130,777 unique "words." I then indexed both collections with the SMART *Lnu* weighting. Because sentences in Japanese or Chinese contains consecutive characters with no word boundaries, IR researcher have tried using n-grams, where $n \geq 2$, to index collections in such a language. However, in my experiments I chose to use individual characters, or unigrams, as the basic indexing items (terms) in the Japanese collection, as I was curious to see how the term-association capability of LSI (and ADE) would do in this case.

For the topics, I extracted the title, description, narrative, and concept fields to form the queries. I discarded the concept words in English and acronyms from topics 0031–0083 as they contains both Japanese, English, and acronym concept fields. The Japanese queries were also converted into ASCII; they were then indexed with SMART *ltn* weighting.

The baseline monolingual VSM results are shown in the top rows of Table 4.14. At first sight, we see a downgrade of performance from the training queries to the test queries. I do not have an explanation for this, but, I have seen results by other researchers having a similar trend with the same query sets on this collection (e.g. [91]).

For monolingual LSI and ADE retrieval, the entire ntc1-j1 collection is too large for the SVD package to process. Therefore, I selected a subset of documents from it to create an LSI space. In fact, since I would use a pair of training corpora

			topic0001–0030		topic0031–0083	
			rel1	rel2	rel1	rel2
MLIR	VSM	AvgP	**0.3452**	**0.3295**	**0.2601**	0.2870
		P10	**0.5143**	**0.5333**	**0.4103**	**0.4974**
	LSI	AvgP	0.2555	0.2569	0.2413	0.2702
		P10	0.4524	0.4667	0.4000	0.4846
	ADE	AvgP	0.2807	0.2795	0.2576	**0.2874**
		P10	0.4571	0.4762	0.3949	0.4872
CLIR	LSI	AvgP	0.1327	0.1375	0.1162	0.1322
		P10	0.2429	0.2619	0.1462	0.1923
	ADE	AvgP	**0.1554**	**0.1606**	**0.1703**	**0.1898**
		P10	**0.2810**	**0.3048**	**0.2000**	**0.2538**
		11pt AP	**0.1734**	**0.1783**	**0.1870**	**0.2043**
CLIR Published	UMD	AvgP	–	–	0.1534	–
	Toshiba	11pt AP	0.2910	–	0.1820	–
	ULIS	11pt AP	–	0.1930	–	–
	Berkeley	AvgP	–	–	–	0.1925

Table 4.14: Monolingual and cross-language results on the NTCIR-1 collection for two set of cross-language topics and two levels of relevance judgments, with SMART *Lnu* indexing; included for comparison are published results from several research groups.

for cross-language retrieval later on, I proceeded to extract parallel documents from the J and E collections for training. Matching documents were identified by the same document ID numbers at the beginning of a Japanese and an English document. Since all English documents in ntc1-e1 were originally extracted from abstracts that are in both languages, nearly all of them had a translational counterpart in ntc1-j1, where documents are either Japanese-only or in both languages in their original form. Thus, the resulting parallel collections I obtained have 181,485 documents in each language, close to the size of the ntc1-e1 collection.

This size not only was still too big for LSI processing, but it also would have meant cheating had I used it for cross-language retrieval, since I could have obtained excellent results by finding the best documents in the Japanese training collection with VSM, and then identifying their parallel counterparts in the English collection. With these considerations, I decided to use only one-fourth, or 45,372 documents, of the parallel corpora for LSI and ADE training. I indexed these smaller parallel collections with similarity thesaurus weighting method from ETH as described by Equation 3.13, but the results I obtained for both monolingual and cross-language LSI retrieval runs were extremely low. I then applied the SMART *ntc* weighting on the training corpora, which had much better results compared to those from the ETH weighting. The matrices created have 33,553 Japanese terms and 84,554 English terms. It took "dragon" approximately 9.5 hours to compute 1,400 SVD dimensions (4% of the full dimensionality and 85% of the total variance) from the matrix of Japanese documents, and 13 hours to compute 1,200 dimensions (3% of the full dimensionality and 54% of the total variance) from the matrix of the English collection.

The results of monolingual and cross-language LSI and ADE retrieval were shown in Table 4.14. The 11-point average precisions of ADE are also included in the table for later comparisons with other published results. Even with this special collection, we see similar results as in the previous sections: ADE improves on LSI in both monolingual and cross-language retrieval.

In cross-language retrieval, scores of ADE compare closely to several published results (also listed in Table 4.14) that used language specific tools such as a bilingual dictionary or a machine translation system. For example, Oard and Wang [78] from the University of Maryland used the freely available "edict" Japanese/English dictionary to automatically translate the queries and obtained an average precision of 0.1534 (compared to ADE's score of 0.1703) for the 39 test queries with the same extracted topic fields as mine. This result is shown in the row labeled "UMD" in in Table 4.14.

Results by Sakai et al. [91] at Toshiba R&D Center—thus "Toshiba" in Table 4.14—are 0.2910 in 11-point average precision for the 21 training queries and 0.1820 for the 39 test ones with an automatic machine translation system. While their scores are higher than mine (0.1734 in 11-point average precision) for the training queries, their test query results are actually lower than the 11-point average precision of 0.1870 that I obtained from ADE. This may be partially due to the fact that they tuned the dictionary of their MT system for the training queries by adding new phrases into it. They also tried the local feedback method on top of the initial cross-language retrieval run; I present results using a similar approach in Section 5.3.4. Fujii and Tetsuya [45] at the University of Library and Information Science in Japan used a compound word translation method with a bilingual dictionary on NTCIR-1. They achieved an 11-point average precision of 0.1930 with the first 21 queries and "rel2" judgments (shown in the row labeled "ULIS" in Table 4.14). My result of 0.1783 is obtained with similar settings except for the topic fields being used for query—they used only the description. Finally, one of the best sets of results reported in the First NTCIR Workshop was by Chen et al. [27]. With non-interpolated average precisions as high as 0.3755 for the 39 test queries, they achieved their results with a bilingual lexicon that was built from aligning and matching Japanese and English keyword fields in NACSIS documents. They noted that their method is only applicable "when the documents containing keywords in both the source and target languages are available for creation of bilingual lexicon." Their pure MT-based run had an average precision of 0.1925 with rel2 judgments on the 39 test topics (shown in the row labeled "Berkeley" in Table 4.14); my result at 0.1898 is close with similar fields selected from the topics and the documents for indexing. In summary, all the published results discussed here were obtained with the incorporation of certain type of language-specific knowledge; for example, the ability to perform word segmentation on Japanese queries is needed for the machine translation system to work. On the contrary, I accomplished comparable results with ADE merely by unigram indexing; this clearly demonstrates the ability of ADE, or if possible, a high-dimension LSI, to derive term associations and translations from bilingual corpora.

4.4 Conclusion

The approximate dimension equalization approach mimics the effect of a high-dimension latent semantic indexing with just a few computed initial singular values and vectors. The dimensions not being computed are used implicitly through the manipulation of singular values of the original matrix. When the collection size is small, ADE is able to approximate the best LSI (and best overall) results with fewer computed dimensions than LSI. When the collection size gets large, ADE seems to improve on LSI and inch close to VSM every time with a limited number of dimensions. ADE becomes especially useful in cross-language retrieval, where VSM is not applicable; it obtains state-of-the-art results on some of the standard test collections.

Chapter 5

Local LSI Sampling

In the previous chapter, I described an approximation algorithm to LSI for retrieval on a reduced-dimension semantic space. The purpose of this ADE algorithm is to mimic the effect of using a large number of equalized dimensions of the training space when we can only find very few significant ones with current limited computation resources. There is another possibility for down-scaling the problem, however: we can make the SVD more computable by using a smaller training collection through the sampling of documents.

One may argue that the training collections need not to be as large as the current retrieval collections are. But, as I pointed out earlier, training is usually done on the retrieval collection itself in monolingual retrieval. So, the training size does keep pace with the retrieval size in this case. Otherwise, we would use a separate training collection, possibly in monolingual retrieval and necessarily in cross-language. Then, a decrease in performance is often observed as the term-term correlations in the training corpus may not entirely reflect those in the retrieval corpus. To cover the term associations more thoroughly, more training documents are probably needed, resulting in, again, a growing size of the training collection.

In this chapter, I examine a sampling approach to reducing the LSI training collection size that is closely related to local feedback. The original idea, known as *local LSI*, was introduced by David Hull in his dissertation [61]. I use a slightly different method than his and extend the local sampling approach to cross-language information retrieval. I show that this local method not only reduced the computational requirement of SVD tremendously, but it also shows significant improvements in retrieval performance, both monolingually and cross-lingually. But, first, in Section 5.1, I discuss two basic sampling approaches to LSI approximation that have been examined before and have yielded only mediocre retrieval results.

5.1 Global Random Sampling

Traditionally, when a training collection is too large to analyze via the SVD using available computing power, we can sample a number of documents from the original collection and compute the SVD on the sub-collection. This is analogous to the approach of selecting good terms (row vectors of the term-document matrix) for expansion that is used in GVSM, the similarity thesaurus method, and the information space method. While the term-selection approach has the flavor of discarding words that may not have their connections with other words properly established by the training corpus, the document-sampling method intends to capture such connections using fewer documents than given.

Two different approaches to sampling are:

- select raw documents from the collection and index the sub-collection, or
- select columns (document vectors) from the term-document matrix of the collection.

In the first approach, which I refer to as "text sampling" henceforth, the weight of the same term in the same document could be very different in the sub-collection than in the original collection. This is due to the fact that most of the current indexing functions incorporate the collection size or some value averaged over a collection in their formulations. The result is little, if any, mathematical connection between the term-document matrix of the sub-collection and that of the original collection.

In terms of query expansion, we can view text sampling as using a smaller example set to learn a term-correlation model. The procedure would be to build an LSI space on the sub-collection, expand the queries with the term-associations found with SVD, and compare the new queries to the retrieval collection. This corresponds to the third interpretation, or the query-expansion view, of LSI described in Table 3.3. As we shall see in Section 5.2, this view of LSI training suits the idea of local LSI better than the other two in the same table.

In the second "vector sampling" approach above, we begin by selecting a sample of document vectors from the matrix of the original training collection A to form a new matrix S. Then, we find the singular value decomposition of S:

$$S = U_S \, \Sigma_S \, V_S^T .$$

In terms of linear algebra, the column space of S is a subspace of the column space of A. This means U_S, being the basis of S, captures some dimensions of A, but not necessarily the dominant ones (in A). Were we able to do this—sampling documents so that the most dominant dimensions of the sampled space captures (or spans) the subspace spanned by the most significant dimensions of the original space—the LSI scalability problem would have been solved. In other words, our objective in global sampling is to make $\bar{S}_k$ to be as close to $\bar{A}_k$ as possible. Then, the similarity score computed from $(\bar{S}_k^T \vec{d}\,) \cdot (\bar{S}_k^T \vec{q}\,)$ would be very similar to that computed from $(\bar{A}_k^T \vec{d}) \cdot (\bar{A}_k^T \vec{q})$. In the weighted sampling approach that I discuss shortly, we see that it emulates the dominant dimensions of the original space with the dominant dimensions of the sampled space with high probability.

In monolingual retrieval, if the sampled documents are directly from the retrieval collection, the documents not included in the sample S' are "folded in" by projecting them onto the reduced sample space S_k as in Equation 3.21:

$$U_S\, I_k\, U_S^T S'.$$

Note that if the sampled documents are folded into the reduced sample space, they are exactly the reduced sample space S_k:

$$U_S\, I_k\, U_S^T S = U_S\, I_k\, U_S^T\, U_S\, \Sigma_S\, V_S^T = U_S\, I_k\, \Sigma_S\, V_S^T = S_k.$$

The query $\vec{q}$ can be folded into the sample space as well before making a similarity comparison with the (projected images of) the documents. This corresponds to the second interpretation of LSI described in Table 3.3.

The most straightforward global sampling procedure, and one commonly used in work on LSI, is to select document vectors to include in the sampling matrix S uniformly at random from the full set of document vectors. This approach has the appeal of simplicity and can be intuitively justified by arguing that the "latent" structure available in the matrix is not embodied by any single document vector; taking a sample of document vectors should still reasonably represent the most important components of meaning.

Nonetheless, uniform sampling lacks a formal justification and, as I show below, can be improved using an approach that samples based on document-vector length. Further, it is possible to construct matrices where uniform sampling produces poor results with high probability. For example, for a large matrix with only one nonzero components, uniform sampling would most likely select those columns with all zeros in them than the one with the only nonzero entry, and thus misrepresent the structure of the original matrix.

5.1.1 Weighted Sampling

I next describe the weighted sampling approach invented by Frieze et al. [43], a detailed study of which is given by Jiang et al. [63]. As mentioned earlier, the objective of global sampling is to make the k-dimensional subspace of the sampled space as similar to the k-dimensional subspace of the full collection space as possible. Given a target number of orthogonal dimensions k, a sample size s, and a probability δ, we can first compute the error bound

$$\varepsilon = 2\sqrt{k/(\delta s)}. \tag{5.1}$$

It is assured that, with at least probability $1 - \delta$, the sample space will be within ε of the true k-dimensional SVD, as described below. Then, we carry out the following steps.

(1) For each column (document) $A_{(j)}$ of A, compute the square of its vector length $(A_{(j)})^T A_{(j)}$.

(2) Select s columns of A with probabilities proportional to their length squared as computed in the previous step. Sampling is performed with replacement, meaning that a single document vector can appear multiple times in the sample. A matrix S of size $m \times s$ (terms by sample size) is obtained.

(3) Normalize the document vectors in S to unit length.

(4) Compute the SVD of S and use its first k left singular vectors $(U_S I_k)$ as a basis for projecting document or query vectors into the range of S_k.

The algorithm runs in time $O(ms^2)$ plus the number of nonzeros in the term-by-document matrix [63].

The reason for normalizing document vectors to unit length in Step (3) is that this is exactly the normalization that implies that for every unit length vector u, the expected length squared of $u^T S$ is precisely $|u^T A|^2/||A||_F^2$ (this can be shown by direct calculation), i.e., the ratio $|u^T S|/|u^T A|$ is a constant in expectation, independent of u. This means that the sampled matrix S behaves similarly to A with regard to multiplication by unit vectors.

Now, from standard properties of SVD, the first left singular vector u_1 of A is a unit-length vector u maximizing $|u^T A|$. So, crudely speaking, the most dominant singular vector of S should give us a good approximation to the most dominant singular vector of A. To establish the correctness formally, we need to bound the variance; this can be done and uses properties of the probability distribution and the normalization selected in the algorithm. The same property holds for the other singular vectors.

By the same criterion as used in Theorem 3.1, the weighted sampling algorithm provides the following guarantee [43, 34].

Theorem 5.1. *Given a matrix A and a matrix $S = U_S \Sigma_S V_S^T$ that consists of columns of A obtained through the weighted random sampling algorithm, we have*

$$||A - U_S I_k U_S^T A||_F^2 \leq \min_{\text{rank(D)=k}} ||A - D||_F^2 + \varepsilon ||A||_F^2 = ||A - A_k||_F^2 + \varepsilon ||A||_F^2,$$

with probability at least $1 - \delta$, where $\varepsilon = 2\sqrt{k/(\delta s)}$.

In words, when the document vectors in A are projected onto the k dominant dimensions of the sampled space, the total error factor we get compared to directly using the k dominant dimensions of A (i.e., the SVD of the full collection) is bounded by ε times the norm of A. Note that the expression for the approximation has the same form as in Theorem 3.1, but instead of using the true singular vector matrix U, the algorithm finds an approximation U_S; the latter can be found more quickly as we use a smaller document matrix.

Theoretically, the error bound in Equation 5.1 can be very large. For example, with $s = 10{,}000$ documents, $\delta = 0.95$ probability, and $k = 300$ dimensions, the error is 35%. To be assured of an error factor of 2% for a 300-dimensional SVD with 95% probability, a 3.2 million document sample is needed. (Notice how the error bound calculation is independent of the size of the full collection. So, be it ten million documents or a hundred million documents, we would need to sample 3.2 million of them to obtain the desired small error and large probability.) If this bound were an accurate reflection of the precision of the weighted sampling method, the method would be of little practical value.

The real error, however, can be measured by computing $\|A - U_S I_k U_S^T A\|_F^2$ and $\|A - A_k\|_F^2 = \|A - U I_k U^T A\|_F^2$. The problem is this is not practical because of the sizes of $U_S I_k U_S^T A$ and $U I_k U^T A$ will be very dense—in the billions of nonzeros even for A with less than $100,000$ documents [63]. However, as shown in Appendix B.6, the approximation error can be rewritten as

$$\begin{aligned}
&\|A - U_S\, I_k\, U_S^T A\|_F^2 - \|A - U I_k U^T A\|_F^2 \\
&= \|A\|_F^2 - \|I_k\, U_S^T\, A\|_F^2 - (\|A\|_F^2 - \|I_k\, U^T A\|_F^2) \\
&= \|I_k\, U^T\, A\|_F^2 - \|I_k\, U_S^T A\|_F^2. \qquad (5.2)
\end{aligned}$$

So, $(\|I_k\, U^T A\|_F^2 - \|I_k\, U_S^T A\|_F^2)/\|A\|_F^2$ is the observed (relative) approximation error. The matrix $I_k U^T A$ is the set of training document vectors determined by “fold in” in the full SVD and $I_k U_S^T A$ is the set of training document vectors determined by “fold in” in the sampled SVD. If we train on the retrieval collection and have already folded in the retrieval documents, the error calculation just involves a single scan through the set of folded in document vectors.

Experimental results described later in Section 5.3.1 show that we can achieve low observed errors with small sample sizes using the weighted sampling procedure.

5.2 Local LSI Feedback

The weighted sampling approach just discussed has a solid theoretical foundation and good experimental bounds in terms of sample error. However, there are still two main problems with this method (and global sampling in general).

First, as I pointed out in last section, even though the error bound in Equation 5.1 does not depend on the size of the full collection, it does increase with k, the number of desired dimensions. What we have learned about LSI from the previous chapters is that the number of dimensions needed (from the training space) often increases with the retrieval collection size. Therefore, when the collection size is huge, to keep up with the dimension requirement (k) and to maintain a decent error bound (ϵ) and a high probability ($1 - \delta$), we have to sample a large number of documents, the SVD of which would again be very difficult to compute.

Secondly and more importantly, retrieval results of using sample collections have mostly been worse than those from using the full collection. In the study by Jiang et al. [63], part of which is described later in Section 5.3.1, the average performance of ten runs of weighted random sampling is below the performance of LSI on the full collection, and even below the average performance of uniform random sampling. In addition, both sampling approaches perform much worse than the baseline VSM result.

The same can be argued for the approximate dimension equalization method as well: since by any means we need to compute some initial singular triplets of the collection matrix, the computational requirement can still exceed system limitations when the corpus contains millions or even more documents. Thus, we need a method that can scale even better than ADE, improves retrieval performance, and works with cross-language applications. Local LSI, an idea first introduced by Hull [61], meets all the above requirements with a small additional cost during the process of retrieval.

5.2.1 Basic Concept

The original local LSI method described in Hull's thesis [61] is as follows: apply the singular value decomposition to the set of documents that are known to be relevant to the query; then, all the documents in the collection can be folded into the reduced space of those relevant documents, which we say forms a *local region* of the query. The rationale behind local LSI is straightforward: When we use SVD to process through tens of thousands of documents and find term-associations, we hope the result is a fairly sophisticated LSI structure good enough for information retrieval. But, the few number of dimensions we can compute is often not enough for distinguishing between words relevant and non-relevant to those that are in a particular query. So, instead, we can turn our focus from the entire corpus to the query itself by examining only the documents that contain the words related to those in the query. The LSI term-association structure built from those documents would certainly help that particular query most.

Local LSI uses the same assumptions as relevance feedback [89]: the documents relevant to a query are similar to each other but dissimilar to those that are not relevant (see Section 2.4.1). In this view, discarding the (mostly) non-relevant documents from the training space has an effect similar to dimension reduction—the dimensions occupied by those documents that are not in the subspace the query is in, and so we "reduce" them automatically by not even considering them. The relevant documents in the discarded set will still get adequate representation in the local space because of their assumed similarity to the known relevant documents.

Hull [60] first proposed the local LSI method in the context of text routing problem: documents are either relevant or non-relevant to a specific information need, and a classification method is to be developed to predict which document belongs to which group based on an existing training sample. This pre-judged set of documents is generally not available for an ad-hoc query, so local LSI in its original arrangement is not applicable to the information retrieval task that we are trying to solve. However, we can borrow the idea from the local feedback method discussed in Sections 2.4.1 and 3.3.3.

Recall that relevance feedback is a semi-automatic process that reformulates (or expands) a user query based on what the user has indicated as relevant from an initial retrieval. Local feedback then extends the idea by assuming the top k documents from the initial retrieval are relevant and using them for query expansion. This way, the user does not have to spend time examining the documents returned from the preliminary retrieval, while systems that utilize this technology also get a boost in retrieval performance due to additional words from the top documents that were not in the original query. The local LSI approach that I study in this chapter employs exactly the same idea: train (in this case, compute an SVD) on the top documents from an initial retrieval.

A combination of both local feedback and local LSI was applied by Schütze et al. [103] on their document classification experiments. Local feedback was used to expand the initial query vector using a modified version of Rocchio's formula [22], and then an SVD was performed on the top 2,000 documents retrieved by the expanded query (which comprise a "local region" to the query). Thus, query expansion is mainly done by local feedback, while LSI is used to create a reduced representation of the expanded query that can be better used by an statistical classifier. I used a simpler (and different, in a sense) approach—computing LSI directly on the top documents from the initial retrieval, because I believe the LSI alone is sufficient for query expansion (see Section 3.4.1).

5.2.2 Procedure

The first step in local LSI as well as other local feedback methods is an initial retrieval on the training corpus that returns a set of documents to be used for further analysis. On paper, any of the known IR methods such as VSM, GVSM, or LSI can be used to accomplish this. In practice, since the objective of training on documents in GVSM and LSI is to build term associations for query expansion, there is redundancy if I use them for the initial retrieval in local LSI. Besides, I do treat local LSI as a variation of the local feedback methods: it is a performance booster for VSM. Hence, I generally complete the initial screening step using VSM, with which efficiency is also achieved since VSM retrieval can be accelerated with an inverted index data structure [94].

The next step is to select the top documents from the initial screening. Here, in addition to vector sampling used in weighted random sampling, text sampling can also be used: select texts of top returned documents directly from the corpus. In this particular approach, we are free to choose an indexing scheme (possibly different from the one used on the full collection) to build local term correlations. When choosing an indexing method for the entire collection, we want to use one that helps to return the best top documents for local training or feedback; that indexing scheme, however, is not necessarily equally good at building term associations. For example, in recent years, the best indexing schemes for the TREC collections tend to give a long document a higher probability of being retrieved than a short one [107]. This means the same query term is more “important” in a long document than in a short one. On the contrary, the similarity thesaurus indexing function considers a short document to be more reflective of the meaning of a word than a long one is [82]. Hence, a different indexing scheme may be used for building the local term associations than the one for retrieving the best possible top documents. On the other hand, in the global sampling methods where only vector sampling is used, we can index the original training corpus directly with a scheme that is better at building a term-association structure.

In summary, with text sampling, local LSI is carried out by the following steps:

(1) Index the training collection with the best known term-weighting scheme for retrieval on its type of text/corpus.

(2) Given a query, perform conventional VSM retrieval on the training collection.

(3) Index the top k documents returned by the initial retrieval, with perhaps a weighting scheme better suited for creating term-associations.

(4) Compute an SVD on the resulting matrix and project the query onto the reduced LSI space.

(5) Compare the projected query vector against the document vectors in the retrieval collection and return the top documents.

With vector sampling, Step (3) above should be replaced by "select the vectors of the top k documents returned by the initial query to form a new matrix."

The number of top documents chosen to be used for LSI training, k, is usually collection dependent and determined experimentally. The size of the local training set for a query should not be too large compared to the actual number of relevant documents; otherwise, too many non-relevant documents in the training set would probably hurt retrieval performance. But, prior to retrieval, a query's number of relevant documents in a collection is not known, and some number has to be picked based on other information such as the average number of relevant documents per query for past queries.

An advantage of local LSI over local feedback is that the top k documents are not actually declared "relevant"—they are only used to create a useful reduced-dimensional space for retrieval. Therefore, including both relevant and non-relevant documents in the local training set would not hurt local LSI (in a sense, it may actually help—see Section 5.3.1) as much as it would local feedback. In other words, we have more "freedom" in choosing k for local LSI than for local feedback.

In previous experiments with relevance feedback, researchers have used k in the range of 10 to 1,000 for the information retrieval tasks [1, 2, 119], and even more for the text routing task [22, 103]. Efficiency again can be a factor in choosing the number k. For example, in their local feedback experiments, Buckley and Salton [21] found that using the top 1,000 documents took twice as much time as using the top 200. In this dissertation, I follow the study by Allan [2], who experimented with several factors including the feedback size that affect the performance of relevance feedback. In results he reported, retrieval performance usually peaks within the range 75 to 100 feedback documents. In the experiments described later in Section 5.3, the feedback sizes that I used ranged from 20 to 200, but I generally found improvement in retrieval performance of local LSI over other methods regardless of the particular size I choose.

Notice the above procedure does not project the retrieval corpus onto the local training space because fold-in here would be too costly in time when it is no longer a pre-processing step. Moreover, assuming the collection is already indexed with the best term-weighting method for retrieval, fold-in will only distort the matrix structure since the local space is much smaller than the collection space. Therefore, mathematically, while the fold-in process involves computing $I_k U_S \vec{d}$ for all documents in the retrieval collection, the local LSI method I propose is

$$\mathrm{Sim}_{Local-LSI-ML}(d,\, q) = (\vec{d}) \cdot (U_S I_k U_S^T \vec{q}\,),$$

where $S = U_S\,\Sigma_S\,V_S^T$ and S is the term-document matrix created from the top k documents returned from an initial retrieval. In other words, it uses the interpretation of LSI that is listed on the third row of Table 3.3. The similarity comparison between $\vec{d}$ and $U_S I_k U_S^T \vec{q}$ is accomplished by mapping the terms (rows) in the local region (S) into the same terms (corresponding rows) in the global space (A).

A technique similar to the local LSI process I just explained was implicitly used by Newby [76] in his TREC-7 experiments. Because of the large size of the TREC corpus being used, he was unable to make a full eigensystems analysis of the term correlation matrix for his information space method, which, as I explained in Section 3.4.2, is very similar to LSI. Instead, he ran the analysis on a sub-matrix of the entire term correlation matrix that corresponds to terms both selected from the TREC topics that he would test with and additional ones that frequently co-occur with selected ones. This way, he kept the total number to terms to be considered down to what can be easily processed by the Ispace system. Those terms, in the terminology of this Chapter, are just the terms in the local region of the topics, or the queries. The main difference between Newby's method and local LSI, as we can easily see, is that instead of selecting particular documents or columns of the term-document matrix, he essentially selects terms or rows of the term-document matrix[1].

5.2.3 Cross-Language Local LSI

Like other corpus-based methods, local LSI can be applied to cross-language retrieval. Given a pair of parallel collections, we face two possibilities here:

1. Pre-translation local LSI

    - Find the top documents from the training corpus in the language of the query using VSM.
    - Find the matching training documents from the collection in the language of the retrieval documents.
    - Perform cross-language LSI using the two local document subsets for training.

2. Post-translation local LSI

    - Perform cross-language retrieval using LSI, ADE, or any other applicable method that "translates" the query into the language of the documents.

[1]Even though the Ispace method works on the term-term matrix C, we can think of C as the product of a term-document matrix A and document-term matrix A^T, and then selecting the i-th and j-th rows of A is the same as selecting the element of C at i-th row and j-th column.

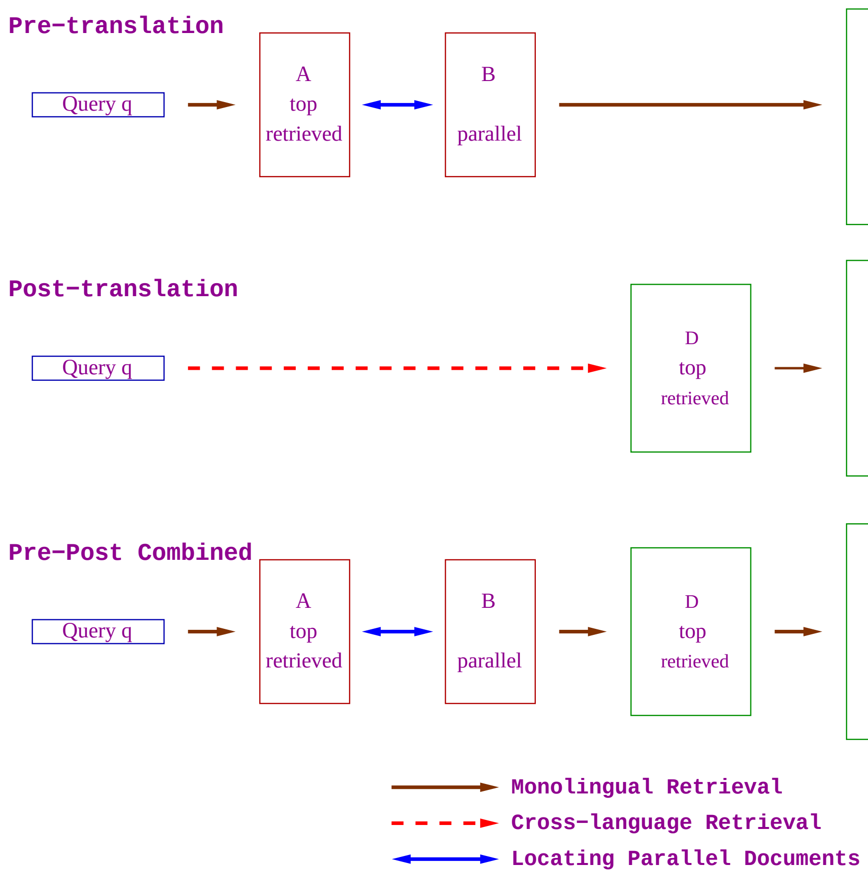

Figure 5.1: Local LSI can be applied before or after query translation, or a combination of both in cross-language retrieval.

- Expand the translated query by running local LSI on the top retrieved documents.
- Perform monolingual retrieval with the expanded query.

These two approaches are illustrated in Figure 5.1. In the first option, the best documents are selected for building cross-language term associations. In other words, we attempt to make the translation to be as accurate as possible. In the second choice, we hope the top retrieved documents are a good source for finding more words in the other language related the query. The functionality of local LSI is basically the same as in the case of monolingual retrieval; the cross-language work is done through some other algorithm. One such algorithm, of course, can just be the pre-translation local LSI. This third approach is also illustrated in Figure 5.1 and described here:

3. Combined pre-translation and post-translation local LSI

    - Perform cross-language retrieval using pre-translation local LSI.
    - Expand the translated query by running LSI on the top retrieved documents.
    - Perform another monolingual retrieval run with the expanded query.

These three options are very much like three cross-language local feedback approaches evaluated by Ballesteros and Croft [5]. In their experiments, they employed a machine readable dictionary (MRD) for word-by-word and phrasal translation of the queries. Local feedback was done before automatic query translation to enhance the translation accuracy (i.e. creating more related words to translate), or after it to reduce translation ambiguity (i.e. down-weighting the irrelevant translations). They also applied local feedback both before and after translation to have the best of both worlds.

The three proposed cross-language local LSI methods differ from those of Ballesteros and Croft mainly in the way query translation is accomplished. In the pre-translation approach, local LSI is used both for query expansion and translation, while in contrast local feedback is only used for expansion with translation being done through MRD lookups. In the post-translation approach, local LSI can be more like local feedback: the first cross-language retrieval step of local LSI can simply be achieved through query translation via MRD.

The cross-language local feedback method adopted by Yang et al. [121] and by Davis and Dunning [30] lies somewhere in between the pre-translation and post-translation approaches. Like pre-translation, they retrieve top documents from the training collection in the language of the query; like post-translation, they expand (replace, actually) the query with documents from the other training collection that are parallel to those top documents (see Section 3.3.3 for details). In a sense, this local feedback approach is closer to the pre-translation local LSI than other methods.

I tested both the pre-translation and post-translation local LSI methods in the cross-language experiments described later in the chapter. The pre-translation approach appears to be the simplest of the three choices since it reduces the training space in the first step of the procedure. Moreover, it retains the elegance of LSI in combining query expansion and translation in one process. In contrast, the post-translation local LSI approach would merely be a repetitious demonstration of monolingual local LSI, as I could potentially use any cross-language method for its screening step. Like all local feedback methods, its final performance would also rely very much on the particular baseline cross-language method it uses. Finally, the combination of pre-translation and post-translation local LSI seems to be an attractive direction to go. However, since the main objective of my study is to show the translation capability of local LSI, the focus will be on pre-translation local LSI in this dissertation. In addition, because the results show that pre-translation local LSI is already achieving great cross-language results, another layer of feedback process after translation is not urgently needed for efficiency (speed of retrieval) concerns.

5.3 Experimental Results

To study the application of the local LSI in an information retrieval setting, I performed a set of retrieval experiments on test collections that I used to study ADE. I ran experiments only on medium to large collections because methods like VSM and LSI already achieve impressive results on small collections such as Cranfield. Besides, Xu and Croft [119] noted that local feedback methods do not work well with some of the standard small test collections because of the fewer relevant documents for each query in those collections.

In this section, with many existing results from Chapter 4, I compare the performance of local LSI with the baseline VSM and LSI. I focus my experiments mainly on cross-language retrieval because there are relatively few experiments on relevance feedback in that area compared to monolingual retrieval. I am also able to compare the performance between global sampling and local LSI on the first corpus I study—the AP 1990 collection.

Again, most of the experiments described in this Chapter are conducted on the SGI machine "dragon" (see Section 4.3). The detailed information of the collections used here are the same as listed in Table 4.3.

5.3.1 TREC AP 1990 Collection

The first set of data is obtained from the TREC AP 1990 collection as described in Section 4.3.3. I begin with this collection because there are not only results by VSM, LSI, and ADE, but also results by the two global sampling methods [63] for comparison.

Uniform random sampling and weighted random sampling were performed on the term-document matrix indexed with SMART *ntc* weighting. The baseline

VSM achieves an average precision of 0.2132 with TREC-4 ad-hoc topics 201–250. In Section 4.3.3, I found that the TREC-4 ad-hoc topics work better with Okapi weighting on this collection. But, since the objective here is to demonstrate the relative performance of local LSI in comparison to global sampling and other methods such as baseline VSM, I did not run the global sampling algorithms on the Okapi-indexed matrix of AP 1990 corpus.

Details of the sampling experiments are found in the report by Jiang et al. [63]. Briefly, sample size 10,000 was chosen to be the focus of the study and repeated ten times with both uniform and weighted sampling, because it represented a relatively small fraction (about 1/8) of the original document set and still gave comparable performance compared to running LSI on the entire collection. Sampling was done with replacement, so a particular document could be sampled multiple times.

For each sampled matrix, an 1,000-dimensional SVD was computed and observed errors were measured using to Equation 5.2. Reproduced from the report by Jiang et al. [63], Figure 5.2 summarizes the errors over the ten 10,000-document samples generated by both global approximation algorithms. The samples producing the maximum and minimum error at 900 dimensions are shown for both uniform and weighted sampling. We can clearly see that weighted sampling produces approximations with greater accuracy and less variance than uniform sampling. Notice approximation accuracy decreases with increasing numbers of dimensions because we are attempting to estimate smaller components of variation (singular values) based on the same size sample.

Although the error results illustrated in Figure 5.2 provide a compelling case for the use of weighted sampling over uniform sampling, retrieval performance using the sampling methods paints a less clear picture. For the two global methods,

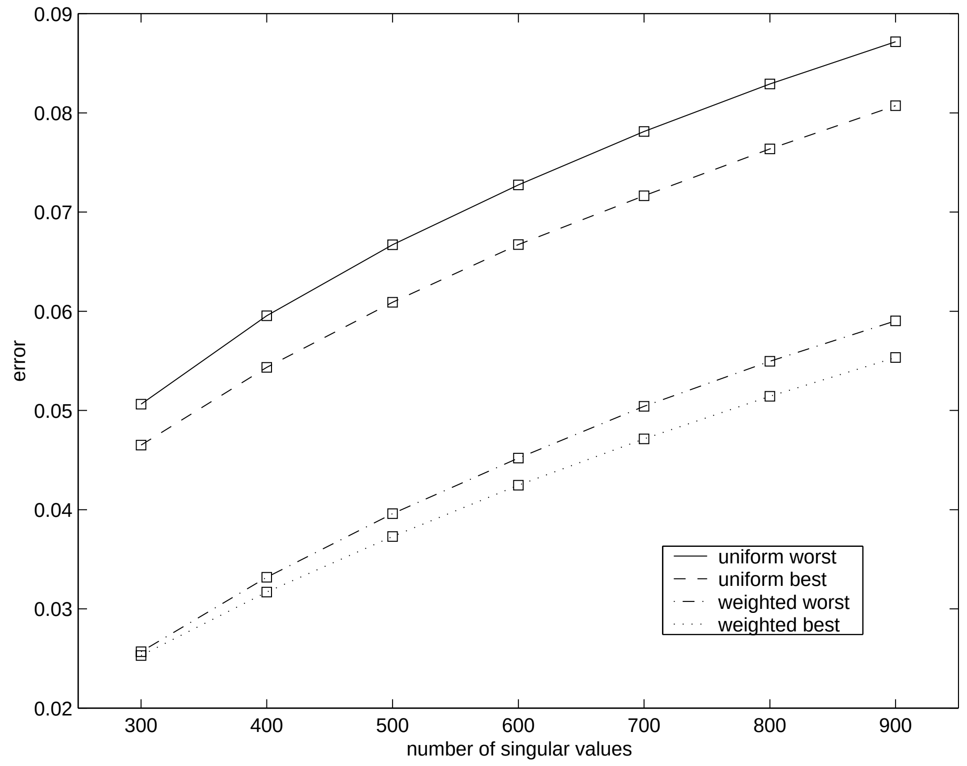

Figure 5.2: Approximation accuracy for a 10,000-document sample size declines for larger sets of singular values, but the weighted sampling technique consistently produces more accurate approximations.

I computed 1,000-dimensional SVD's of the sample matrices, performed retrieval on the resulting latent semantic subspace, and measured performance at dimensions from 300 to 1,000 by increments of 100. Figure 5.3 gives the average of the aforementioned ten runs of 10,000-document samples with uniform and weighted sampling on the AP 1990 collection with 50 TREC-4 ad-hoc topics. The graph also shows the performance of VSM and that of LSI without sampling for reference. We see that the average performance of weighted sampling is somewhat surprisingly below that of uniform sampling, which is, again, surprisingly close to that of LSI on the full collection. However, both sampling methods as well as the full LSI all lie well below the baseline VSM.

In the same figure, I included three plots for the local LSI method. These

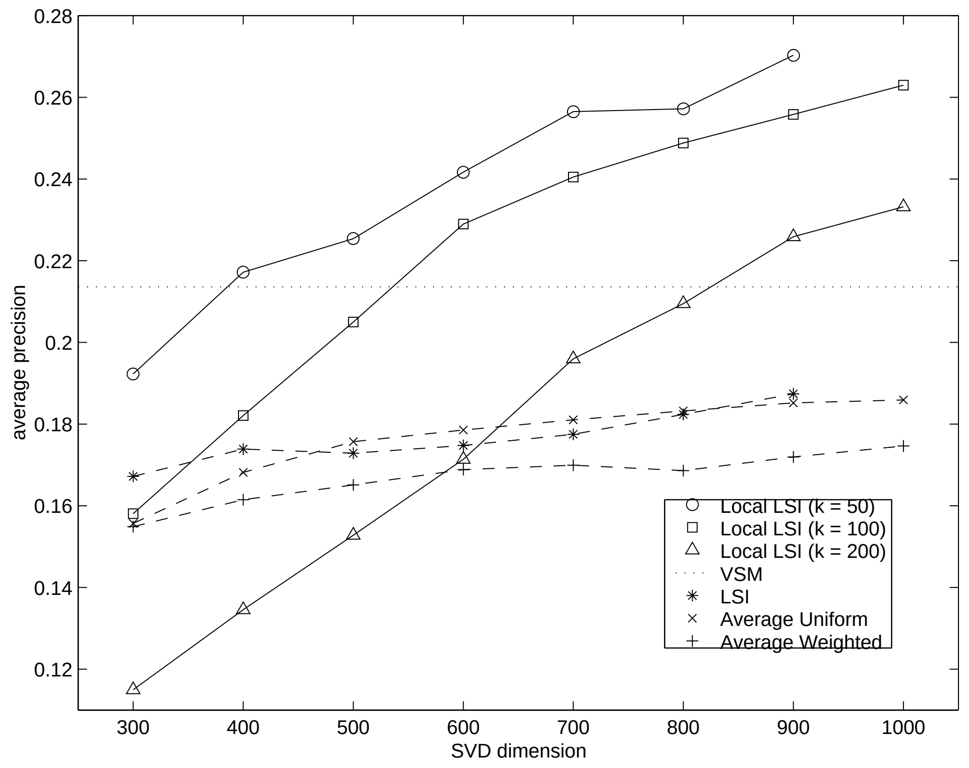

Figure 5.3: Comparison between local and global sampling methods on the AP 1990 collection: with more dimensions, the average precisions of local LSI not only increase much more sharply than global methods, but they also climb above the baseline VSM result.

results were obtained in the following way: For example, for the feedback size of 200, I extracted the texts of each query's top 200 documents from VSM retrieval for training. I indexed all the top documents for all queries as a whole collection with ETH's similarity thesaurus (see Section 3.3.2) weighting scheme, which I learned from experience worked well to establish term-term correlations. The feedback size (200) and the number of queries (50) together gave a total number of 10,000 documents in the combined training set, though the actual total is 8,418 due to removed duplicates. This number provides a close comparison between local LSI and the two global sampling methods with 10,000 samples.

The reason that I indexed all the top documents together was mainly for efficiency, because running SVD's on 50 small matrices may be a simpler task for a computer, but not so for human beings who have to create 50 indexed collections. Another reason is that since the job of LSI is to figure out the "proper" structure of word associations in the texts, we would want to train it with not just the relevant documents but also others, so to help it to better distinguish content-bearing words and other words. The top documents of other queries may just be the perfect fit for this purpose, since we save the effort of doing it for every query in one computation step. In a real-life situation, we may want to add a few of the documents not in the top returned list for "negative" training. This is similar to Rocchio's relevance feedback formula depicted in Section 3.3.3, which subtracts the vectors of the known (or assumed) irrelevant documents from the query vector. In this case, the local LSI approach is more elegant: let SVD to figure out the relevance; what matters is enough (assumed) relevant documents are in the training set and the set is sufficiently small.

I also ran local LSI with feedback sizes of 50 and 100, which resulted in a total of 2,304 and 4,488 documents, respectively, for 50 queries. I computed 1,000-dimensional SVD's on the three feedback matrices and performed LSI retrieval. In Figure 5.3, we see that the local LSI plots are rising much more sharply than those of the global sampling methods. For a similar sample size, the local LSI with 200-document feedback outperforms LSI at 700 dimensions and even VSM at 900 dimensions. With a smaller feedback size of 100, the results are better still. This shows local LSI not only reduces the computation of LSI, but also improves retrieval effectiveness. In the same figure, we also see the improvement of retrieval performance with decreasing sample size. This is a positive sign because a smaller training set means less training time.

To truly show the performance advantage of local LSI, I need to run and compare it against the best VSM result I obtained for this collection. So, I tried it with the Okapi weighting, which, from Section 4.3.3 gave an average precision of 0.2964 for TREC-4 topics. This time I used vector sampling—select document vectors directly from the Okapi-indexed document matrix. I computed SVD's on combined documents gathered by using feedback sizes from 20 to 100 in increments of 20. For comparison, I implemented the basic local feedback schema as described in Section 3.3.3, and set the Rocchio weights of Equation 3.16 to $\alpha : \beta : \gamma = 1 : 1 : 0$, the same as Xu and Croft [119] did. Figure 5.4 shows these results in terms of varying feedback size. For local LSI, I selected a specific dimension at which to compute LSI similarity, and plotted average precision for that dimension against selected feedback sizes. For local feedback, it is a simple plot of average precision versus feedback size. The baseline VSM is shown as the dotted straight line in the middle. We see local LSI outperforms VSM and local feedback at most combinations of feedback size and SVD dimension. We cannot, however, draw the conclusion that local LSI is better than local feedback, because the local feedback method I have implemented corresponds to its simplest formulation. There are many techniques and heuristics that can be added onto Rocchio's formula to enhance its performance. For example, local context analysis [119] is an advanced technique that selects "concepts" (noun groups) from the best matching passages (fixed-size text windows) to be added to the query. At the same time, we can imagine that local LSI being run on a noun-phrase-only term-document matrix indexed from the passages that matches the query best. Hence, the comparison between local feedback and local LSI in their basic forms is actually very appropriate.

To make a different, yet clearer, view of the effect of feedback sizes on retrieval performance, I plotted in Figure 5.5 five different sets of results for the

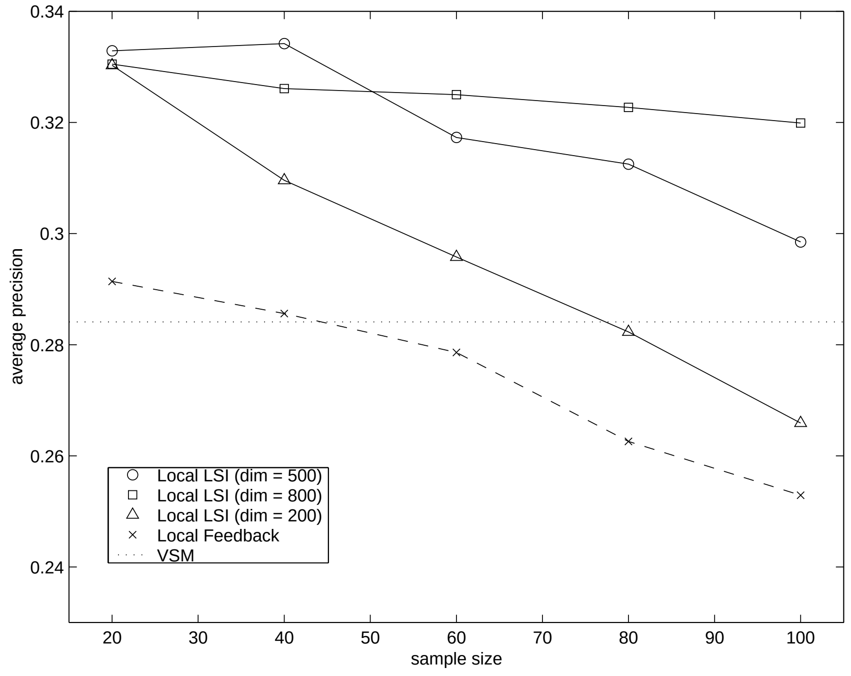

Figure 5.4: Local LSI outperforms basic local feedback and baseline VSM at different sample sizes and with different dimensions on the AP 1990 collection with TREC 4 ad-hoc topics and Okapi indexing on the retrieval documents.

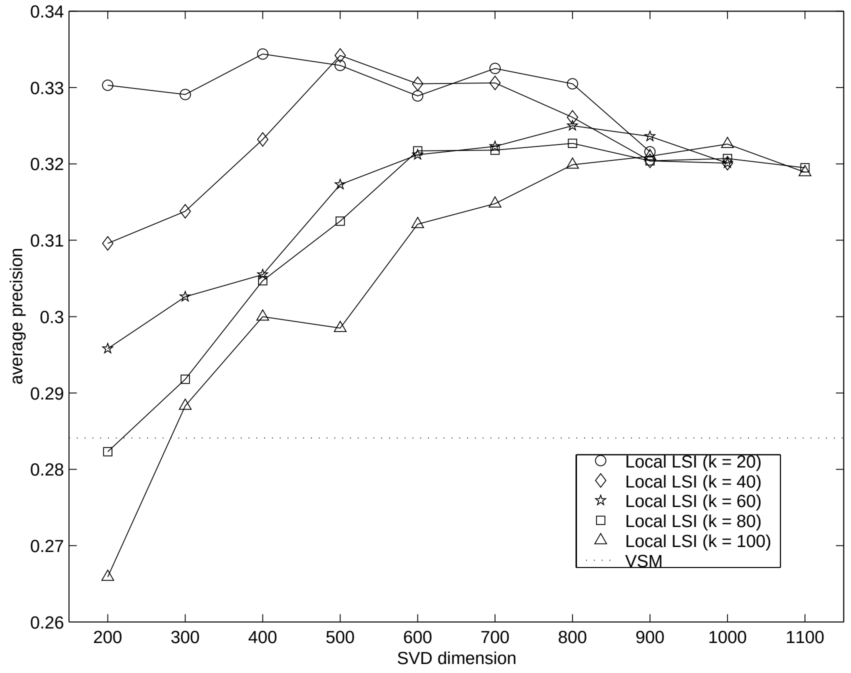

Figure 5.5: Performance of local LSI varies with SVD dimension and sample size on the AP 1990 collection with TREC 4 topics and Okapi weighting; the average precisions appear to converge at high dimensions, however.

aforementioned five feedback sizes with local LSI. The first and most important characteristic of this graph is: all but two points of local LSI plots lie above that of baseline VSM. Then, we notice the rather interesting configuration formed by the collection of five plots, which shows, among other things, how feedback size matters at low SVD dimensions but not so much at high ones. If we ignore the relative position of each plot on the graph, we see that the dimension at which I obtained performance peak for each feedback size has a positive relationship to its size. In other words, more training data means more SVD dimensions needed.

We cannot draw a conclusion from this figure and Figure 5.3 as to what feedback size to be used in the future on other collections. There has been previous research on the effect of feedback size on retrieval performance [2], but generally there are no clear guidelines (except for relying on previous experience) as to what specific size to use on a given collection. All that can be learned from the experiments is: the number feedback documents will always be collection dependent.

5.3.2 TREC French and German Collections

My next set of experiments are conducted on the TREC French and German collections as described in Section 4.3.4, as I turn my focus quickly to cross-language retrieval.

But, since there are four sets of monolingual retrieval results available from Section 4.3.4, it would have been negligent for me not to try local LSI on those cases. The original results of VSM, GVSM, LSI, and ADE are listed in Tables 4.9 and 4.10, from which I separated monolingual from cross-language results and reorganized them into Tables 5.1 and 5.2. Recall that I used the SMART *ntc* weighting scheme to index both the training and retrieval collections, and there

		French		German	
		TREC 6	TREC 7	TREC 6	TREC 7
VSM	AvgP	0.3162	0.2796	0.2958	0.2541
	P10	0.4714	0.4357	0.4043	0.3259
LSI	AvgP	0.1711	0.2070	0.1025	0.0651
	P10	0.2190	0.3036	0.1609	0.0852
ADE	AvgP	0.2064	0.2929	0.1635	0.1527
	P10	0.3095	0.4000	0.2304	0.2148
Local LSI	AvgP	**0.3608**	**0.3372**	**0.3598**	**0.2815**
	P10	**0.4810**	**0.5143**	**0.4652**	**0.3542**
	%VSM	114.1	120.6	121.6	111.9

Table 5.1: Monolingual local LSI results on the TREC French and German corpora with TREC 6 and 7 cross-language topics and SMART *ntc* indexing on the local documents.

		German ⇒ French		French ⇒ German	
		TREC 6	TREC 7	TREC 6	TREC 7
LSI	AvgP	0.1175	0.0947	0.1495	0.0751
	P10	0.1810	0.1250	0.2217	0.1000
ADE	AvgP	0.1671	0.1933	0.2177	0.1572
	P10	0.2810	0.2607	**0.2957**	0.1963
Local LSI Pre-translation	AvgP	**0.2010**	**0.2279**	**0.2294**	**0.1902**
	P10	**0.3143**	**0.2877**	0.2870	**0.2148**
Local LSI Post-translation	AvgP	0.1712	0.2101	0.2207	0.1707
	P10	0.2857	0.2857	0.2826	0.2111

Table 5.2: Cross-language local LSI results on the TREC French and German corpora with TREC 6 and 7 cross-language topics and SMART *ntc* indexing on the local documents.

were two sets of queries: TREC-6 and TREC-7.

For monolingual local LSI, I used a feedback size of 20 (from my experience with the AP 1990 collection) and built the LSI spaces on the top documents from VSM retrieval. When selecting top documents, I used the text sampling approach discussed in Section 5.2.2: I extracted the actual text of the top documents and re-indexed them with the SMART *ntc* weighting; for each term in the query I used its global weighting from the original collection, because the *idf* factor derived from a large collection is more effective in distinguishing content words from non-content words. However, since *ntc* is just a *tf* $\times$ *idf* scheme, and the only factor that depends on the collection size is the *idf* factor, what I did was essentially equivalent to the vector sampling process.

Again, for each query set I combined the feedback documents for all queries and computed a single SVD. Since the numbers of TREC-6 and TREC-7 topics are 25 and 28, respectively, the total number of feedback documents in a combined set never exceeded 560, while the number of unique terms was no more than 17,000. For such small collections, it always took less than 60 seconds to compute the full SVD's of the matrices on "dragon." Here, we see the speed of local LSI (in computing an SVD) is not a real issue as the computer being used was certainly far from the fastest available, and queries were also processed in batches. It remains to be seen whether the performance of local LSI justifies its usage.

The numbers shown in the last rows of Table 5.1 certainly give an affirmative answer, as we see the performance gains by local LSI over VSM range from 10% to 20%. The dimensions at which these best scores are obtained are small as well: With the exception of the French TREC-7 run, whose best average precision was found at 100 dimensions, all peak performances were reached at 200 dimensions. This shows that we continue to see the effect of dimension reduction even with local LSI. In addition, I made a performance comparison among local LSI runs with different feedback sizes on the TREC-6 German retrieval. As in Figure 5.5, I chose feedback sizes from 20 to 100 in increments of 20. I plotted the average precision against SVD dimensions from 100 to 1,200 by the interval of 100 for each run. For the two smaller feedback sizes of 20 and 40, since there are not enough documents to have 1,200 dimensions, I drew their plots from 100 to their largest possible dimensions. The overall pattern of the plots in the figure is similar to

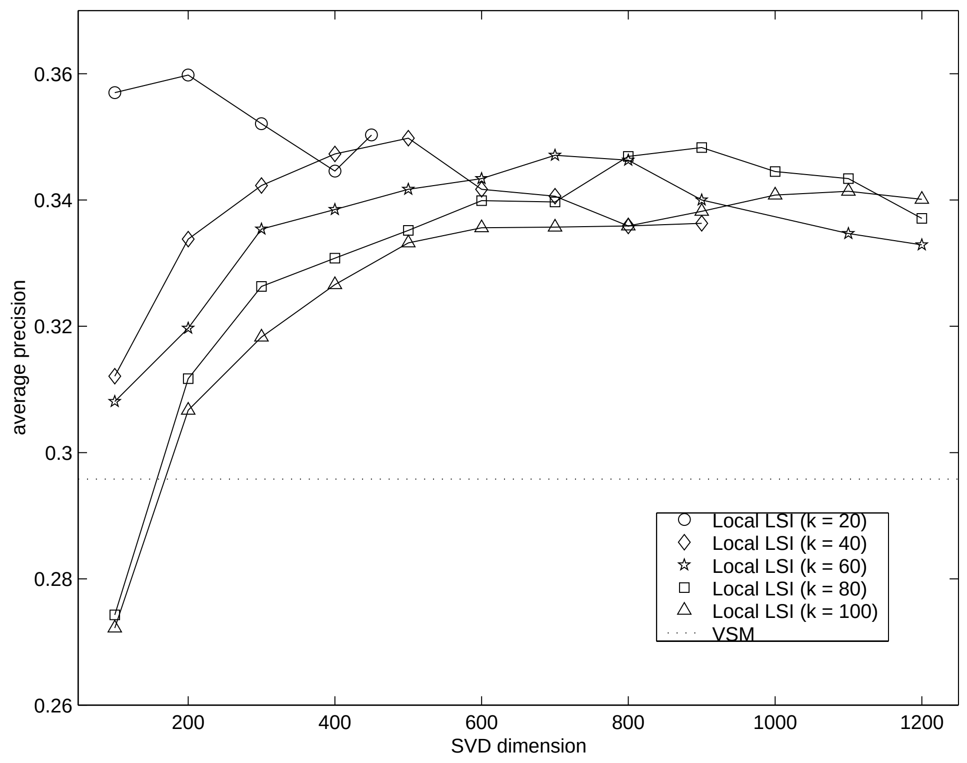

Figure 5.6: Performance of monolingual local LSI varies with feedback size and SVD dimension on the TREC German collection with TREC-6 cross-language queries.

that of Figure 5.5: they converge in higher dimensions and most of the points lie above the baseline VSM.

In the cross-language runs, I experimented with both the pre-translation and the post-translation local LSI, as discussed in Section 5.2.3. Since the combined pre-translation and post-translation method is essentially a specific implementation of the post-translation method, I did not test it here—the post-translation runs I made were sufficient to showcase its ability.

In the pre-translation method, the idea is to find the best training documents for query translation. In Section 4.3.4, I used the 40,000 comparable documents in French and German as the training corpora. Therefore, here I computed VSM similarities between the queries and those training documents in the same language. Then, I selected the texts of 100 top ranked documents for each query from the training collection, as well as their counterparts in the other training corpus. I used a larger feedback size of 100 because from my previous experience it is usually the case that more dimensions (and hence training documents) are needed to establish cross-language associations and obtain good results. The combined collection of the top documents of a set of topics has about 2,000 documents. I then computed two separate SVD's for each corresponding pair of feedback collections. For example, for the German TREC-6 topics, I computed an SVD on the top-100 documents from the German training corpus retrieved by VSM, and another SVD on the matching documents in the French training corpus. Cross-language LSI as described in Section 3.5.1 was then run for each training sub-collection, and the dimensions of the training matrices were "probed" to find the best average precision results.

For post-translation local LSI, the process is simpler: I applied local LSI on the results of cross-language ADE retrieval (since its results are the best among the methods that I ran). The feedback size was set to 20, the same as in monolingual local LSI.

The results of both local LSI approaches are shown in Table 5.2. We see that the combination of ADE and post-translation feedback is still outperformed by pre-translation local LSI consistently on these test collections. Since the training corpora I used are only comparable to each other, it is really remarkable that pre-translation local LSI can build from these term associations across the languages well enough for high-quality cross-language retrieval.

5.3.3 TREC English and French Collections

Continuing with my cross-language experiments, my next set of results come from doing retrieval on the TREC English and French collections that I used in Section 4.3.5, where the processing details of the collections were provided. The main differences between these collections and the French and German collections from the last section are in that stemming and SMART *Lnu* term-weighting were used here compared to no stemming and *ntc* indexing.

Recall that I obtained quite high baseline VSM average precision scores for the two collections, especially with TREC-7 topics. I re-organized the monolingual results from Section 4.3.5 into Table 5.3. Then, I ran local LSI with VSM retrieval in hope to better them. Once again, feedback size was set at 20 and top documents for a set of query were bundled together and processed by a single SVD. The actual computation times were all under a minute, again on "dragon."

When extracting the top documents, I used the text sampling approach. Then, I indexed the documents with the SMART *ntc* weighting scheme, which normalizes all document to unit length, instead of the *Lnu* weighting that was used on the full corpus and gives more importance to long documents. The Okapi weighting scheme, which favors short documents, was also tried but gave poor performance. When calculating the weight of a term in a local document, I applied its *idf* factor obtained from the full corpus, again in hope that this will provide better differentiation power.

The monolingual local LSI results are shown in Table 5.3, along with the SVD dimensions at which the best scores are achieved. It is somewhat surprising to see the numbers for local LSI come below those of baseline VSM on English retrieval with TREC-7 topics. But, a closer look will help us to perhaps understand the

		English		French	
		TREC 6	TREC 7	TREC 6	TREC 7
VSM	AvgP	0.3597	**0.4882**	0.3661	0.4056
	P10	0.5095	**0.6385**	0.5048	0.5250
LSI	AvgP	0.2308	0.3845	0.2402	0.2975
	P10	0.3514	0.5123	0.3619	0.4286
ADE	AvgP	0.2999	0.4216	0.2931	0.3185
	P10	0.4524	0.5815	0.4333	0.4500
Local LSI	AvgP	**0.3877**	0.4524	**0.3934**	**0.4075**
	P10	**0.5143**	0.5885	**0.5143**	**0.5264**
	Dim	300	100	100	300

Table 5.3: Monolingual local LSI results on the TREC English and French corpora with TREC 6 and 7 cross-language topics and SMART *ntc* indexing on the local documents.

		French ⇒ English		English ⇒ French	
		TREC 6	TREC 7	TREC 6	TREC 7
LSI	AvgP	0.1936	0.3027	0.1965	0.2667
	P10	0.2857	0.4546	0.2952	0.3786
ADE	AvgP	0.2656	0.3482	0.2582	0.2978
	P10	0.4048	0.4783	0.3571	0.4179
Local LSI Pre-translation	AvgP	0.2740	0.3432	0.2610	0.2947
	P10	**0.4238**	0.4738	0.3619	0.4157
	Dim	2000	2000	2000	2000
Local LSI Post-translation	AvgP	**0.2797**	**0.3756**	**0.2812**	**0.2983**
	P10	0.4005	**0.4815**	**0.3857**	**0.4181**
	Dim	300	500	450	300

Table 5.4: Cross-language local LSI results on the TREC English and French corpora with TREC 6 and 7 cross-language topics and SMART *ntc* indexing on the local documents.

reason behind it: The VSM average precision of 0.4882 is such a high number that additional "help" from local LSI only serves to sully it. After all, however, local LSI does improve upon VSM in three out of four cases, and the improvement seems to be the largest when VSM is the worst.

For cross-language retrieval, I again experimented with both pre-translation and post-translation local LSI methods. For the pre-translation methods, I used the feedback size of 100 and indexed the combined top documents for a query set with SMART *ntc* weighting function. The resulting matrices all had fewer than 2,800 documents and 16,000 terms and full SVD's took about 2 hours each to complete on "dragon." Had the computations been done separately for each query, the time it took would have been much less—recall that for the top-20 feedback of 28 queries it took less than a minute to compute the SVD. The results of the four cross-language runs along with the SVD dimensions at which these best results were achieved are shown in Table 5.4. In all four cases, the best average precisions were coincidentally reached at 2,000 dimensions. We see that the performance of pre-translation local LSI on the English-French corpora is similar to that of ADE at best. But, it has definite advantages in its low computational cost, even though it is done at instead of before retrieval time.

The post-translation local LSI was tested on top-20 feedback of cross-language ADE results. These results are listed in the last rows of Table 5.4. Although it shows improvement on all four cases, it is somewhat interesting to see that the improvement is the greatest on the retrieval of English documents by TREC-7 French queries and the retrieval of French documents by TREC-6 English queries. The explanation for the former case could be that local LSI is definitely going to help because, compared to the other three cases, the cross-language ADE result is farthest from that of the baseline monolingual VSM. The other three cases, on the other hand, are quite consistent between monolingual and cross-language retrieval: the "degree" of improvement by local LSI on VSM is similar to that by local LSI on ADE on the same collection. In summary, we see that local LSI methods, both pre-translation and post-translation, are valuable assets to have for cross-language retrieval.

5.3.4 NACSIS Test Collection

The last test collection for local LSI is the NACSIS collection of Japanese and English academic paper abstracts. The details of this NTCIR-1 collection are described in Section 4.3.6; I repeat some of them here: I used 332,918 Japanese (ntc1-j1) and 187,080 English documents, 21 training and 39 test topics all in Japanese, and two types of relevance judgment files (rel1 and rel2); I indexed the retrieval collections with the SMART *Lnu* weighting function, and converted two-byte codes in Japanese text into four-character words in ASCII before initial processing; I created parallel training documents by extracting the Japanese and English documents with the same ID's, but I only used one-fourth of them since the full English parallel corpus is almost the entire English collection.

As before, I first tried local LSI on top of VSM in monolingual retrieval, with the usual feedback size of 20 and the SMART *ntc* weighting on the local documents. The performance improvement over VSM was not all that great, as Table 5.5 indicates[2]. The results of VSM, LSI, and ADE in the same table are copied from those of Table 4.14 in Section 4.3.6. The ineffectiveness of local feedback on NTCIR-1 was also observed by Sakai et al. [91] in their experiments. They not only found little performance advantage of post-translation local feed-

[2]I also tried the ETH's similarity thesaurus weighting function but it gave even worse results.

		topic0001-0030		topic0031-0083	
		rel1	rel2	rel1	rel2
VSM	AvgP	0.3452	0.3295	0.2601	0.2870
	P10	0.5143	0.5333	**0.4103**	**0.4974**
LSI	AvgP	0.2555	0.2569	0.2413	0.2702
	P10	0.4524	0.4667	0.4000	0.4846
ADE	AvgP	0.2807	0.2795	0.2576	0.2874
	P10	0.4571	0.4762	0.3949	0.4872
Local LSI	AvgP	0.3498	0.3306	0.2627	0.2908
($k = 20$)	P10	0.5238	0.5429	0.4051	0.4846
Local LSI	AvgP	**0.3530**	**0.3315**	**0.2643**	**0.2923**
($k = 50$)	P10	**0.5429**	**0.5571**	0.4077	0.4923

Table 5.5: Monolingual local LSI results on NTCIR-1 with two sets of cross-language topics and SMART *ntc* weighting on the local documents.

back over baseline cross-language retrieval, but sometimes they also obtained a decline in average precision with local feedback.

However, knowing Japanese is quite different from languages like English or French, I ran another set of local LSI experiments with a larger feedback size. This time the top-50 documents returned by VSM are kept for building the local LSI space. The matrix of the top documents of 21 training topics had 2,896 rows (terms) and 962 columns (documents), while the matrix of the feedback documents of 39 training topics had 4,703 rows and 1,705 columns. The full-SVD computation of the larger matrix took about 10 minutes to finish on "dragon." As we see from Table 5.5, the improvement in retrieval with a larger feedback size is still at most trivial. My results, thus, seem to be consistent with those of Sakai et al. [91] in finding that local feedback methods are less useful on the NACSIS collections than on, for example, the TREC collections.

The last set of results for this chapter as well as this dissertation is from the local LSI runs on Japanese-English retrieval of NTCIR-1. With the mediocre results of monolingual retrieval by local LSI, I focused my attention on the pre-translation local LSI method instead of post-translation, since the latter is essentially a variation of the monolingual local LSI.

Because there is no English translation of the Japanese queries, I could not run monolingual English retrieval as a baseline to compare with my cross-language results. The comparison from monolingual Japanese retrieval would be quite inaccurate since the Japanese retrieval collection has 77% more documents than its English counterpart. With the aid of local LSI, however, I was able to emulate monolingual results from cross-language retrieval. Recall from Section 4.3.6 that the entire set of English parallel documents consists of most of the English retrieval collection; I took advantage of this fact and ran pre-translation local LSI on the full parallel document sets: I located the top-100 documents returned by VSM for each query on the Japanese training collection, and I identified their parallels in the English training collection; I computed separate SVD's on the two feedback collections and transformed Japanese queries into the parallel English term space with LSI. Not surprisingly, the retrieval results from this procedure are very good; the first rows of Table 5.6 shows the two runs for two queries sets. The average precisions of cross-language local LSI with the training queries are as high as those of the monolingual Japanese retrieval, while those with the test queries actually exceed those of baseline VSM. In essence, my local LSI with the full training set is a test of ability of LSI in doing mate retrieval, which is the task of identifying translational equivalents from document-aligned parallel corpora [65]—LSI does seem to excel in this area [65, 70].

But, with training collections that contain only one-fourth of the document from the full collections, I was able to test the ability of local LSI in "pulling in" other relevant documents from among those not selected for training. Table 5.6 reveals the results of pre-translation local LSI on the smaller training collection

		topic0001-0030		topic0031-0083	
		rel1	rel2	rel1	rel2
Local LSI (Full Training)	AvgP	0.3430	0.3352	0.2868	0.3169
	P10	0.4381	0.4571	0.3282	0.3872
LSI	AvgP	0.1327	0.1375	0.1162	0.1322
	P10	0.2429	0.2619	0.1462	0.1923
ADE	AvgP	0.1554	0.1606	0.1703	0.1898
	P10	0.2810	0.3048	0.2000	0.2538
Local LSI Pre-translation	AvgP	**0.2620**	**0.2581**	**0.2383**	**0.2568**
	P10	**0.3857**	**0.4143**	**0.3103**	**0.3667**
Local LSI Post-translation	AvgP	0.1597	0.1567	0.1621	0.1815
	P10	0.2762	0.3000	0.2026	0.2590

Table 5.6: Cross-language local LSI results on NTCIR-1 with two sets of cross-language topics and SMART *ntc* weighting on the local documents.

with the usual feedback size of 100. The local LSI method does show high performance gain over the global methods like LSI and ADE, whose results were copied from Table 4.14 in Section 4.3.6. Whereas in Section 4.3.6, I already made comparisons between my ADE results with a few published ones on this collection, the results here by local LSI would compare favorably against them. Still, I did not perform word segmentation on Japanese texts or use a bilingual dictionary. In the same table, I also presented the results for running post-translation local LSI on top of cross-language ADE. As we can clearly see, the extra feedback step after the initial retrieval does not help improve the performance, just like in the monolingual case.

5.4 Conclusion

Local LSI has several significant merits. By focusing the analysis on the selected documents related to the query, it outperforms VSM on monolingual retrieval when ADE is not able to do so. The additional computational cost at retrieval time does not seem to be a significant factor, as the SVD of a small number of documents takes only seconds to compute on an average machine. Since term relevance is automatically deduced through SVD in local LSI, this saves the need for statistically selecting terms from top feedback documents and re-weighting them in the expanded query, as in the case of local feedback.

In cross-language retrieval, the pre-translation local LSI approach learns more accurate translational term-associations by concentrating on the training documents that actually contain the words to be translated, while the post-translation approach is a mere extension of the monolingual local LSI. I recommend pre-translation local LSI for cross-language applications because it can be used as long as there is a parallel or comparable corpus available, and it will be most effective when the training corpus covers the same subject domain as the retrieval collection. The post-translation approach, on the other hand, would depend on the cross-language method it employs for its performance. One possible method here is the pre-translation method, though this has not been fully tested, since the pre-translation method alone is getting high-quality retrieval results.

Chapter 6

Conclusion

In this dissertation I have studied, created, experimented, and compared methods that conduct information retrieval by modeling terms and documents in a high dimensional vector space. In a detailed analysis of those methods, I show that the term relationships in documents is intrinsically determined by the structure of the vector space being used, and that we can apply some simple matrix computation techniques to alter that structure. The relationship between terms can be used to supply the original user query with additional terms that will help the retrieval of relevant documents. Moreover, the derived term associations between terms in two different languages make the document retrieval across those languages not only possible but also effective.

6.1 Results

There are three main varieties of the vector-based information retrieval methods: VSM, GVSM, and LSI. GVSM and LSI are extensions to basic VSM in that they analyze a set of training documents to derive term-term associations to facilitate information retrieval. I have shown how the three methods can be represented in a unified mathematical framework called dimension equalization, which is accomplished through the singular value decomposition of the term-document matrix of a corpus being considered. Under this unification, we have learned how equalization of reduced SVD dimensions can produce better term-term associations for improving performance in information retrieval.

The only method that uses both reduction and equalization of SVD dimensions of a term-document matrix is LSI. LSI has demonstrated its advantages over the other two methods on some of the small, classic information retrieval test collections. However, the computational requirement of SVD presents a major problem for LSI in its scalability. My results show that when the retrieval collection becomes very large, LSI has failed to compete with the traditional VSM method because of the tiny number of SVD dimensions that can be computed with current limited computing resources.

Upon a close examination of the matrix SVD's of many different types of corpora, I observed the so-called low-rank-plus-shift structure that exists in all those term-document matrices. Such structure, as I discovered, can be utilized to create an approximation of a large number of equalized dimensions with only a few initial ones being actually computed through the SVD software. The approximation process is fairly easy to compute: it only involves the basic matrix additions and multiplications of the different SVD parts of the original matrix. I have named this method approximate dimension equalization (ADE) and have presented its retrieval results on various test collections and compared it to VSM, GVSM, and LSI. My findings are conclusive: ADE does improve on LSI with the same number of computed dimensions; it bridges the gap between the performances of LSI and VSM in monolingual retrieval; and it is very effective in cross-language retrieval where VSM is not applicable. These findings further confirm my hypothesis that equalization of reduced dimensions of a term-document matrix produces better associations among related terms.

The computation of LSI can be further scaled down with the application of document sampling. I have made a brief study of the global sampling approaches to LSI approximation. I have applied the theoretical findings by Frieze et al. [43]

to the implementation of a weighted random sampling algorithm. My experiments show that the weighted approach is very promising in sampling accuracy but much less so in retrieval effectiveness.

My focus of the sampling study has been on the use of LSI on the top returned documents from an initial retrieval. Local feedback methods have been very popular in the IR research community, but are traditionally accomplished by either blind addition of top documents to the original query, or by careful selection of most related terms from the top documents and supplying them to the query. Local LSI is a similar method to local feedback: it automatically derives the appropriate term relationship structure through the computation of the SVD of the matrix of the top returned documents. The rationale behind local LSI is that the documents in the local region of the query give LSI an opportunity to make a focused analysis of the term associations that would really help the retrieval of documents relevant to the query. This presents a solution to the LSI scalability problem because selecting the documents local to the query can be viewed as selecting the SVD dimensions that have the most significant effect on performance.

In this research, I have extended the study by Hull [61, 60] and have applied the local LSI method to information retrieval, especially in cross-language applications. My experimental results show that in most monolingual cases, local LSI improves upon the baseline VSM retrieval by a significant amount. For cross-language retrieval, I have proposed two different approaches to use local LSI. The post-translation local LSI method is similar to the monolingual local LSI method—it runs LSI on the initial cross-language retrieval results. The novel approach is pre-translation local LSI. This method runs an initial monolingual retrieval on the training documents in the language of the query, and then performs cross-language LSI on the parallel top documents returned in the initial run. The advantage of the pre-translation method is that we maximize the accuracy of translation before the retrieval, while the advantage of the post-translation method is we can insert any other algorithm in for the initial cross-language retrieval, including pre-translation local LSI. My experimental results indicate that pre-translation local LSI is very effective in cross-language retrieval and post-translation local LSI is in most cases an asset to have to enhance the retrieval performance.

6.2 Future Directions

An alternative approach to document sampling is term selection, where the rows instead of columns of a term-document matrix are chosen according to a certain criterion. Of course, the criteria would be different in the two approaches, since in term selection we want to pick those terms that are well represented by the corpus. One criterion is to select terms that do not have very high or very low frequency counts [74, 75], and this is similar to the global sampling approaches for LSI. Another can be the selection of terms that are closely related those in the query [76], which is a brand new approach to implementing local LSI feedback. I believe the selection of terms in the local region of a query to build LSI space deserves more investigation.

From the numerous experiments I conducted for this dissertation, I have learned that no matter how well a method or weighting schemes does in average precision, there is always some queries on which another method or weighting scheme outperforms it. Thus, I wonder whether it is possible to be able to choose the particular weighting scheme and retrieval method for each query. The underlying problem here is to detect the characteristics of a query that may decide its applicability with certain weighting scheme and retrieval method. For example, the length of the query or the percentage of obscure terms in the query, may be two of these factors. Studies are needed to determined if my observations are only arbitrary.

In conclusion, we have learned that the fundamental idea in LSI—of using equalized dimensions of the term-document matrix for projection—is a very sound approach to building the term-association space for query expansion. Since different semantics and meanings are “stored in” or “captured by” the different dimensions of the this space, the dimension truncation of LSI at the limit of current computing capability would only hurt the retrieval performance. This thesis have investigated two approaches that manage to retain as many useful dimensions of the semantic space as possible: ADE does this by including all the “latent” dimensions and local LSI by keeping most of the dimensions that are “relevant” to the current query while discarding the rest. A thorough understanding of the role that each dimension plays would definitely be a major breakthrough in not only vector-based information retrieval, but information retrieval in general.

Appendix A

Sample Queries and Documents

This section describes and lists some of the queries and documents from the test collections used in this thesis.

A.1 Comparable Documents: TREC French and German Collections

These are news articles from the Swiss new agency (Schweizerische Depeschen Agentur) from 1988 to 1990, and are used in the the TREC Cross-Language Information Retrieval (CLIR) track [101]. In the experiments by Rehder et al. [84], 40,000 document pairs were found through an automatic alignment procedure. These documents are comparable, not parallel, since they are not strictly translations of each other.

Here is an sample pair of documents. The first is in French and the second in German:

> Pérou: une filiale de Nestlé accusée de manoeuvres spéculatives
>
> Lima, 4 jan (ats) Le gouvernement péruvien a accusé dimanche la société "Perulac", filiale du groupe Nestlé ayant son siège à Vevey (VD), de s'être livrée à des manoeuvres spéculatives en rationnant la distribution de sa production de lait en poudre. Alors que Perulac rejette catégoriquement cette accusation, le président péruvien Alan Garcia a demandé l'application de sanctions contre "les coupables de l'accaparement de ce produit".
>
> "Nous allons prendre des mesures et sanctionner la société Perulac, afin que ne se répète pas cette manoeuvre condamnable", a-t-il déclaré, ajoutant

que le gouvernement péruvien désormais participera directement à la production et à la fabrication du produit, pour éviter de nouvelles actions spéculatives. Le gouvernement péruvien a lancé ses accusations trois jours aprés avoir effectué une opération coup de poing dans les entrepôts de Perulac au nord de la capitale Lima. En présence du ministre de l'agriculture Remiglio Morales Bermudez, la police péruvienne a en effet saisi dans la journée du 31 décembre plusieurs millions de botes de lait en poudre "Ideal".

Contactée à Lima, la direction de Perulac a estimé que le nombre de bôites saisies était supérieur aux 2,8 millions préalablement annoncé. Une bote de lait en poudre "Ideal" vaut actuellement sur le marché péruvien entre 25 et 30 centimes suisses. L'opération de la police péruvienne dans les entrepôts de Perulac à Santa Anita et à Chiclayo faisait suite à de nombreuses dénonciations selon lesquelles la filiale de Nestlé n'approvisionnait plus le marché. Le lait en poudre était en effet devenu pratiquement introuvable dans la capitale.

Perulac-"Affäre": Garcia droht mit Sanktionen

Lima, 5. Jan. (sda) Die peruanische Regierung hat am Sonntag die Nestlé-Tochtergesellschaft Perulac beschuldigt, mit der Verknappung des Milchpulverabsatzes "Spekulationsmanöver" unternommen zu haben. Staatschef Alan Garcia kündigte Sanktionen gegen die "Schuldigen an der Hortung dieses Produkts" an, "damit ein solches verdammenswertes Manöver nicht nochmals passiert". Die peruanische Regierung werde sich darum künftig direkt an der Herstellung von Milchpulver beteiligen, um neuen Spekulationsmachenschaften vorzubeugen. Die Perulac hat die Anschuldigungen kategorisch dementiert.

Die Regierung brachte ihre Anschuldigungen drei Tage nach der Blitzaktion gegen die Lagerhallen der Perulac im Norden der Hauptstadt Lima vor. Im Beisein des Agrarministers Remiglio Morales Bermudez hatte die Polizei am Silvestertag mehrere Millionen Dosen Milchpulver der Handelsmarke "Ideal" beschlagnahmt. Auf Anfrage schätzte die Perulac-Direktion in Lima die Zahl der beschlagnahmten Dosen, die auf dem heimischen Markt zu umgerechnet 25 bis 30 Rappen abgesetzt werden, auf mehr als die zunchst bekanntgegebenen 2,8 Millionen ein.

Zu den Razzien der Polizei in den Perulac-Depots in Santa Anita und Chiclayo war es gekommen, nachdem zahlreiche Klagen eingegangen waren, die Nestlé-Tochtergesellschaft beliefere den Markt nicht mehr. Milchpulver war in der Tat in allerjngster Zeit in Lima kaum mehr aufzutreiben.

To see their closeness in meaning, the first sentence of the French article can be translated into English as

> Lima, Jan 4 (ats). The Peruvian government accused on Sunday the company "Perulac", a subsidiary of the Nestlé group located at Vevey (VD), of doing speculative maneuvers by rationing the distribution of its powdered milk production.

while the first German sentences can be translated as

> Lima, Jan 5 (sda). The Peruvian government accused on Sunday Nestlé's subsidiary company Perulac of speculating with the shortage of the powdered milk sales.

It should be noted that not all document pairs in the two collections are this close, though.

A.2 Parallel Documents: CMU UNICEF Collection

This collection is extracted from the LDC's United Nations Parallel Text Corpus; the training, retrieval collections and the queries are created and described by Carbonell et al. [26]. It contains 1,134 training documents and 1,121 test documents, each in both English and Spanish. Yang *et al.* also prepared a set of

30 English queries along with exhaustive relevance judgments for these queries over all 1,121 test documents. The relevance judgments were made between the English queries and the English test documents—relevance for the Spanish test documents were assumed to be the same, since these are translationally equivalent to the English.

Here is a document from the English training corpus, which consists of two paragraphs extracted from an original UN document:

> 5. Although the order of priority varies according to the country, the principal causes of infant and child deaths in the region include: diarrhoeal diseases; the expanded programme of immunization (EPI) target diseases, particularly measles; neonatal tetanus and whooping cough; acute respiratory infections (ARI); malaria; and low birth weight (often associated with maternal malnutrition). In addition, this year as in the past, some countries faced epidemics: cholera (Guinea-Bissau, Mali, Mauritania, Nigeria and Senegal); yellow fever (Cameroon, Guinea, Mali and Nigeria); cerebral-spinal meningitis (Benin, Central African Republic and Nigeria); and paralytic poliomyelitis (Senegal).
>
> 6. A considerable effort has been made to improve the collection and analysis of data concerning immunization coverage at local, regional and/or national levels. However, few countries have a reliable epidemiological monitoring system to measure programme impact on morbidity and mortality, particularly for measles, neonatal tetanus and poliomyelitis. Maternal morbidity and mortality, due principally to abortion and obstetrical complications, remain a serious cause for concern, although precise data are still scanty in most countries. In general, 80 per cent or more of all deliveries still take place at home, often without qualified assistance.

The matching document in the Spanish collection is:

> 5. Si bien el orden de prioridades varía según el pas, las principales causas de mortalidad en los primeros años de vida en la región son: las enfermedades

diarreicas; las enfermedades que combate el programa ampliado de inmunización, particularmente el sarampión; el tétanos neonatal y la tos ferina; las infecciones agudas de las vías respiratorias; el paludismo; el peso bajo al nacer (a menudo vinculado a la malnutrición materna). Además, este año, como en años anteriores, algunos países sufrieron epidemias: cólera (Guinea-Bissau, Malí, Mauritania, Nigeria y Senegal); fiebre amarilla (Camerún, Guinea, Malí y Nigeria); meningitis cerebroespinal (Benin, Nigeria y República Centroafricana); y poliomielitis (Senegal).

6. Se realizó un esfuerzo considerable para mejorar la reunión y el análisis de datos relativos a la cobertura de la inmunización a nivel local, regional y nacional. Sin embargo, pocos países tienen un sistema confiable de vigilancia epidemiológica para medir los efectos del programa en la morbilidad y mortalidad, particularmente en el caso del sarampión, el tétanos neonatal y la poliomielitis. La morbilidad y mortalidad maternas, derivadas principalmente de los abortos y las complicaciones obstétricas, sigue siendo motivo de grave preocupación, aunque en la mayoría de los países sólo se dispone de pocos datos exactos. En general, el 80% o más de los partos tienen lugar en el hogar, a menudo sin asistencia profesional.

The last Spanish sentence, for example, can be translated as

> In general, 80% or more of all child births take place at home, often without professional assistance.

which is nearly the same as the last sentence of the English document.

Two of the 30 test queries for this collection are shown below:

water purification sanitation water supply project clean water personal hygiene health sanitation

disease sickness diarrhea diarrhoeal contaminated sanitation coli bacteria sceptic excrement drinking water cholera guinea worm parasite ameba amoeba

A.3 TREC-6 Associated Press Collection

This document collection consists 242,918 Associated Press (AP) newswire articles from 1988–1990. They are used in the TREC conferences for both ad hoc track and cross-language track, and they can be found on TREC disk 1–3 [55]. The following is a typical news article found is this collection:

> Brazilian Airline Workers on Strike
> With AM-Carnival, Bjt
>
> The flight crews of Brazil's two largest airlines went on strike for higher wages Friday, tying up most domestic and some international flights on the eve of Carnival celebrations.
>
> Non-Brazilian carriers were not affected by the strike at Varig and Vasp airlines. Transbrasil, Brazil's other major airline, operated normally after reaching a separate agreement with its flight crews.
>
> The Sao Paulo and Rio de Janeiro international airports said the strike stopped all departures and arrivals of Varig and Vasp. Shuttle service between Rio and Sao Paulo also was suspended.
>
> An estimated 30,000 to 40,000 passengers, including many Carnival tourists, will be affected by the planned three-day strike, the airlines' press offices said.
>
> The huge Carnival celebration kicks off Saturday and runs until Ash Wednesday, Feb. 17.
>
> Edson Antonio Matosinho, one of the directors of Brazil's 12,000-strong flight crew union, said in a telephone interview that about 7,000 Varig and Vasp employees joined the strike.
>
> The strike was called to demand a 65 percent wage hike, Matosinho said.
>
> Flight crew workers' salaries range from $385 to $3,300 a month, he said.

TREC-4 ad-hoc topics (Topic Number 201-250) are one of the sets of topics that have relevance judgments made for this collection. These topics are natural language queries that consist of one sentence each, an example (Topic Number 248) of which is shown here:

Number: 248

Description:

What are some developments in electronic technology being applied to and resulting in advances for the blind?

TREC-6 and TREC-7 cross-language topics (Topic Number 1–53) are two other sets of queries with relevance judgments for the AP collection. The topics are more elaborate in that they contain three fields: title, description, and narrative. The title field is usually a noun phrase while the other two are composed of one or more sentences. Typically people use the title and description to form the "short" version of the query and combine all three fields to create the "long" version. The example below is Topic Number 28.

Number: 29

The Ustica Affair

Description:

Did the armed forces and the secret services try to hide the truth about the disaster of Ustica?

Narrative:

On 27 June 1980 an ITAVIA DC9 crashed in the sea off the coast of Ustica. The reasons for this tragedy are still not clear. There have been many enquiries, hearings and commissions to investigate the true causes of the disaster. Strong doubts have emerged concerning the involvement of Italian and non-Italian secret services and officers of the armed forces in this event. Relevant documents are those that discuss or deny the possibility that the military or the secret services have been involved in a plot to throw the inquiry off the track.

The TREC-6 cross-language topics are also available in Dutch, French, German, and Spanish, while the translations of the TREC-7 cross-language English topics can be found in French, German, and Italian.

Appendix B

Mathematical Derivations

B.1 Term Correlation

For two terms τ_x and τ_y, define

$$O_x = \begin{cases} 1 & \text{if } \tau_x \text{ occurs} \\ 0 & \text{otherwise} \end{cases}, \quad O_y = \begin{cases} 1 & \text{if } \tau_y \text{ occurs} \\ 0 & \text{otherwise} \end{cases}$$

and note that

$$O_x\, O_y = \begin{cases} 1 & \text{if } O_x = 1 \text{ and } O_y = 1 \\ 0 & \text{otherwise} \end{cases}$$

is an indicator random variable for whether both τ_x and τ_y occur in a document. The correlation of O_x and O_y is defined as

$$\mathrm{Corr}(O_x,\, O_y) = \frac{\mathrm{Cov}(O_x,\, O_y)}{\sqrt{\mathrm{Var}(O_x)\,\mathrm{Var}(O_y)}} \tag{B.1}$$

where $\mathrm{Cov}(O_x,\, O_y)$ is the covariance of O_x and O_y, and Var(.) denotes the variance of a random variable.

The variance of an indicator random variable is the product of the probabilities of the occurrence and non-occurrence of its corresponding event [90]. Hence,

$$\begin{aligned} \mathrm{Var}(O_x) &= \Pr(\tau_x) \cdot [1 - \Pr(\tau_x)] &\approx \frac{df(\tau_x)}{n} \cdot \left[1 - \frac{df(\tau_x)}{n}\right] \\ \mathrm{Var}(O_y) &= \Pr(\tau_y) \cdot [1 - \Pr(\tau_y)] &\approx \frac{df(\tau_y)}{n} \cdot \left[1 - \frac{df(\tau_y)}{n}\right], \end{aligned}$$

where the probability of the occurrence of a term is estimated by the proportion of its document frequency (denoted by $df()$) to the collection size (denoted by n as before).

The covariance of two random variables is related to their expectations, while the expectation of an indicator random variable for an event is just the probability that the event occurs. So, we can compute the covariance of O_x and O_y as follows:

$$\begin{aligned}\mathrm{Cov}(O_x,\, O_y) &= \mathrm{E}[O_x,\, O_y] - \mathrm{E}[O_x]\cdot\mathrm{E}[O_y] \\ &= \Pr(\tau_x,\, \tau_y) - \Pr(\tau_x)\cdot\Pr(\tau_y) \\ &\approx \frac{df(\tau_x \,\&\, \tau_y)}{n} - \frac{df(\tau_x)}{n}\cdot\frac{df(\tau_y)}{n},\end{aligned}$$

where we use $df(\tau_x \,\&\, \tau_y)$ to represent τ_x and τ_y's co-occurrence score, or the number of times they both appear in a document.

Putting them all together, Equation B.1 becomes

$$\mathrm{Corr}(O_x,\, O_y) \approx \frac{n\, df(\tau_x \,\&\, \tau_y) - df(\tau_x)\, df(\tau_y)}{\sqrt{df(\tau_x)\,(n - df(\tau_y))\, df(\tau_y)\,(n - df(\tau_y))}}.$$

B.2 Orthogonal Procrustes Problem

Suppose we have two matrices A and B with $A, B \in \Re^{m\times p}$. In the *orthogonal Procrustes problem* we look for an orthogonal matrix Q that can be applied to rotate B into the subspace of A. Specifically, we try to solve the following problem:

$$\min_{Q^TQ=I} \|A - BQ\|_F.$$

Recall that since $\|A\|_F^2 = \operatorname{trace}(A^T A)$, where $\operatorname{trace}(X)$ is defined as the sum of the diagonal entries of square matrix X, we have

$$\begin{aligned}
\|A - BQ\|_F^2 &= \operatorname{trace}((A - BQ)^T (A - BQ)) \\
&= \operatorname{trace}((A^T - Q^T B^T)(A - BQ)) \\
&= \operatorname{trace}(A^T A - Q^T B^T A - A^T BQ + Q^T B^T BQ).
\end{aligned}$$

Let $X \in \Re^{m\times m}$ and $Y \in \Re^{m\times m}$ be given. The trace function has the following properties:

1. $\operatorname{trace}(X^T) = \operatorname{trace}(X)$.
2. $\operatorname{trace}(X + Y) = \operatorname{trace}(X) + \operatorname{trace}(Y)$.
3. $\operatorname{trace}(XY) = \operatorname{trace}(YX)$.
4. $\operatorname{trace}(YXY^{-1}) = \operatorname{trace}(X)$, where Y is invertible.

We can thus derive that

$$\|A - BQ\|_F^2 = \operatorname{trace}(A^T A) + \operatorname{trace}(B^T B) - 2\operatorname{trace}(Q^T B^T A).$$

Hence, our Procrustes problem is equivalent to looking for an orthogonal matrix Q that maximizes $\operatorname{trace}(Q^T B^T A)$. Now, if we let $B^T A = U\Sigma V^T$ be the singular value decomposition of $B^T A$, then

$$\operatorname{trace}(Q^T B^T A) = \operatorname{trace}(Q^T U\Sigma V^T) = \operatorname{trace}(V^T Q^T U\Sigma) \leq \operatorname{trace}(\Sigma),$$

where the last inequality holds because $V^T Q^T U$ is an orthogonal matrix. In order to maximize $\operatorname{trace}(Q^T B^T A)$, we set $V^T Q^T U = I$ or $Q = UV^T$.

B.3 Monolingual LSI and Procrustes Analysis

In the monolingual case, let $A = U\,\Sigma\,V^T$ as usual. Following the rationale of the Procrustes analysis method from Section 3.4.4, we are to find the orthogonal rotation matrix between $\Sigma_k V^T$ and $\Sigma_k V^T$. Since

$$(\Sigma_k\,V^T)\,(\Sigma_k\,V^T)^T = \Sigma_k\,V^T\,V\,\Sigma_k = \Sigma_k^2 = I\,\Sigma_k^2\,I^T,$$

then $II^T = I$ (the identity matrix) is the orthogonal rotation matrix we are looking for. Therefore, substituting U' and V' in Equation 3.28 by I, the similarity comparison formula for Procrustes analysis on monolingual retrieval is

$$\begin{aligned}\mathrm{Sim}_{PA-ML}(d,\,q) &= (I^T I_k\,U^T\vec{d}\,)\cdot(I^T I_k\,U^T\vec{q}\,)\\ &= (I_k\,U^T\vec{d}\,)\cdot(I_k\,U^T\vec{q}\,)\\ &= \mathrm{Sim}_{LSI-ML}(d,\,q),\end{aligned}$$

exactly the LSI formula in Equation 3.19.

B.4 Cross-Language LSI and Procrustes Analysis

We have

$$\begin{aligned}A &= U\,\Sigma\,V^T\\ B &= W\,\Omega\,X^T\end{aligned}$$

as the matrices for the parallel training documents. As mentioned in Section 3.4.4, in Procrustes analysis, we may choose to derive a rotation matrix between $I_k V^T$ and $I_k X^T$, the equalized version of $\Sigma_k V^T$ and $\Omega_k X^T$.

At full dimension, and if there are more terms than documents in the collection, X^T and V^T are unitary. Then, the SVD of $X^T V$ is simply $X^T I V = X^T V$, which means the orthogonal rotation matrix T we are looking for is just $X^T V$ itself. Now, this variation of the Procrustes analysis approach becomes:

$$\begin{aligned}
\mathrm{Sim}_{PA-CL}(d,\, q) &= (I_k U^T \vec{d}\,) \cdot (T^T I_k W^T \vec{q}\,) \\
&= (U^T \vec{d}\,) \cdot (T^T W^T \vec{q}\,) \qquad\qquad \text{(since } k = r\text{)} \\
&= \vec{d}^T\, U\, V^T X\, W^T\, \vec{q} \\
&= \vec{d}^T\, \bar{A}\, \bar{B}^T\, \vec{q} \\
&= (\bar{A}^T \vec{d}\,) \cdot (\bar{B}^T \vec{q}\,) \\
&= \mathrm{Sim}_{LSI-CL}(d,\, q)
\end{aligned}$$

This is exactly the new formulation of the cross-language LSI similarity with full dimensionality (see Section 3.5.1).

B.5 Proof of Lemma 3.1

We have $A = U\Sigma V^T$, $P = U\Phi V^T$ and $Q = U\Psi V^T$. Then,

$$\begin{aligned}
&||\mathrm{Sim}_P(A, A) - \mathrm{Sim}_Q(A, A)||_F \\
&= ||A^T P P^T A - A^T Q Q^T A||_F \\
&= ||V\Sigma U^T U\Phi V^T V\Phi U^T U\Sigma V^T - A^T Q Q^T A||_F \\
&= ||V\Sigma^2\Phi^2 V^T - V\Sigma^2\Psi^2 V^T||_F \\
&= ||V\Sigma^2(\Phi^2 - \Psi^2)V^T||_F \\
&= \mathrm{trace}((V\Sigma^2(\Phi^2 - \Psi^2)V^T)^T(V\Sigma^2(\Phi^2 - \Psi^2)V^T)) \\
&= \mathrm{trace}(V(\Sigma^2(\Phi^2 - \Psi^2))^2 V^T) \\
&= \mathrm{trace}((\Sigma^2(\Phi^2 - \Psi^2))^2 V^T V) \\
&= ||\Sigma^2(\Phi^2 - \Psi^2)||_F.
\end{aligned}$$

B.6 Frobenius Norm of Projection Error

Given an $m \times k$ matrix U whose columns are orthonormal (*i.e.* $U^T U = I$) and an $m \times n$ matrix A, we want to find a simplified formula for calculating $\|A - UU^T A\|_F^2$.

Recall that $\|A\|_F^2 = \mathrm{trace}(A^T A)$, where $\mathrm{trace}(X)$ is defined as the sum of the diagonal entries of the square matrix X. With the following property of the trace function:

$$\mathrm{trace}(X + Y) = \mathrm{trace}(X) + \mathrm{trace}(Y),$$

we can derive that

$$\begin{aligned}
\|A - UU^TA\|_F^2 &= \text{trace}((A^T - A^TUU^T)\,(A - UU^TA)) \\
&= \text{trace}(A^TA - 2A^TUU^TA + A^TUU^TUU^TA) \\
&= \text{trace}(A^TA - 2A^TUU^TA + A^TUU^TA) \\
&= \text{trace}(A^TA) - \text{trace}(A^TUU^TA) \\
&= \|A\|_F^2 - \|U^TA\|_F^2.
\end{aligned}$$

Bibliography

[1] IJsbrand Jan Aalbersberg. Incremental relevance feedback. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 11–22. Association for Computing Machinery, 1992.

[2] James Allan. Relevance feedback with too much data. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '95)*, pages 337–343. Association for Computing Machinery, 1995.

[3] R. Attar and A. S. Fraenkel. Local feedback in full-text retrieval systems. *Journal of the Association for Computing Machinery*, 24(3):397–417, 1977.

[4] Lisa Ballesteros and Bruce W. Croft. Dictionary methods for cross-lingual information retrieval. In *Proceedings of the Seventh International Conference on Database and Expert Systems Applications (DEXA '96)*, pages 791–801. Springer-Verlag, 1996.

[5] Lisa Ballesteros and Bruce W. Croft. Phrasal translation and query expansion techniques for cross-language information retrieval. In *Proceedings of the 20th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 84–91. Association for Computing Machinery, 1997.

[6] Lisa Ballesteros and W. Bruce Croft. Resolving ambiguity for cross-language retrieval. In *Proceedings of the 21st ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 64–71. Association for Computing Machinery, August 1998.

[7] M. W. Berry and P. G. Young. Using Latent Semantic Indexing for multi-language information retrieval. *Computers and the Humanities*, 29(6):413–429, 1995.

[8] Michael W. Berry. Large scale singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49, 1992.

[9] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595,

1995.

[10] Guo-Wei Bian and Hsin-Hsi Chen. Intergrating query translation and document translation in a cross-language information retrieval system. In David Farwell, Laurie Gerber, and Eduard Hovy, editors, *Machine Translation and the Information Soup, Third Conference of the Association for Machine Translation in the Americas AMTA'98*, pages 250–265, Langhorne, PA, 1998.

[11] R. Bleir. Treating hierarchical data structures in the SDC time-shared data management system (TDMS). In *Proceedings, 22nd ACM National Conference*, pages 41–49. Association for Computing Machinery, 1967.

[12] Abraham Bookstein and Don R. Swanson. A decision theoretic foundation for indexing. *Journal of the American Society for Information Science*, 26(1):45–50, January-February 1975.

[13] Ronald N. Bracewell. *The Fourier Transform and Its Applications.* McGraw-Hill, New York, NY, 3rd edition, 1999.

[14] Martin Braschler, Min-Yeb Kan, and Peter Schäuble. The SPIDER retrieval system and the TREC-8 cross-language track. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 287–295. Department of Commerce, National Institute of Standards and Technology, 2000.

[15] Martin Braschler, Jürgen Krause, Carol Peters, and Peter Schäuble. Cross-language information retrieval (CLIR) track overview. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 25–32. Department of Commerce, National Institute of Standards and Technology, 1999.

[16] Martin Braschler, Bojidar Mateev, Elke Mittendorf, Peter Schäuble, and Martin Wechsler. SPIDER retrieval system at TREC7. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 509–517. Department of Commerce, National Institute of Standards and Technology, 1999.

[17] John Broglio, James P. Callan, W. Bruce Croft, and Daniel W. Nachbar. Document retrieval and routing using the INQUERY system. In Donna K. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 29–38. Department of Commerce, National Institute of Standards and Technology, 1994.

[18] Ralf D. Brown. Automatically-extracted thesauri for cross-language IR: When better is worse. In *Proceedings of the First Workshop on Computational Terminology (COMPUTERM'98), Montreal, Canada*, pages 15–21, 1998.

[19] Chris Buckley, James Allan, and Gerard Salton. Automatic routing and ad-hoc retrieval using SMART: TREC 2. In Donna K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 45–55. Department of Commerce, National Institute of Standards and Technology, 1993.

[20] Chris Buckley, Mandar Mitra, Janet Waltz, and Claire Cardie. Using clustering and superconcepts within SMART: TREC 6. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, pages 107–121. Department of Commerce, National Institute of Standards and Technology, 1998.

[21] Chris Buckley and Gerard Salton. Optimization of relevance feedback weights. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 351–357. Association for Computing Machinery, 1995.

[22] Chris Buckley, Gererd Salton, and James Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 292–300. Association for Computing Machinery, 1994.

[23] Chris Buckley, Gererd Salton, James Allan, and Amit Singhal. Automatic query expansion using SMART: TREC 3. In Donna K. Harman, editor, *Proceedings of Third Text Retrieval Conference (TREC-3)*, pages 69–80. Department of Commerce, National Institute of Standards and Technology, 1995.

[24] Chris Buckley, Amit Singhal, and Mandar Mitra. New retrieval approaches using SMART: TREC 4. In Donna K. Harman, editor, *Proceedings of Fourth Text Retrieval Conference (TREC-4)*, pages 25–43. Department of Commerce, National Institute of Standards and Technology, 1996.

[25] James P. Callan, W. Bruce Croft, and Stephen M. Harding. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, pages 78–83. Springer-Verlag, 1992.

[26] Jaime G. Carbonell, Yiming Yang, Robert E. Frederking, Ralf D. Brown, Yibing Geng, and Danny Lee. Translingual information retrieval: A comparative evaluation. In *Proceedings of Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 708–715. Morgan Kaufmann, 1997.

[27] Aitao Chen, Fredric C. Gey, Kazuaki Kishida, Hailing Jiang, and Qun Liang. Comparing multiple methods for Japanese and Japanese-English text retrieval. In *Proceedings of the First NTCIR Workshop on Research in Japanese Text Retrieval and Term Recognition*, pages 49–58. National Center for Science Information Systems, Tokyo, Japan, 1999.

[28] Hao Cheng. LSI or term-matching: Comparing techniques for computing lexical similarity. Master's thesis, Department of Computer Science, Duke University, Durham, NC, 1999.

[29] W. B. Croft and D. J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35(4):285–295, December 1979.

[30] Mark Davis and Ted Dunning. A TREC evaluation of query translation methods for multi-lingual text retrieval. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 483–498. National Institute of Standards and Technology, Special Publication 500-238, 1995.

[31] Mark W. Davis and William C. Ogden. QUILT: Implementing a large-scale cross-language text retrieval system. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 92–98. Association for Computing Machinery, 1997.

[32] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[33] Darrin Dimmick, G. O'Brien, Paul Over, and Willie Rogers. Guide to Z39.50/PRISE 2.0: Its installation, use, & modification. At URL `http://www.itl.nist.gov/iaui/894.02/works/papers/zp2/zp2.html`, 1998.

[34] P. Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering in large graphs and matrices. In *Proceedings of the 10th Annual*

Symposium on Discrete Algorithms, pages 291–299. SIAM, 1999.

[35] Susan T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23(2):229–236, 1991.

[36] Susan T. Dumais. LSI meets TREC: A status report. In Donna K. Harman, editor, *Proceedings of the First Text REtrieval Conference (TREC-1)*, pages 137–152. Department of Commerce, National Institute of Standards and Technology, 1992.

[37] Susan T. Dumais. Latent semantic indexing (LSI) and TREC-2. In Donna K. Harman, editor, *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pages 105–116. Department of Commerce, National Institute of Standards and Technology, 1993.

[38] Susan T. Dumais. Using LSI for information filtering: TREC-3 experiments. In Donna K. Harman, editor, *The Third Text Retrieval Conference (TREC-3)*, pages 219–230. Department of Commerce, National Institute of Standards and Technology, 1995.

[39] Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Scott Deerwester, and Richard Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of CHI '88: Conference on Human Factors in Computing*, pages 281–285. Association for Computing Machinery, 1988.

[40] C. Eckart and G. Young. A principle axis transformation for non-hermitian matrices. *Bulletin of the American Mathematical Society*, 45:118–121, 1939.

[41] Larry Fitzpatrick and Mei Dent. Automatic feedback using past queries: Social searching? In Nicholas J. Belkin, A. Desai Narasimhalu, and Peter Willett, editors, *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 306–315. Association for Computing Machinery, 1997.

[42] Martin Franz, J. Scott McCarley, and Salim Roukos. Ad hoc and multilingual information retrieval at IBM. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 157–168. Department of Commerce, National Institute of Standards and Technology, 1999.

[43] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science (FOCS '98)*, pages 370–378. IEEE Computer Society, 1998.

[44] Norbert Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[45] Atsushi Fujii and Tetsuya Ishikawa. Cross-language information retrieval for technical documents. In *Proceedings of the Joint ACL SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-99)*, pages 29–37. Association for Computational Linguistics, 1999.

[46] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the Eleventh International Conference on Research and Development in Information Retrieval*, pages 465–480. Association for Computing Machinery, 1988.

[47] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[48] Fredric C. Gey, Hailing Jiang, Aitao Chen, and Ray R. Larson. Manual queries and machine translation in cross-language retrieval and interactive retrieval with Cheshire II at TREC-7. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 527–540. Department of Commerce, National Institute of Standards and Technology, 1999.

[49] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 2nd edition, 1989.

[50] David Graff and Rebecca Finch. Multilingual text resources at the Linguistic Data Consortium. In *Proceedings of the 1994 ARPA Human Language Technology Workshop.* Morgan Kaufmann, 1994.

[51] David Haines and W. Bruce Croft. Relevance feedback and inference networks. In Robert Korfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of the Sixteenth Annual International ACM SIGIR Conference*

on Research and Development in Information Retrieval, pages 2–11. Association for Computer Machinery, 1993.

[52] Donna Harman. Overview of the First Text REtrieval Conference (TREC-1). In Donna K. Harman, editor, *Proceedings of the first Text REtrieval Conference (TREC-1)*, pages 1–20. Department of Commerce, National Institute of Standards and Technology, 1993.

[53] Donna Harman. Overview of the Second Text REtrieval Conference (TREC-2). In Donna K. Harman, editor, *Proceedings of the second Text REtrieval Conference (TREC-2)*, pages 1–20. Department of Commerce, National Institute of Standards and Technology, 1994.

[54] Donna Harman. Overview of the Third Text REtrieval Conference (TREC-3). In Donna K. Harman, editor, *Proceedings of the thrid Text REtrieval Conference (TREC-3)*, pages 1–20. Department of Commerce, National Institute of Standards and Technology, 1995.

[55] Donna Harman. Overview of the Fourth Text REtrieval Conference (TREC-4). In Donna K. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 1–23. Department of Commerce, National Institute of Standards and Technology, 1996.

[56] Donna Harman. Overview of the Fifth Text REtrieval Conference (TREC-5). In Donna K. Harman, editor, *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 1–28. Department of Commerce, National Institute of Standards and Technology, 1997.

[57] Donna K. Harman, editor. *Proceedings of the Second Text REtrieval Conference (TREC-2)*, Gaithersburg, Maryland, 1995. Department of Commerce, National Institute of Standards and Technology.

[58] Stephen P. Harter. A probabilistic approach to automatic keyword indexing: Part I. On the distribution of specialty words in a technical literature. *Journal of the American Society for Information Science*, 26(4):197–206, July-August 1975.

[59] Stephen P. Harter. A probabilistic approach to automatic keyword indexing: Part II. An algorithm for probabilistic indexing. *Journal of the American Society for Information Science*, 26(5):280–289, September-October 1975.

[60] David Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 282–291. Association for Computing Machinery, 1994.

[61] David A. Hull. *Information retrieval using statistical classification.* PhD thesis, Stanford University, November 1994.

[62] David A. Hull and Gregory Grefenstette. Querying across languages: A dictionary-based approach to multilingual information retrieval. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 49–57. Association for Computing Machinery, 1996.

[63] Fan Jiang, Ravindran Kannan, Michael L. Littman, and Santosh Vempala. Improved document sampling for efficient singular value decomposition. Technical Report CS–99–05, Duke University, Durham, NC, 1999.

[64] Noriko Kando, Kazuko Kuriyama, Toshihiko Nozue, Koji Eguchi, Hiroyuki Kato, and Soichiro Hidaka. Overview of IR tasks. In *Proceedings of the First NTCIR Workshop on Research in Japanese Text Retrieval and Term Recognition*, pages 11–44. National Center for Science Information Systems, Tokyo, Japan, 1999.

[65] Thomas K. Landauer and Michael L. Littman. Fully automatic cross-language document retrieval using latent semantic indexing. In *Proceedings of the Sixth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research*, pages 31–38. UW Centre for the New OED and Text Research, Waterloo Ontario, October 1990.

[66] Serge Lang. *Linear Algebra.* Springer-Verlag, New York, NY, 3rd edition, 1989.

[67] David D. Lewis and Richard M. Tong. Text filtering in MUC-3 and MUC-4. In *Proceedings of the Fourth Message Understanding Conference (MUC-4)*, pages 51–66, Los Altos, CA, 1992. Defense Advanced Research Projects Agency, Morgan Kaufmann.

[68] Michael Littman, Thomas K Landauer, and Dian Martin. Procrustes CL-LSI: Automatic multi-lingual information retrieval without parallel text or dictionaries. In preparation, 1999.

[69] Michael L. Littman, Susan T. Dumais, and Thomas K. Landauer. Automatic cross-language information retrieval using latent semantic indexing. In Gregory Grefenstette, editor, *Cross Language Information Retrieval*, pages 51–62. Kluwer Academic Publishers, Boston, MA, 1998.

[70] Michael L. Littman and Fan Jiang. A comparison of two corpus-based methods for translingual information retrieval. Technical Report CS–98–11, Duke University, Durham, NC, 1998.

[71] Michael L. Littman, Fan Jiang, and Greg A. Keim. Learning a language-independent representation for terms from a partially aligned corpus. In Jude Shavlik, editor, *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 314–322. Morgan Kaufmann, 1998.

[72] Julie Beth Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[73] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Cambridge, MA, 1999.

[74] Gregory B. Newby. Metric multidimensional information space. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, pages 521–536. Department of Commerce, National Institute of Standards and Technology, 1997.

[75] Gregory B. Newby. Context-based statistical sub-spaces. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, pages 735–745. Department of Commerce, National Institute of Standards and Technology, 1998.

[76] Gregory B. Newby. Information space gets normal. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 567–571. Department of Commerce, National Institute of Standards and Technology, 1999.

[77] Jian-Yun Nie, Michel Simard, Pierre Isabelle, and Richard Durand. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the web. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 74–81. Association for Computing Machinery, 1999.

[78] Douglas W. Oard and Jianqiang Wang. NTCIR CLIR experiments at the University of Maryland. In *Proceedings of the First NTCIR Workshop on Research in Japanese Text Retrieval and Term Recognition*, pages 157–162. National Center for Science Information Systems, Tokyo, Japan, 1999.

[79] Christos Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 159–168. ACM Press, 1998.

[80] Eugenio Picchi and Carol Peters. Cross language information retrieval: A system for comparable corpus querying. In Gregory Grefenstette, editor, *Cross-language Information Retrieval*, pages 81–92. Kluwer Academic Publishers, Boston, MA, 1998.

[81] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[82] Yonggang Qiu. *Automatic Query Expansion Based on a Similarity Thesaurus*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 1995.

[83] Yonggang Qiu and Hans-Peter Frei. Concept based query expansion. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–169. Association for Computing Machinery, 1993.

[84] Bob Rehder, Michael L. Littman, Susan Dumais, and Thomas K. Landauer. Automatic 3-language cross-language information retrieval with latent semantic indexing. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, pages 233–239. Department of Commerce, National Institute of Standards and Technology, 1998.

[85] Philip Resnik. Mining the web for bilingual text. In *37th Annual Meeting of the Association for Computational Linguistics (ACL'99)*, pages 527–534. Association for Computational Linguistics, 1999.

[86] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, May-June 1976.

[87] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. Harman, editor, *The Third Text Retrieval*

Conference (TREC-3), pages 219–230. Department of Commerce, National Institute of Standards and Technology, 1995.

[88] Stephen E. Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241. Springer Verlag, 1994.

[89] J. J. Rocchio. Relevance feedback in information retreival. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Retrieval*, pages 313–323. Prentice Hall, Englewood Cliffs, New Jersey, 1971.

[90] Sheldon M. Ross. *Introduction to Probability and Statistics for Engineers and Scientists.* John Wiley & Sons, Inc, New York, NY, 1987.

[91] Tetsuya Sakai, Yasuyo Shibazaki, Masaru Suzuki, Masahiro Kajiura, Toshihiko Manabe, and Kazuo Sumita. Cross-language information retrieval for NTCIR at Toshiba. In *Proceedings of the First NTCIR Workshop on Research in Japanese Text Retrieval and Term Recognition*, pages 137–144. National Center for Science Information Systems, Tokyo, Japan, 1999.

[92] Gerard Salton. Automatic processing of foreign language documents. *Journal of the American Society for Information Sciences*, 21:187–194, 1970.

[93] Gerard Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Retrieval.* Prentice Hall, Englewood Cliffs, New Jersey, 1971.

[94] Gerard Salton. *Dynamic Information and Library Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1975.

[95] Gerard Salton. *A Theory of Indexing.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1975.

[96] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer.* Addison-Wesley, Reading, MA, 1989.

[97] Gerard Salton and Chris Buckley. Term weighting approaches in automatic

text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[98] Gerard Salton and Chris Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[99] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, NY, 1983.

[100] Gerard Salton, A. Wang, and C. S. Yang. A vector space model for information retrieval. *Journal of the American Society for Information Science*, 18(11):613–620, November 1975.

[101] Peter Schäuble and Páraic Sheridan. Cross-language information retrieval (CLIR) track overview. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, pages 31–43. Department of Commerce, National Institute of Standards and Technology, 1998.

[102] Hinrich Schütze. Dimensions of meaning. In *Proceedings of the 1992 Conference on Supercomputing*, pages 787–796. Association for Computing Machinery, 1992.

[103] Hinrich Schütze, David A. Hull, and Jan O. Pedersern. A comparison of classifiers and document representations for the routing problem. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 229–237. Association for Computing Machinery, 1995.

[104] Páraic Sheridan and Jean Paul Ballerini. Experiments in multilingual information retrieval using the SPIDER system. In Hans-Peter Frei, Donna Harman, Peter Schäuble, and Ross Wilkinson, editors, *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 58–65. Association for Computing Machinery, 1996.

[105] Páraic Sheridan, Jean Paul Ballerini, and Peter Schäuble. Building a large multilingual test collection from comparable news documents. In Gregory Grefenstette, editor, *Cross-language Information Retrieval*, pages 137–150. Kluwer Academic Publishers, Boston, MA, 1998.

[106] Amit Singhal. *Term Weighting Revisited*. PhD thesis, Cornell University, Ithaca, NY, January 1997.

[107] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. Association for Computing Machinery, 1996.

[108] Amit Singhal, John Choi, Donald Hindle, and David D. Lewis. AT&T at TREC-7. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*. Department of Commerce, National Institute of Standards and Technology, 1999.

[109] Alan Smeaton and Ross Wilkinson. Spanish and Chinese document retrieval in TREC-5. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of The Fifth Text REtrieval Conference (TREC-5)*, pages 57–64. Department of Commerce, National Institute of Standards and Technology, 1997.

[110] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[111] Xiaobai Sun. Personal communication. 2000.

[112] John A. Swets. Information retrieval systems. *Science*, 141:245–250, July 1963.

[113] John A. Swets. Effectiveness of information retrieval methods. *American Documentation*, 20(1):72–89, January 1969.

[114] Howard Turtle. *Inference Networks for Document Retrieval*. PhD thesis, University of Massachusetts, Amherst, MA, February 1991.

[115] Ellen M. Voorhees. Natural language processing and information retrieval. In M. T. Pazienza, editor, *Information Extraction: Towards Scalable, Adaptable Systems*, pages 32–48. Springer, 1999.

[116] S. K. M. Wong and Y. Y. Yao. A note on inverse document frequency weighting scheme. Technical Report TR 89-990, Cornell University, Ithaca, NY, April 1989.

[117] S. K. M. Wong, Wojciech Ziarko, Vijay V. Raghavan, and P. C. N. Wong. On modeling of information retrieval concepts in vector spaces. *ACM Transactions on Database Systems*, 12(2):299–321, June 1987.

[118] S. K. M. Wong, Wojciech Ziarko, and Patrick C. N. Wong. Generalized vector space model in information retrieval. In *Proceedings of the Eighth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 18–25. Association for Computing Machinery, 1985.

[119] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 4–11. Association for Computing Machinery, 1996.

[120] Yiming Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Delevopment in Information Retrieval (SIGIR '95)*, pages 256–263. Association for Computing Machinery, 1995.

[121] Yiming Yang, Jaime G. Carbonell, Ralf D. Brown, and Robert E. Frederking. Translingual information retrieval: Learning from bilingual corpora. *Artificial Intelligence*, 103(1–2):323–345, 1998.

[122] Hongyuan Zha, Osni Marques, and Horst Simon. Large-scale SVD and subspace-based methods for information retrieval. In Afonso Ferreira, José D. P. Rolim, Horst Simon, and Shang-Hua Teng, editors, *Solving Irregularly Structured Problems in Parallel, 5th International Symposium, IRREGULAR '98, Proceedings. Lecture Notes in Computer Science, Vol. 1457*, pages 29–42. Springer-Verlag, 1998.

[123] Hongyuan Zha and Horst D. Simon. On updating problems in latent semantic indexing. *SIAM Journal on Scientific Computing*, 21(2):782–791, 1999.

[124] Hongyuan Zha and Zhenyue Zhang. On matrices with low-rank-plus-shift structures: Partial SVD and latent semantic indexing. *SIAM Journal on Matrix Analysis and Applications*, 21(2):522–536, 1999.

Biography

Fan Jiang was born on June 10, 1974 in Nantong, Jiangsu, China. He grew up in Shanghai, China and arrived in the United States in January 1992. He has received the following degrees:

- Ph.D. in Computer Science, Duke University, 2000.
- M.S. in Computer Science, Duke University, 1999.
- B.S. in Computer Science, University of Houston, 1996.

His published articles include:

- "Approximate dimension equalization in vector-based information retrieval"
- "Proverb: The probabilistic cruciverbalist"
- "Learning a language-independent representation for terms from a partially aligned corpus"

He has received the following fellowships and honors:

- Computer Science Department Fellowship, Duke University, 1996–1997
- National Science Foundation Research Experiences for Undergraduates Award, 1995–1996.
- Albert Newhouse Memorial Fellowship, University of Houston, 1995.
- Golden Key National Honor Society, 1994.
- Phi Kappa Phi National Honor Society, 1994.
- Dean's List (5 Semesters), University of Houston, 1993-1996.
- Alpha Lambda Delta Freshman Honor Society, 1993.
- Phi Eta Sigma Freshman Honor Society, 1993.