Cross-Language Information Retrieval Using Latent Semantic Indexing

Paul G. Young

Computer Science Department

CS-94-259 October 1994

Cross-Language Information Retrieval Using Latent Semantic Indexing

A Thesis

Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Paul G. Young

December 1994

Acknowledgements

I thank my thesis advisor, Dr. Michael Berry, for his support and guidance. I wish to express my deepest appreciation for his patience, direction, and boundless energy. I also thank Dr. Brad Vander Zanden and Dr. David Straight who served on my thesis committee. I also thank my wife, Susan, for her proofreading which gave Dr. Berry a big break. I thank my daughter for her patience in waiting for Dad while I sat at the computer console for what seemed an eternity to her.

Abstract

In this thesis, a method for indexing cross-language databases for conceptual query-matching is presented. Two languages (Greek and English) are combined by appending a small portion of documents from one language to the identical documents in the other language. The proposed merging strategy duplicates less than 7% of the entire database (made up of different translations of the Gospels). Previous strategies duplicated up to 34% of the initial database in order to perform the merger. The proposed method retrieves a larger number of relevant documents for both languages with higher cosine rankings when Latent Semantic Indexing (LSI) is employed.

Using the proposed merge strategies, LSI is shown to be effective in retrieving documents from either language (Greek or English) without requiring any translation of a user's query. An effective Bible search product needs to allow the use of natural language for searching (queries). LSI enables the user to form queries with using natural expressions in the user's own native language. The merging strategy proposed in this study enables LSI to retrieve relevant documents effectively while duplicating a minimum of the entire database.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The need to search multilanguage databases with only the knowledge of one language arises in many fields of study such as international law, medicine, physics, theology, mathematics, or scientific computing. The problem is one of translation. In order to match queries with text from other languages, translation of the query to such languages is sometimes necessary. Queries may be translated differently, affecting the performance of searches, because translation involves interpretation of the source and target languages. When the translation of the query and the translation of the text use different terms, sometimes relevant documents are not returned.

The primary concern of this thesis is multiple translations of a single work, the Bible. These translations differ at various scales ranging from only a word to the complete paraphrasing of the text. The languages that differ the most and have no direct translation into English are Hebrew and Greek, from the Old and New Testament, respectively. Current software can only search one translation at a time using literal-matching schemes. These include, but are not restricted to, the following: QuickVerse (Parsons Technology), Wordsearch (NavPress Software), Online Bible (shareware), BibleWorks (Hermeneutica), PC Study Bible (BibleSoft), Logos Bible Software (Logos), Deluxe Bible for Windows (Rocky Mountain Laboratories), and Multimedia Family Bible (Candlelight Publishing) [Hew93]. These software products have linked the translations together so that if a verse is returned in one translation, the same verse will also be displayed in the other translations. Cross-referencing performed by these products has been done using look-up tables, which may require many man-hours to code properly. Topical searches are either programmed into the software (where the user has to decipher layers of menus to find the appropriate topic), or such searches do not exist. Most users have a limited knowledge of the Bible, i.e., they know that a verse or story is contained in the Bible but are not able to remember the exact words. Because of this typical searching scenario, lexical-matching schemes may not perform well. For the average user of Bible search software, the ability to retrieve verses, stories, or even topics from the Bible using their own words (i.e., natural language) is a very attractive option.

1.2 Overview

In this thesis, an approach to automatically index a cross-language database of the Gospels is evaluated. Within this chapter, the indexing scheme used for query matching, Latent Semantic Indexing (LSI) [DDF+90], and the singular value decomposition (SVD) [GL89] which is used to define the underlying word structure of data are both explained. Starting with four English versions of the Gospels, the performance of LSI is evaluated. Then, a two-language database composed of Greek and English is used to test two LSI-based methods for automatically indexing the database. One method is a coarse-grained combination of data similar to that used by Landauer et al. [LL90]. The coarse-grained method takes whole documents from both languages and combines them. The second method is a fine-grained combination of data. The fine-grained method takes a verse (sentence) from the beginning of a document in one language and combines it with a corresponding verse (sentence) from the other language. Landauer et al. used bilingual training sets [LL90] (composed of complete documents in both languages) to combine languages. While training sets of whole documents are used for comparison in this thesis, training sets using *small* portions of each document are shown to combine languages in a more global fashion.

Particular problems of dealing with cross-language databases are presented in Chapter 2. In Chapter 3 a progression of three different case studies is explained. The first study uses four English translations of the Gospels with verses as documents. For the next study, the document size is increased to an average of ten verses per document, and the final study uses English and Greek translations with larger document sizes. In Chapter 4 the performance of the different approaches for optimizing cross-language databases is analyzed. Finally, Chapter 5 discusses the future possibilities for cross-language database indexing, whether the database is a translation of the Bible or technical journals for a multilingual scientific community.

An effective Bible search product needs to allow the use of natural language for searching (queries). LSI has been shown to be very effective for this type of searching. In the next section, LSI is introduced as a way of computing a low-rank approximation to an original term-document matrix using the SVD.

1.3 Preliminaries

1.3.1 Latent Semantic Indexing

Latent Semantic Indexing (LSI) is initially applied to a matrix of terms by documents. Bible verses in English and Greek will be used to demonstrate the LSI process. The verses (see Table 1.1) are in English (E1-4) and Greek (G1-4). The notation **(KJV)** and **(HG)** refer to the *King James* and *Greek* versions of the verses, respectively. Terms indexed are underlined in Table 1.1 and appear more than once in the small database. Table 1.2 illustrates the corresponding 14×8 term by document matrix.

In general, a term by document matrix is defined as

$$A = [a_{ij}], \tag{1.1}$$

where a_{ij} is the frequency that term i occurs in document j. It is possible by preprocessing (or reweighting of the rows or columns of A) to limit the differential effect of document length or to impose preconceptions of which terms are more important. In other words, elements of matrix A may be given by

$$a_{ij} = L(i,j) \times G(i), \tag{1.2}$$

where $L(i,j)$ is the local weighting for term i in document j, and $G(i)$ is the global weighting for term i. In the example from Table 1.2, local term frequency weighting and no global weighting (i.e., $G(i) = 1$) are used. However, the experiments in this thesis used local logarithmic term weighting

$$L(i,j) = \log(a_{ij} + 1), \tag{1.3}$$

with global entropy weighting

Table 1.1: Bible verses used as documents in English and Greek.

Document	Text
E1	**Mat 3:17 (KJV)** And lo a voice from heaven, saying, This is my beloved Son, in whom I of heaven am well pleased.
E2	**Mat 12:18 (KJV)** Behold my servant, whom I have chosen; my beloved, in whom my soul is well pleased: I will put my spirit upon him, and he shall shew judgment to the Gentiles.
E3	**Mat 17:5 (KJV)** While he yet spake, behold, a bright cloud overshadowed them: and behold a voice out of the cloud, which said, This is my beloved Son, in whom I am well pleased; hear ye him.
E4	**Mark 9:7 (KJV)** And there was a cloud that overshadowed them: and a voice came out of the cloud, saying, This is my beloved Son: hear him.
G1	**Mat 3:17 (HG)** . idou. phone ouranos lego houtos agapetos huios . hos.ouranos . eudokeo eudokeo
G2	**Mat 12:18 (HG)** idou. pais hos. . hairetizo. agapetos. hos. psuche. eudokeo eudokeo. . tithemi. pneuma. . apaggello krisis. . ethnos.
G3	**Mat 17:5 (HG)** eti laleo idou. photeinos nephele episkiazo . idou. phone. . nephele. . lego houtos . agapetos huios paroikeo .eudokeo eudokeo akouo . .
G4	**Mark 9:7 (HG)** . nephele. episkiazo. . . phone erchomai . nephele lego houtos. . agapetos huios akouo.

Table 1.2: Term by document matrix.

Term	E1	E2	E3	E4	G1	G2	G3	G4
voice	1		1	1				
heaven	1	1						
saying	1			1				
beloved	1	1	1	1				
son	1		1	1				
pleased	1	1	1	1				
cloud			2	1				
phone					1		1	1
ouranos					1	1		
lego					1		1	1
agapetos					1	1	1	1
huios					1		1	1
eudokeo					2	2	2	2
nephele							2	1

$$G(i) = 1 - \Sigma_j \frac{p_{ij}\log(p_{ij})}{\log(n\text{docs})}, \tag{1.4}$$

where

$$p_{ij} = \frac{a_{ij}}{gf_i},$$

with gf_i representing the number of occurances of the term in the collection, and $n$docs is the number of documents in the collection [Dum91].

This log-entropy weighting scheme is similar to that of Landauer et al. [LL90]. The local logarithmic term weighting from Equation (1.3) is a compromise between the common practice of binary weights (one if present, zero otherwise) and the raw frequency (a_{ij}) as an indicator of the degree to which a document is characterized by the term. Regardless of the weighting scheme used, most terms do not occur in every document so that the matrix A in Equation (1.1) tends to be extremely sparse.

1.3.2 Singular Value Decomposition

The **singular value decomposition** (SVD) of the sparse matrix A is given by

$$A = U\Sigma V^T \tag{1.5}$$

where U and V are orthogonal matrices and Σ is the diagonal matrix of singular values [Ber92]. Specifically, the columns of U and V contain the left and right singular vectors, respectively, corresponding to the singular values of A (see Figure 1.1).

The following two theorems illustrate how the SVD can reveal important information about the structure of a matrix.

Theorem 1. Let the SVD of A be given by Equation (1.5) and

$$\sigma_1 \geq \sigma_2 \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$$

and let $R(A)$ and $N(A)$ denote the range and null space of A, respectively. Then

1. rank property: $\text{rank}(A) = r$, $N(A) \equiv \text{span}\{v_{r+1}, \cdots, v_n\}$, and $R(A) \equiv \text{span}\{u_1, \cdots, u_r\}$, where $U = [u_1 u_2 \cdots u_m]$ and $V = [v_1 v_2 \cdots v_n]$.
2. dyadic decomposition: $A = \sum_{i=1}^{r} u_i \cdot \sigma_i \cdot v_i^T$.
3. norms: $\|A\|_F^2 = \sigma_1^2 + \cdots + \sigma_r^2$, and $\|A\|_2^2 = \sigma_1$.

Theorem 2. [Eckart and Young] Let the SVD of A be given by Equation (1.5) with $r = \text{rank}(A) \leq p = \min(m, n)$ and define

$$A_k = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^T. \tag{1.6}$$

Then,

$$\min_{r(B)=k} \|A - B\|_F^2 = \|A - A_k\|_F^2 = \sigma_{k+1}^2 + \cdots + \sigma_p^2.$$

(For a proof, see [GL89]).

The $m \times n$ matrix A_k, which is constructed from the k-largest singular triplets of A, is the closest rank-k matrix to A [GL89]. In fact, A_k is the best rank-k approximation to A for any unitarily invariant norm [Mir60], i.e.,

$$\min_{r(B)=k} \|A - B\|_2 = \|A - A_k\|_2 = \sigma_{k+1}.$$

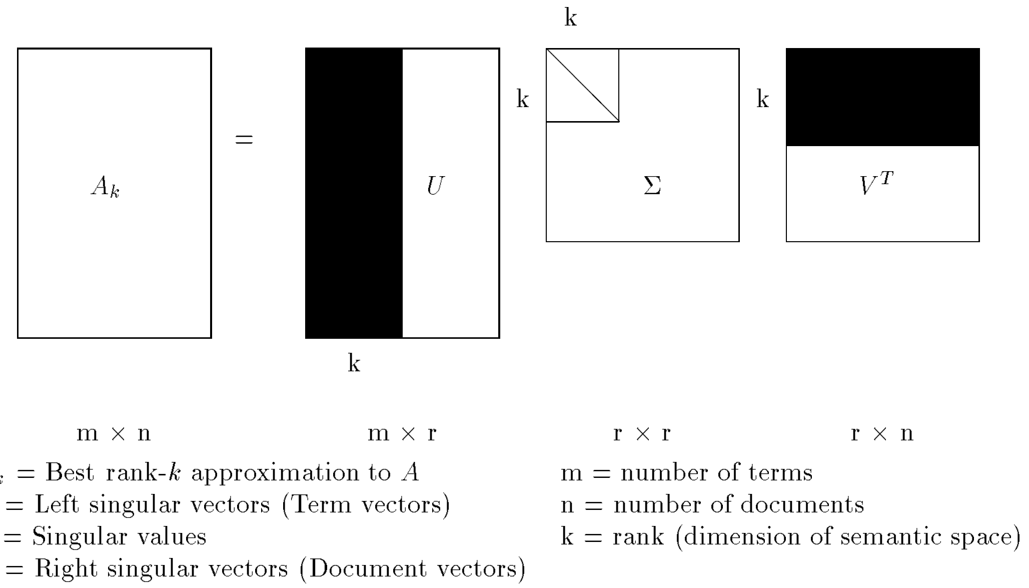
k
k
k
=
A_k
U
Σ
V^T
k
m × n
m × r
r × r
r × n
A_k = Best rank-k approximation to A
U = Left singular vectors (Term vectors)
Σ = Singular values
V = Right singular vectors (Document vectors)
m = number of terms
n = number of documents
k = rank (dimension of semantic space)

Figure 1.1: Mathematical representation of the matrix A_k from Equation (1.6).

Following Theorem 2, the best rank-k approximation (A_k) to A is given by

$$A_k = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^T \;\; \textit{with} \;\; k < r \qquad (1.7)$$

where $\{u_i, \sigma_i, v_i\}$ is the i-th largest singular *triplet* of the sparse matrix A [Ber92], and $r = \text{rank}(A)$. Using LSI, the optimal value for k is data dependent, but $k \in [100, 200]$ has been suggested [DDF$^+$90]. Fortunately, the value of k is usually much smaller than the rank (r) of the matrix A.

The shaded region in Figure 1.1 reflects the reduced (rank-k) or truncated SVD of A, which is used by LSI to capture the semantic structure of word usage. If $k = r$, LSI performs similarly to literal-matching. If $k = 1$, LSI treats all documents and words as related.

By using the *reduced model* in Equation (1.7), minor differences in terminology (noise) are virtually ignored. Moreover, the closeness of objects is determined by the overall pattern of term usage, so documents can be classified together (i.e., word choice suppression) regardless of the precise words that are used to describe them. The description of documents depends on a consensus of their term meanings, thus dampening the effects of polysemy (words with multiple meanings). Terms that do not actually appear in a document may still be used as referents, if they are consistent with the major patterns of association in the data. The position in the reduced space ($\mathrm{R}(A_k)$) then serves as a new kind of *semantic indexing* [Ber92]. Therefore, when using the truncated SVD, LSI achieves *noise* reduction (variable of word choice suppression) via the reduced model A_k.

1.3.3 Multi-Dimensional Representation

Each term and document is represented by a vector in k-space using elements of the left or right singular vectors. By using the reduced model A_k to approximate the original matrix A, documents comprised of different terms may be potentially mapped into nearly the same vector. Terms used similarly in documents also may be mapped to *close* vectors. A rank-2 approximation to matrix A (denoted by A_2) uses the first two elements of each row of U and V from Equation (1.5) to plot the terms and documents respectively. The graph in Figure 1.2 illustrates the coordinates of terms and documents on the Cartesian plane using A_2 to approximate the term by document matrix in Table 1.2. LSI typically uses the cosines between vectors as an indicator of relevance within a multi-dimensional space composed of term and document vectors.

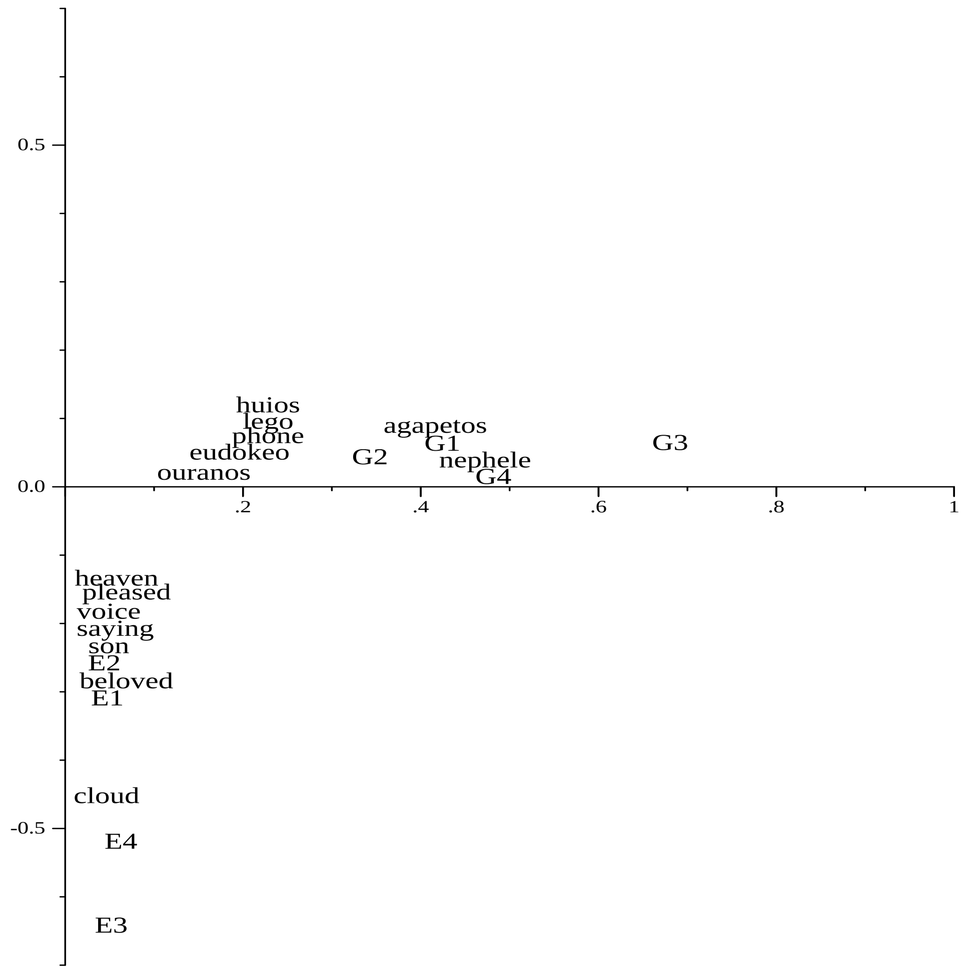

Figure 1.2: Two-dimensional plot of terms and documents.

If k is too small, the *relevance* between documents may be abnormally forced, as reflected by cosines close to 1. A higher (approaching 1) cosine value should not be used to indicate semantic similarities between documents because a high cosine value is only a potential *indicator* of relevance, however, and not a *guarantor* of relevance.

In Figure 1.2, the terms and documents of the English version are clustered along the y-axis, and the terms and documents of the Greek version are clustered along the x-axis. While the terms and documents are clustered *by version* in the multi-dimensional space, there is a possibility that the languages may be placed in disjoint regions of the multi-dimensional space, as illustrated by Figure 1.2.

1.3.4 Queries

A query can be represented in the same multidimensional space generated by A_k. Suppose the term vector t_v is composed of occurrences of each term in Table 1.2 within a particular query. The term vector t_v can then be represented as a *pseudo-document* q via $q = t_v^T U_k \Sigma_k^{-1}$. For example, suppose one is interested in Bible verses where God spoke or people could *hear a voice from heaven* . A term vector for this query is built using the terms in the query (*hear a voice from heaven*). A list of common words such as $\{$*a, and, from, or*$\}$ are not parsed and are not used as referents to verses. Also, *hear* is not an indexed term in the database, so it is dropped from the query leaving **voice heaven**. The parsing rule used in this thesis requires that a term must appear more than once in the database for the term to be included in the term by document matrix A. Of course, other parsing rules may allow unique terms to be included in the matrix A. The Cartesian coordinates of the query **voice heaven** are derived in Figure 1.3.

Since a query (or pseudo-document) can be represented in the same semantic space as all other documents, it can be compared for similarity with all the documents in the LSI-generated database. All documents whose cosine with the query vector is greater than a threshold value such as .90 may be returned. The shaded region in Figure 1.4 illustrates those documents for our 14×8 example which satisfy this constraint. Any documents within the shaded region are considered above the threshold (.90) and are returned. None of the Greek terms or documents are returned and all of the English terms and documents are returned. This illustrates a major problem

Figure 1.3: Coordinates of the query *voice heaven.*

$$(\; 0.1047, \; -0.0957 \;) = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}^T \begin{pmatrix} 0.0190 & -0.2474 \\ 0.3245 & -0.0368 \\ 0.1961 & 0.0183 \\ 0.0148 & -0.1958 \\ 0.0465 & -0.6028 \\ 0.4164 & 0.0491 \\ 0.2282 & 0.0190 \\ 0.2282 & 0.0190 \\ 0.4724 & 0.0324 \\ 0.2282 & 0.0190 \\ 0.0120 & -0.1818 \\ 0.1560 & 0.0204 \\ 0.0163 & -0.1830 \\ 0.0190 & -0.2474 \end{pmatrix} \begin{pmatrix} 3.2801 & 0 \\ 0 & 2.9685 \end{pmatrix}^{-1}$$

for query-matching within a cross-language database. Of course, a different cosine threshold could have been used so that a larger set of documents would be returned, although the LSI-rank of the Greek documents (according to cosines with query vectors) would be very low. The cosine is merely used to rank-order documents and its explicit value is not always an adequate measure of relevance when $k \in [100, 200]$.

1.3.5 Folding-In

Suppose an LSI-generated database already exists; that is, a body of text has been parsed, a term-document matrix has been generated, and the SVD of the term-document matrix has been computed. If more terms and documents must be added, two alternatives for incorporating them currently exist: recomputing the SVD of a new term-document matrix or *folding-in* the new terms and documents. Recomputing the SVD of a larger term-document matrix requires more computation time and, for large problems, may be impossible due to memory constraints. Folding-in requires less time and memory but can have deteriorating effects on the representation of the

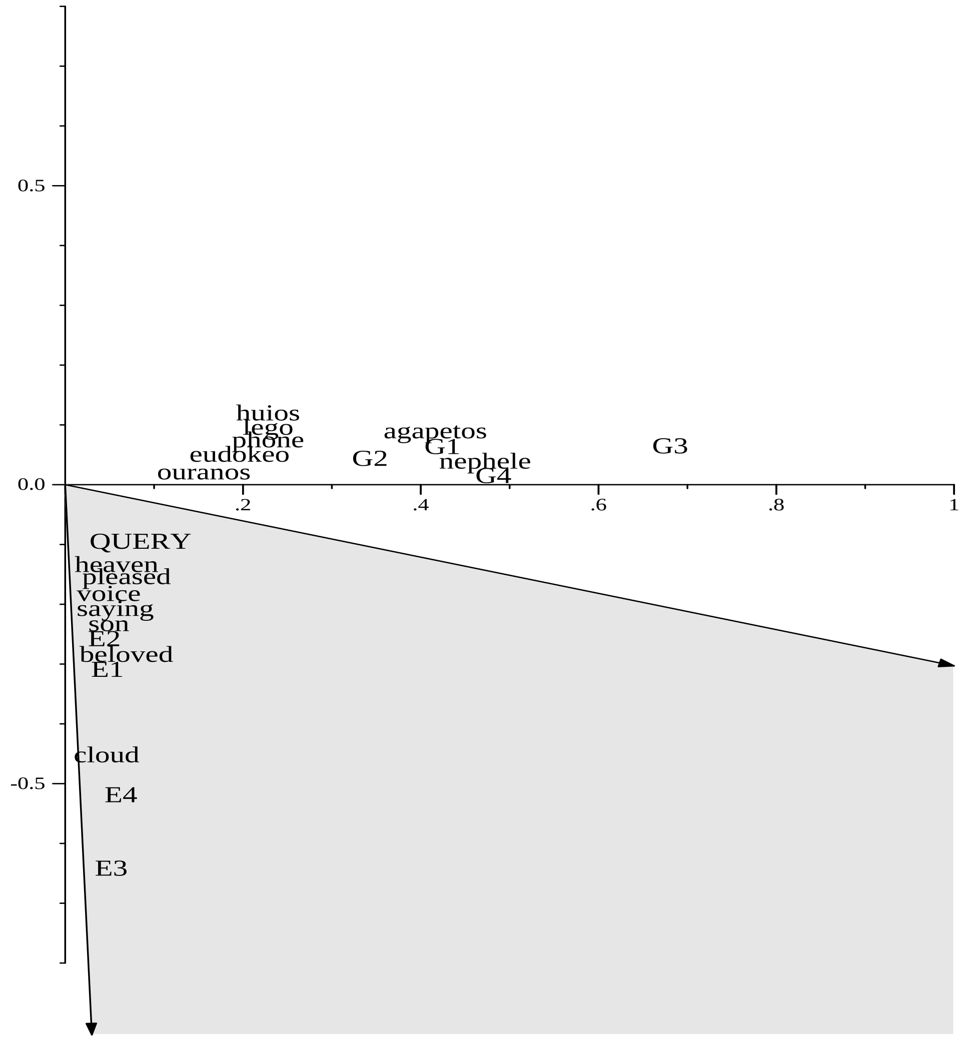

Figure 1.4: Two-dimensional plot of the documents returned for the query **voice heaven.**

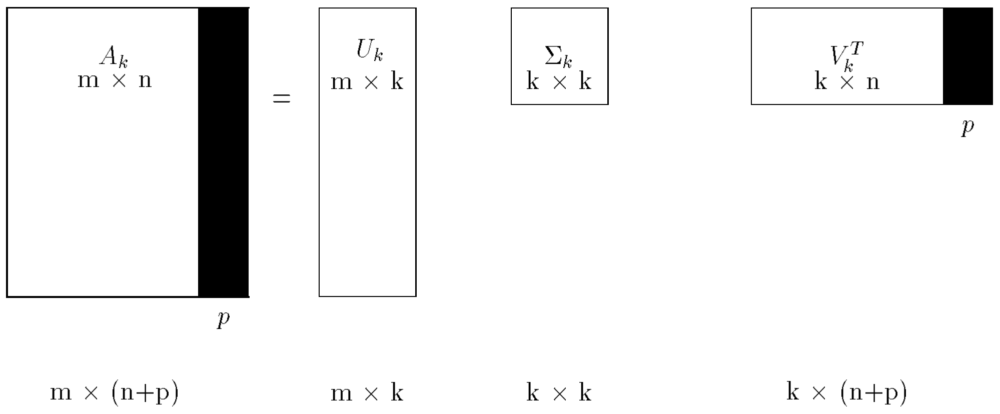

Figure 1.5: Mathematical representation of folding-in p documents.

new terms and documents [BDO95]. Folding-in is based on the existing latent semantic structure (the current A_k), and hence new terms and documents have no effect on the representation of the pre-existing terms and documents. Folding-in documents is essentially the process described in Section 1.3.4 for pseudo-document representation of queries. Each new document is represented as a weighted sum of its component term vectors. Once a new document vector has been computed, it is appended to the set of existing document vectors or columns of V_k (see Figure 1.5). Similarly, new terms can be represented as a weighted sum of their component document vectors. Once the term vector has been computed, it is appended to the set of existing term vectors or columns of U_k (see Figure 1.6).

1.4 Cross-Language Databases

It is precisely because of LSI's ability to suppress *noise,* exploit the *semantic structure,* and process queries as pseudo-documents, that it has been used for information retrieval from cross-language databases [LL90]. The critical factor for effective information retrieval in a cross-language database is the *orthography* of individual words,

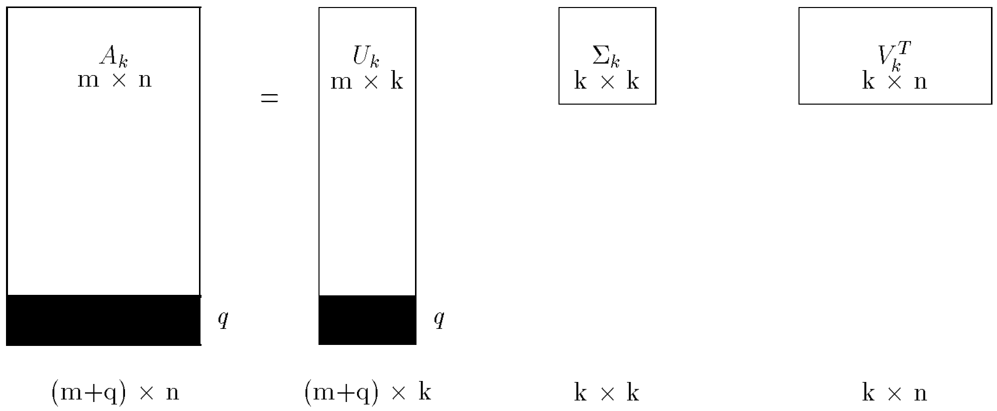

Figure 1.6: Mathematical representation of folding-in q terms.

i.e., word-for-word correspondence between languages. If the translations were word-for-word translations of each other, then the semantic structure would be exactly the same. This is unrealistic, of course, but the closer the translations are to each other, the closer their orthography.

In Landauer et al. [LL90] *training sets* are used to accomplish the orthography of words translated in both languages. They started with 2,482 *documents* containing at least five lines for each document in both English and French versions (for a total of 4,964 documents). The documents in each language averaged 85 words (84 in English and 86 in French). The training sets used 300, 600, and 900 randomly sampled sets of documents (matching documents from both languages) which were then processed using the local and global term weightings specified by equations 1.3 and 1.4 respectively. Each training set was tested with the remaining sets of documents (out of a total of 2,482 documents) *folded into* the space (see Section 1.3.5). Specifically, a document not found in the training sets is treated as a query or pseudo-document, and the highest-ranked documents from the other language (as determined by cosines) are returned. For the 300-, 600-, and 900-document training sets, the top returned document from the second language was the identical document to the query 78%, 87%, and 92% of the time, respectively. According to the authors, the tests they conducted involved what might be called *ideal* queries. These queries express the meaning of the desired target document as nearly as possible, in the same way and in the same style as the original document , but use a different language. Landauer et al. suggested testing the performance of ordinary, less well formed queries, only speculating the performance would be less favorable.

1.5 Database Merge Strategy

In Landauer et al. [LL90], there was no indication of how high the LSI-rankings were for the returned documents. Returned documents may actually be so far down the LSI ranking (below 25) that they may not be useful. The LSI-ranking of the top returned document of the cross-language was evaluated, but nothing was reported about how many of the other documents in the cross-language database were returned as, for example, the top 50 LSI-ranked documents. This would have given more information about the placement of documents of both languages in the multi-dimensional vector space.

In this thesis a strategy of automatically combining less than 8% of each document (about one sentence) using LSI will be shown to perform well. The LSI-ranking of the returned documents of the other language are typically higher, and there is more of a mixing of the languages in the list of top 50 returned documents using the proposed strategy. It is obvious that if one needs only to duplicate the size of less than 8% of the database, one will need to use much less memory space in storage than a strategy that requires duplication of 36% of the database [LL90].

Chapter 2

Cross-Language Databases

2.1 Cross-Language Difficulty

While LSI's use of the SVD can be effective in representing the semantic structure of documents in a multi-dimensional space, there are problems with cross-language documents. Since there may be few common words between the two languages (e.g., French and English), LSI can (geometrically) place documents of the two languages at disjoint regions of the multi-dimensional space (recall Figure 1.2). In other words, the lack of common words precludes the geometrical representation of cross-language documents in close proximity (via the SVD).

2.2 Creating Common Terms

Different documents can contain material that is conceptually the same without using similar words. One document might use the word *mice* to describe a computer input device while another document about mice might use the words *pointing device.* Users create queries by concept, and the words the user chooses for the query often do not match the words used in relevant documents. An optimal indexing strategy would allow for such differences to be taken into account. This is especially true of cross-language databases where terms may be found translated in a second language with conceptually similar documents. A multi-dimensional space that represents *similar terms* from both languages as close as possible to each other is very desirable. Using such a space, LSI should return highly-ranked but relevant documents for the more usual *short* queries (few keywords) that users have.

2.3 Two Approaches

This thesis will show that by combining a low percentage of each document in both languages, an effective cross-language retrieval system can be built. Experiments using the Gospels of the Bible have been conducted to assess LSI's ability to return relevant documents among four translations. The translations are *The King James, The Living Bible, The New International Version,* and *The Revised Standard version.* A set of queries from the *New International Version Study Bible* [BBS+85], has been constructed from phrases used to describe the miracles and life of Jesus (see Table 2.1).

The same queries have been used in all of our tests to simplify the interpretation of the results. An appropriate number of factors (k) has been empirically determined to ensure that LSI performs well from a user's perspective, i.e., relevant documents are returned with a high enough LSI-ranking that the documents can be quickly identified (not just in the top 50, but the top 10-20 returned documents). After effective performance is obtained from LSI using the four English translations, the two languages (English and Greek) are combined using two different strategies.

The first strategy for creating a base of common terms in the database mimics

Table 2.1: Queries used for testing.

Number	Query
1	The Baptism of Jesus
2	Healing of Bartimaeus
3	Tribute to Caesar
4	Cleansing the Temple
5	The Great Commandment
6	Entry into Jerusalem
7	Epileptic Boy Healed
8	Feeding Five Thousand
9	Fig Tree Cursed
10	Lamp Under a Bowl
11	New Cloth Old Coat
12	New Wine Old Wineskins
13	Sower and the Soils
14	Mustard Seed
15	Tenants
16	Fig Tree

the previous study by Landauer et al. [LL90]. Their strategy used a range of 8.27% to 36.26% of the database to create the common term base. In this thesis 1%, 2%, 4%, 8%, 16%, 24%, and 32% of the database is combined to create the common term base. In this way the effectiveness of the strategy proposed by Landauer et al. is studied in more detail.

The second strategy involves combining a smaller portion of each document across the entire database. This strategy combines one verse of each document to create the new common term base (using only 7.7% of the database). The resulting LSI model for both strategies is tested using the same set of queries, and the number of documents returned in the Greek text is recorded. All the returned LSI-rankings for Greek text are recorded in order to better understand the amount of mixing and reliability of the resulting LSI model. A series of three case studies (discussed in Chapter 3) is used to test the performance of the different LSI models.

Chapter 3

Design of Case Studies

LSI models for cross-language databases can be created using different weighting schemes, numbers of factors (k), and different language merge strategies. In this chapter, three case studies are used to study the performance of the different LSI models. The first involves four English translations of the Gospels using individual verses as the documents. The same four translations are organized into stories (combining an average of 10 verses) that are used as documents in the second case study. Finally, a Greek and an English version are combined using the stories as documents.

3.1 Case1: Four Translations

Starting with four versions of the Gospels, different numbers of LSI factors (k) are tested. The four translations used are (i) *The King James Version,* a translation of the Vulgate (Latin) version of the Bible; (ii) *The Revised Standard Version,* a translation of the Greek and Hebrew with an attempt to update the language to a more modern vocabulary; (iii) *The New International Version,* a more recent translation of the

Table 3.1: Locations of queries in Gospels.

Query Number	Matthew	Mark	Luke	John
1	Mt.3:13-17	Mk. 1:9-11	Lk 3:21-23	Jn 1:29-39
2	Mt 20:29-34	Mk 10:46-52	Lk 18:35-43	
3	Mt 22:15-22	Mk 12:13-17	Lk 20:20-26	
4	Mt 21:12-13	Mk 11:12-14		Jn 2:14-22
5	Mt 22:34-40	Mk 12:18-31		
6	Mt 21:1-11	Mk 11:1-10	Lk 19:29-44	Jn 12:12-19
7	Mt 17:14-18	Mk 9:17-29	Lk 9:38-43	
8	Mt 14:15-21	Mk 6:35-44	Lk 9:12-17	Jn 6:5-13
9	Mt 21:18-22	Mk 11:12-14,20-25		
10	Mt 5:14-15	Mk 4:21-22	Lk 8:16;11:33	
11	Mt 9:16	Mk 2:21	Lk 5:36	
12	Mt 9:17	Mk 2:22	Lk 5:37-38	
13	Mt 13:3-8,18-23	Mk 4:3-8,14-20	Lk 8:5-8,11-15	
14	Mt 13:31-32	Mk 4:30-32	Lk 13:18-19	
15	Mt 21:33-44	Mk 12:1-11	Lk 20:9-18	
16	Mt 24:32-35	Mk 13:28-29	Lk 21:29-31	

Greek and Hebrew where the writers try to keep the poetry of the King James Version; and (iv) *The Living Bible,* a paraphrase of the Bible.

The Gospels are useful for testing purposes since many of the stories are shared, especially in the Synoptic (first three) Gospels: Matthew, Mark, and Luke. In other words, it is possible to determine which returned documents are misses (false hits), which documents are related to the query (precision), and how many of the known relevant documents are returned (recall).

The queries obtained from the NIV Study Bible [BBS$^+$85] are descriptive titles containing either parables or miracles of Jesus appearing in more than one of the Gospels. This implies that there should be more than one relevant document or verse returned depending upon the number of appearances of the story in the four Gospels. Table 3.1 shows where the queries are found in the different Gospels.

3.2 Case2: Story Versions

The relatively low number of terms (average of 6.28) contained in a given Bible verse suggest that document or verse similarities may be adequately judged by using a literal-matching scheme as opposed to using LSI. The semantic structure of documents is limited by such a small document size. However, for a total of 18,895 verses, LSI is still able to return a high number of hits (relevant verses). Determining the relevancy of verses without seeing them in their proper context, however, is not an easy task. The usefulness of such verses out of context is also questionable.

An alternative contextual approach for evaluating the performance of LSI is to construct a story version of the Gospels, i.e., group verses by context. This approach uses documents similar in size to the database used by Landauer et al. [LL90] whose documents averaged 85 words. The highest number of hits returned is found when slightly over 100 factors or dimensions are used (113 factors to be exact). With this new LSI-generated database, a smaller number of documents (1471) is used with a larger document size available (80.62 words per document average). Each document contains an average of 12.85 verses, and varies in size from 5 to 45 verses. While LSI returns a good mixture from the different versions (translations) of a given query, all verses processed contain terms from the English language (i.e., no Greek terms).

3.3 Case3: Greek and King James

A combination of the Greek and King James versions of the Gospels has been constructed as a cross-language database of the Gospels. The reason for this particular combination lies in the fact that the King James Version originated from Latin not Greek and, therefore, should pose a strong challenge for LSI. Also Greek has few adjectives, pronouns, or other parts of grammer which are found in English, so word order is not important. If LSI can determine the appropriate semantic relationships between Greek words, there is little doubt that LSI can perform well with other translations of the Gospels.

The work done previously by Landauer et al. [LL90] used the strategy of combining complete documents together (see Section 1.4). They combined varying portions (12.08%, 24.17%, and 36.26%) of the database by taking *equivalent* documents in each language and forming a single document from both. The combined portion of the database was run through LSI and the remaining documents *folded-in.*

The LSI-rank of returned Greek documents may be affected by queries containing terms used in English titles. Terms used in a query which are also found in the title of a Greek document may raise the LSI-rank of the document because titles occur only in English for both the English and Greek versions. This will be especially true of smaller Greek documents (less than 5 verses) because the title will make up a larger percentage of the document. To help understand the performance of different queries, a list of query terms is included in Table 3.2.

Two different ways of combining English and Greek text are examined in this thesis. First, whole documents are combined using different portions of the Greek documents (1.09%, 2.18%, 4.36%, 8.72%, 17.44%, 26.16%, and 34.88%). This is accomplished by taking the Greek documents and appending them to the end of the English documents. The previous percentages reflect the different numbers of Greek documents appended to the English documents (1, 2, 4, 8, 16, 24, and 32 Greek documents, respectively). The second approach takes a single verse from the beginning of each Greek document and adds it to the end of the first verse in the corresponding English document. Since there are 9448 verses and 734 documents

Table 3.2: Query terms in titles and documents.

Query Number	Term	Number Found in Titles	Number Found in Documents (Not Titles)
1	Baptism	12	26
1	Jesus	580	616
2	Healing	40	4
2	Bartimaeus	4	1
3	Tribute	0	10
3	Caesar	12	19
4	Cleansing	0	2
4	Temple	28	65
5	Great	28	150
5	Commandment	8	32
6	Entry	16	0
6	Jerusalem	4	68
7	Boy	16	0
7	Healed	8	34
8	Feeding	0	4
8	Five	16	32
8	Thousand	24	14
9	Fig	20	21
9	Tree	16	45
9	Cursed	0	3
10	Lamp	12	20
10	Under	0	63
11	New	0	77
11	Cloth	0	43
11	Old	16	303
11	Coat	0	8
12	Wine	4	38
13	Sower	12	5
14	Mustard	12	5
14	Seed	16	30
15	Tenants	12	0
16	Fig	20	21
16	Tree	16	45

in the entire database, the average document size is about 13 verses. Using only one verse from each document in the second approach means that only 7.77% of the database is combined (Landauer et al. [LL90] combined up to 36.26%).

The size of the King James and Greek database is one-half the size of the database comprising the four versions of the Gospels. LSI factors $k = 80$ and $k = 113$ were initially tested with this smaller database. The higher number of LSI factors $k = 113$ returned more hits and related documents while returning less misses. For this reason subsequent tests used LSI factors ranging from 113 to 122. The two strategies for combining documents follow two lines of thought: a coarse-grained approach and a fine-grained approach. A sequential mixing of whole documents is used in the first strategy, which combines a larger *local* amount of both languages (coarse-grained). This strategy is appropriate if the documents (or verses) do not vary much in context. In other words, the spatial location of a document within the database will determine its relevancy to other documents. In contrast, if the context of documents varies within the database, this strategy would be inappropriate because only the actual documents used for combining languages are likely to be returned by an LSI-generated query.

The second strategy is closer to a random mixing of the database. The first verse of each document is combined in both languages (fine-grained), resulting in *all* documents contributing to the *common base* (i.e., more *global* mixing). Using the second strategy *all* documents have a better chance of being returned by an LSI-generated query. The first strategy allows much more of the database to be combined than the second strategy, but the second strategy is more effective in merging the languages and enabling LSI to return hits in both languages with a reasonable LSI-ranking (10-20).

Chapter 4

Performance

Tests were conducted on the following databases: (i) a verse-based version of the Gospels including four translations, (ii) a story-based version of the Gospels including four translations, and (iii) a story-based version of the Gospels including Greek and King James translations. Each series of tests gave results that indicated how LSI performs with the database, and what changes might be made before the next set of tests were run (i.e., change in the number of factors k or weighting). Different LSI models were created using various choices for k, term weightings, or indexing schemes, in order to accomplish comparisons that were appropriate for each set of tests. The first series of tests using the first version of the database examined local term weighting versus local logarithmic term weighting (see Section 1.3.1), global entropy weighting (see Equation 1.4), and various values for k (12, 13, 35, 36, 92, and 96). The second series of tests examined LSI performance with three values for k (46, 82, and 113) using local logarithmic term weighting and global entropy term weighting for the second version of the database. The third series of tests examined the performance of two schemes for combining a cross-language database of Greek and

English (the third version of the database). The third series of tests evaluated the coarse-and fine-grained methods of indexing the database. The recall and precision were monitored as well as the average LSI-rank of the hits for each LSI model. Based upon the results of the tests, the fine-grained indexing scheme was shown not only to perform the best, but also to use less disk memory space for indexing purposes.

4.1 Test by Verses

The first tests were conducted on the database of four English versions of the Gospels (see Section 3.1). The database contained 18,895 documents (i.e., verses) with each document containing an average of 6.28 terms. The top 50 LSI-rank documents for the queries in Table 2.1 were classified into one of three categories: (i) *hits* (documents listed in Table 3.1); (ii) *related* (documents not listed in Table 3.1 but related in subject matter to the query); and (iii) *misses* (documents not listed in Table 3.1 and not related to the subject matter of the query). The queries should have returned multiple hits, since each query describes a story (made up of an average of 10 verses) that appears in at least two of the Gospels (see Table 3.1). The LSI models used either term frequency-entropy weighting described in Equations (1.2) and (1.4) with $k = 12$, 35, and 92 factors, or log-entropy weighting described in Equations (1.3) and (1.4) with $k = 13$, 36, and 96 factors.

The greater the number of *hits* for a given LSI model, the better the model's *recall.* The smaller the number of *misses* for a given LSI model, the better the model's *precision* (see Section 3.1). The tests on the first group of LSI models give clear performance results for both models with varied k and different term weighting schemes.

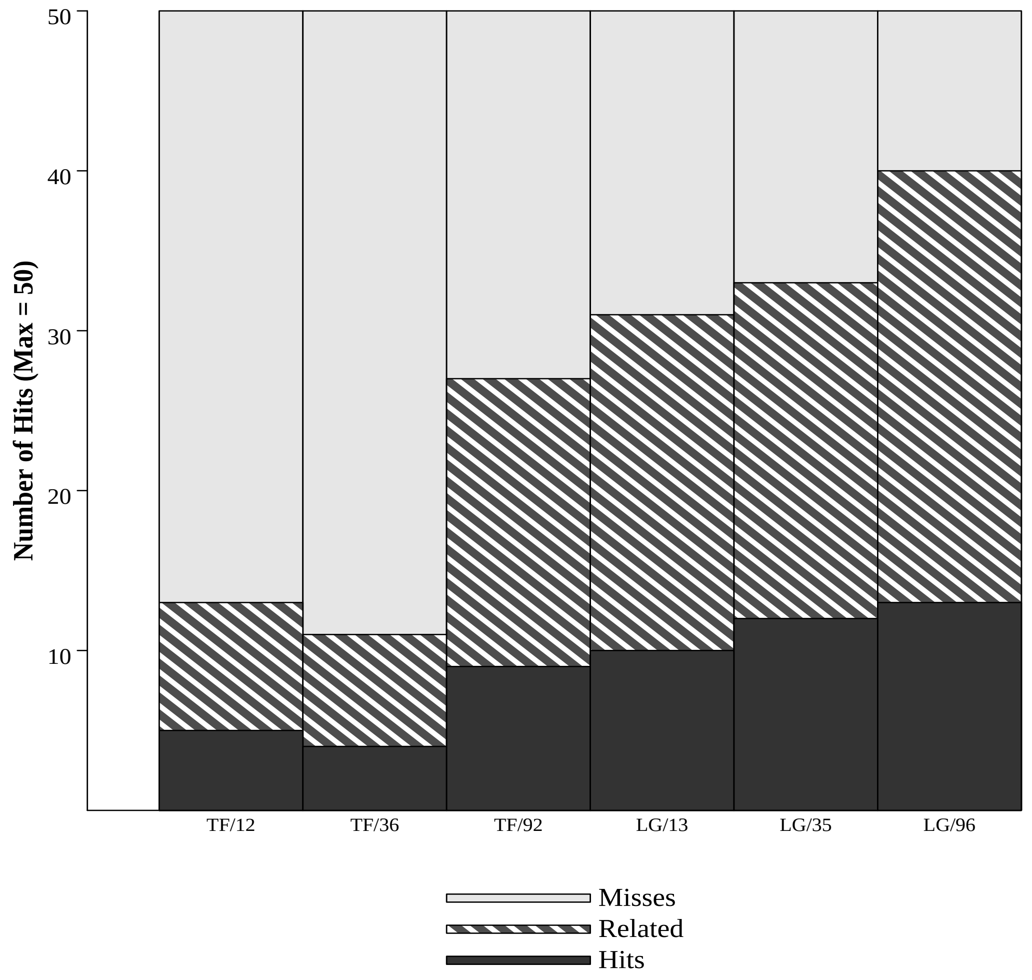

Figure 4.1: Hit distribution average across versions.

Figure 4.1 clearly indicates that log-entropy weighting performs much better than term frequency-entropy weighting. For each number of factors k, the log-entropy weighting (LG/k) returned more hits and related documents. With the smallest number of factors, log-entropy weighting returned more hits and related documents than even the best of the term frequency-entropy (TF/k) models. This confirms the earlier findings of Deerwester et al. in [DDF$^+$90] that log-entropy weighting with $k \approx 100$ factors performs quite well.

Working with verses as documents presented other problems as it is not always easy to understand the context of the verse, and therefore the verse's meaning. Verses are sentences which make up only part of a story, or some other form of literature (i.e., parable, sermon, or wisdom literature). This necessitates the grouping of verses into their larger context. The larger context for the verses is called a *story* for reasons of simplicity, even though *story* can mean any form of literature found in the Bible.

4.2 Test by Stories

The grouping of verses into stories was done according to the *New International Version Study Bible* (NIV) [BBS$^+$85]. This version (NIV) organizes the verses into the desired *story* documents and also provides titles for each story (or document). The titles were used for identification of documents returned by LSI. Only log-entropy term weighting was used for testing (local term frequency was abandoned due to poor performance). LSI models with $k = 46$, 82, and 113 factors were tested, and are referred to as model 46, model 82, and model 113, respectively.

Most of the significant documents (i.e., documents listed in Table 3.1) were returned in model 46. LSI should have returned 8, 12, or 16 hits, depending upon how many of the 4 Gospels the story appeared in. The lower number of factors for model 46 produced good *recall* at the expense of poorer *precision* (see Figure 4.2). The cosine values of the documents returned in model 46 were significantly higher than the other two models, indicating that relevance may have been somewhat forced in model 46 (see page 34). This may account for the higher number of misses found in model 46.

LSI model 82 performed well and returned most of the significant documents.

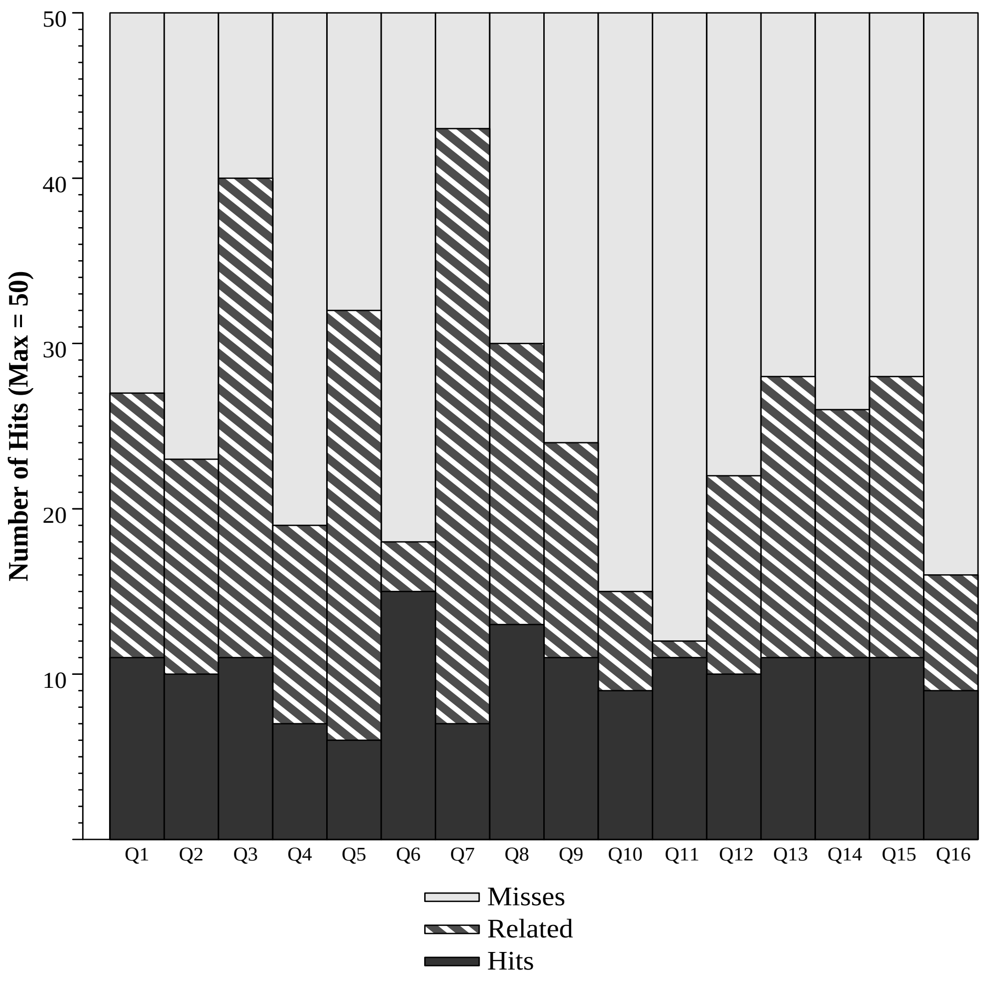

Figure 4.2: Hit distribution across queries for LSI model 46.

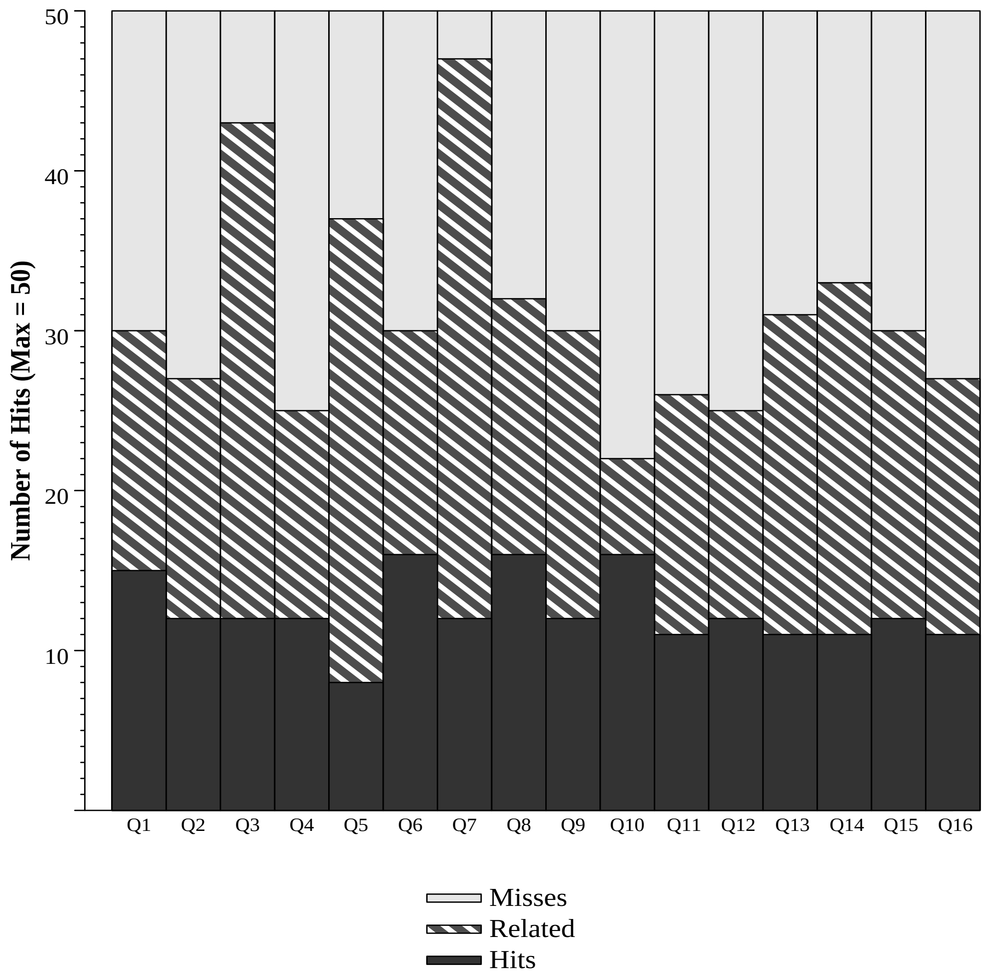

Figure 4.3: Hit distribution across queries for LSI model 82.

Unfortunately, it missed significant documents for queries 1, 11, 13, 14, and 16 (see Figure 4.3). Although fewer misses were returned by model 82 (as opposed to model 46) there were more misses than with model 113, resulting in poorer *precision.*

With $k = 113$ factors, LSI returned all expected documents (excellent recall) with a smaller number of *misses* (better precision) than the other two models (see Figure 4.4).

Figure 4.5 illustrates the increase in precision and recall of LSI model 113 over

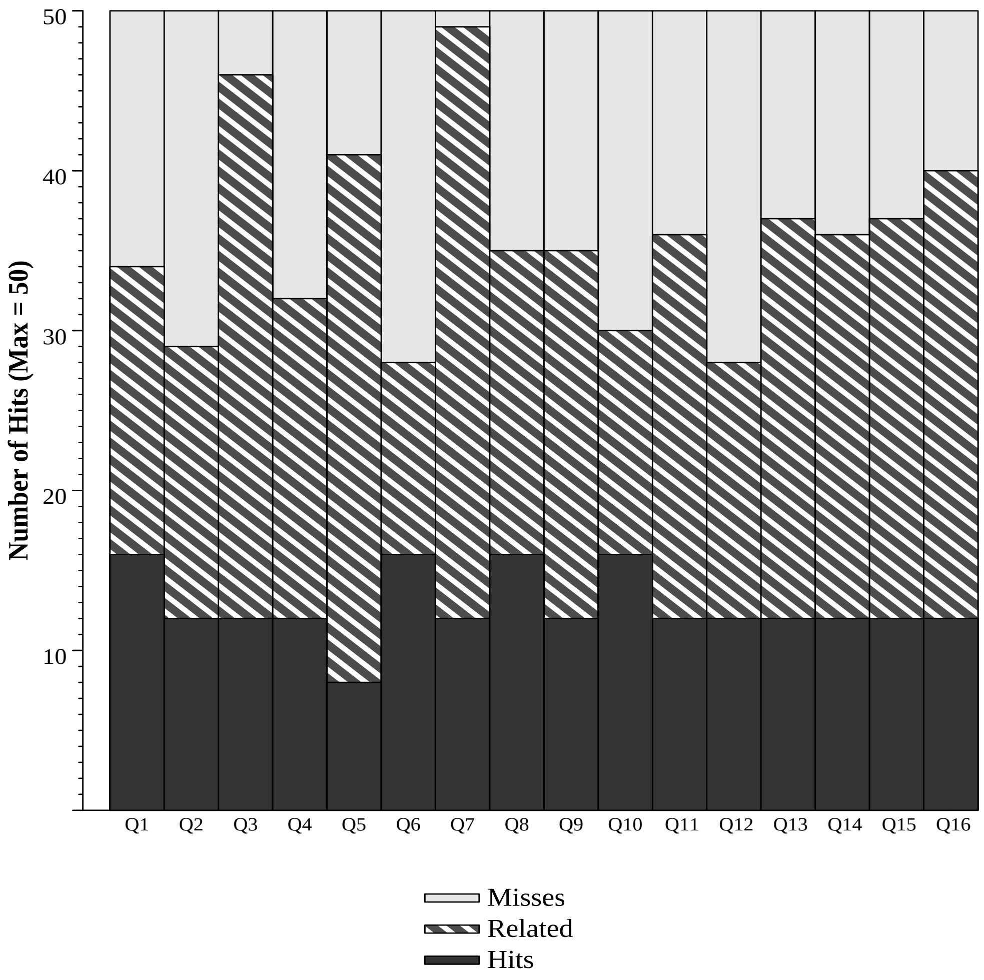

Figure 4.4: Hit distribution across queries for LSI model 113.

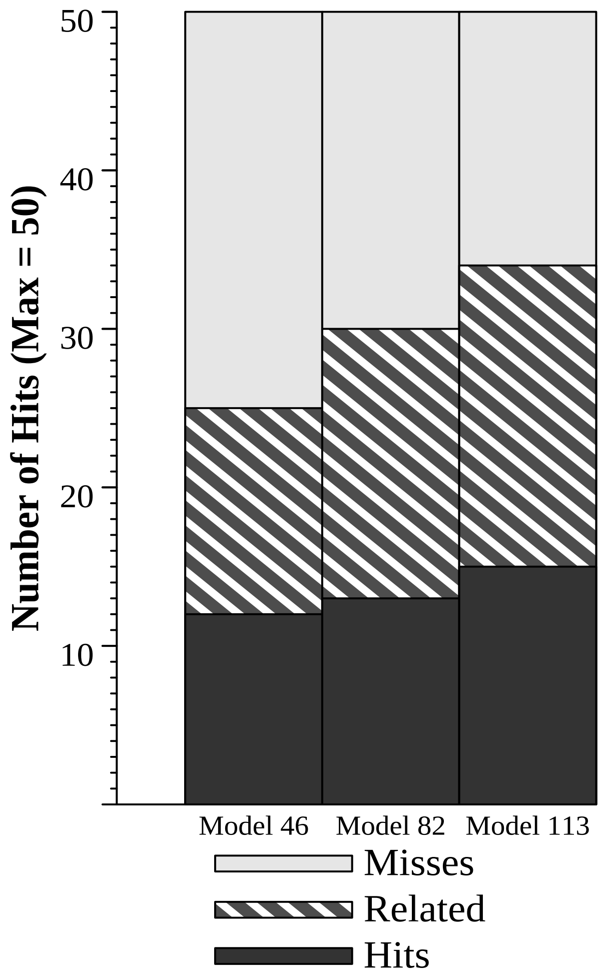

Figure 4.5: Hit, related, and miss distribution across LSI models 46, 82, and 113.

the other two models. A comparison of Figure 4.5 and Figure 4.6 demonstrates why cosine values are not a reliable indicator for relevance. The cosines were higher for the models with a smaller number of factor k at the cost of poorer precision and recall.

The models tested so far have all been variations of English. The performance of similar models for a database of both English and Greek is discussed in the next section.

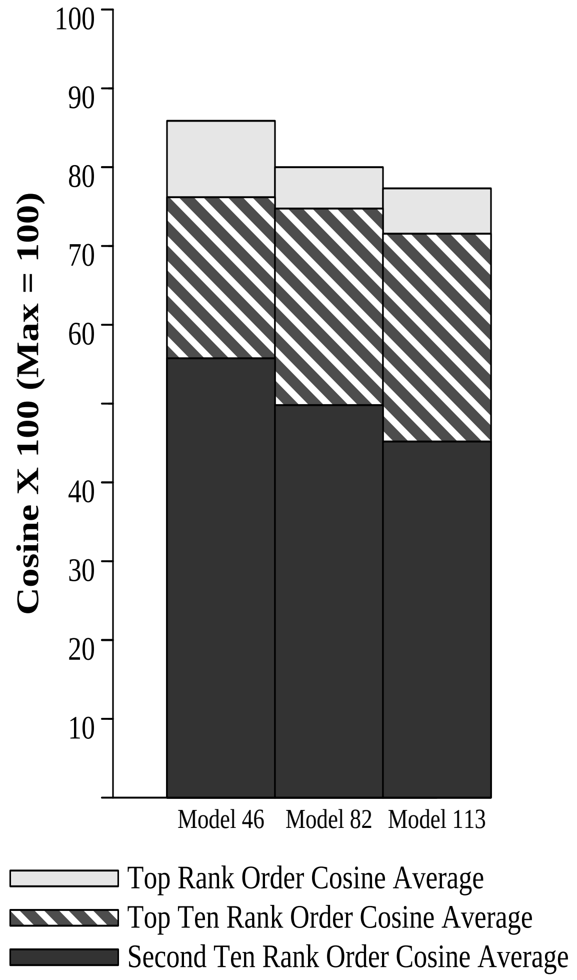

Figure 4.6: Maximum and average cosines (for highest-ranked document, first and second set of 10 highest-ranked documents) for LSI models 46, 82, and 113.

4.3 English and Greek

The LSI models used for a database comprised of both English and Greek text are listed in Table 4.1 along with the different database merge schemes discussed in Section 3.3.

The V1-115 LSI model used fine-grained indexing and all the other models used coarse-grained indexing (see Section 1.2). The two LSI models using $k = 80$ factors were studied to see if a value of $k < 100$ would perform well. Both of these models (T-80 and F1-80), however, returned most of the hits below the LSI-rank of 10 and many below 20. The average LSI-rank of the returned hits for all the LSI models in Table 4.1 are shown in Figure 4.7 for queries 6, 14, and 16. The target document (story) for queries 6, 14, and 16 is included with the coarse-grained models F8-119, F24-121, and F32-122. Hence, the average LSI-rank for hits can be compared for coarse-grained models before the inclusion of the target document for merging, and the fine-grained model V1-115 can be compared with all other models for the average LSI rank for hits.

The location of the target documents (stories) in the different Gospel accounts provides information about the performance of the coarse-grained models. Table 4.2 lists the numeric order (position) of target documents in each Gospel for queries 6, 14, and 16. For example, the target document for query number 6 occurs as the seventh document in the Gospel of John. As *target* documents were included in coarse-grained models (see Table 4.2), queries that were directly related to the *target* documents should have returned hits with an improved average LSI-rank (see Figure 4.7).

Queries 6, 14, and 16 demonstrated the effectiveness of the fine-grained method in model V1-115 (see Figure 4.7). Query 6 has the target document included at the 7th

Table 4.1: Description of LSI models in English and Greek.

Model Name	Merge Strategy	Number of Factors (k)	Granularity
T-80	Titles only	80	coarse
F1-80	First document	80	coarse
T-118	Titles only	118	coarse
F1-116	First document	116	coarse
F2-114	First 2 documents	114	coarse
F4-117	First 4 documents	117	coarse
F8-119	First 8 documents	119	coarse
F16-120	First 16 documents	120	coarse
F24-121	First 24 documents	121	coarse
F32-122	First 32 documents	122	coarse
V1-115	First verse of each document	115	fine

Table 4.2: Numeric order of target documents by query and Gospel.

Query Number	Matthew	Mark	Luke	John
6				7
14		10	22	
16		19		

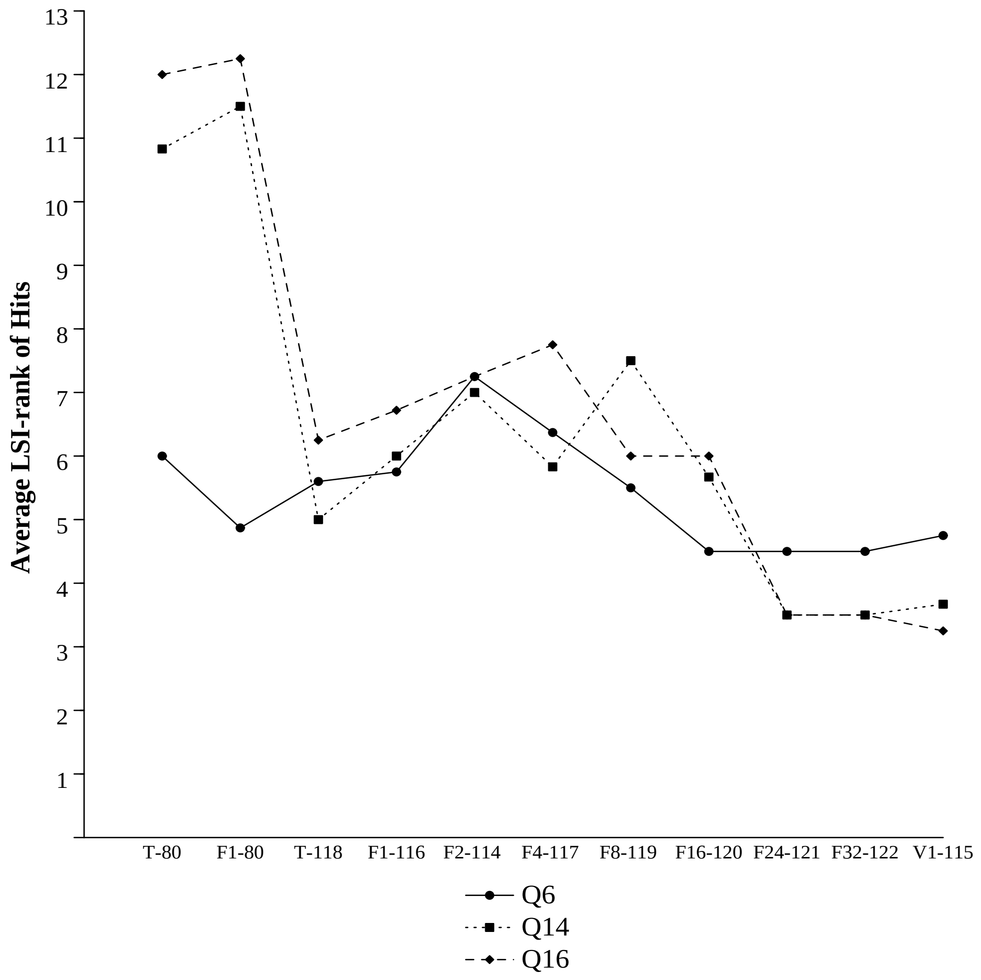

Figure 4.7: Average LSI-rank for query 6, 14, and 16 hits.

document in the Gospel of John. The inclusion of the target document in the *training sets* improves the average LSI-rank of the hits in the LSI models that merge 8 or more documents (coarse-grain). This type of improvement is also illustrated for query 14 after the 10th and 22nd document from Mark and Luke are included in the *training sets,* and for query 16 after the 19th document from Mark was included. Clearly, the performance of the coarse-grained approach to indexing was very dependent upon whether or not the target document was included in the training sets. The recall

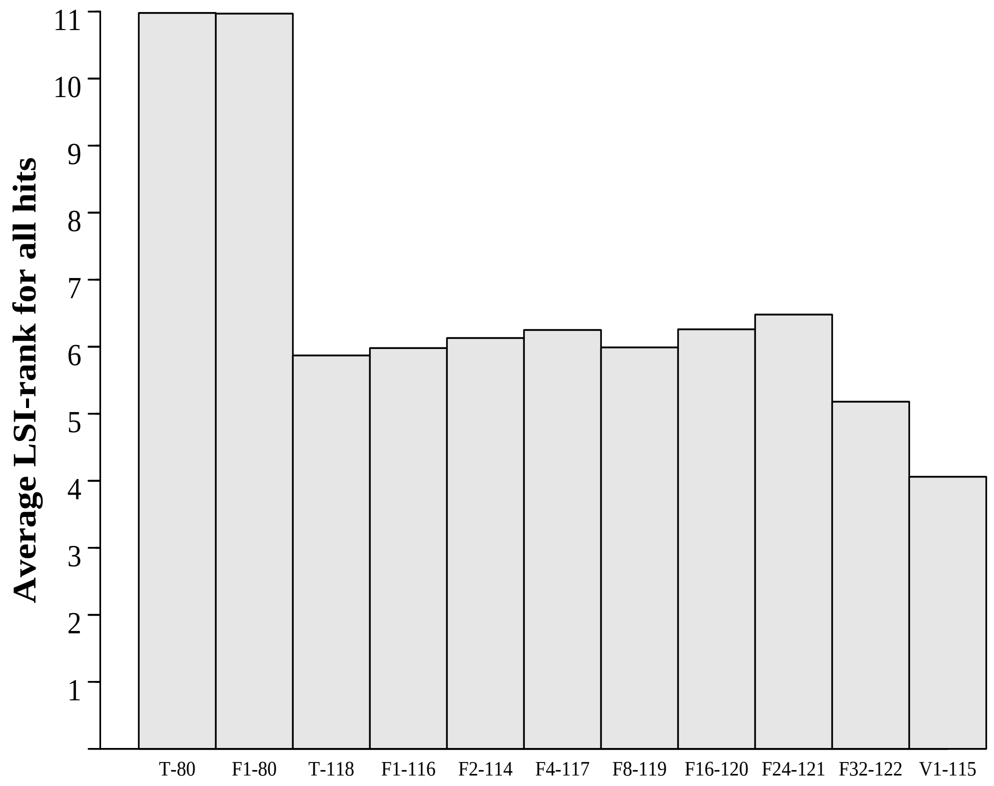

Figure 4.8: Average LSI-rank for all hits.

performance of the fine-grained approach was much more consistent across all the queries.

Figure 4.8 illustrates the average LSI-rank of the hits for each of the LSI models from Table 4.1. The T-80 and F1-80 models did not perform particularly well, returning hits that do not even average in the top 10. The other coarse-grained models fluctuate in their performance, indicating that inclusion of whole documents actually hinders the performance of queries that do not involve documents found in the *training sets.* The fine-grained V1-115 model performed better than all of the others with an average LSI-rank of 4.06 for hits.

The LSI models of particular interest are: (i) F8-119, since it combines approximately the same percentage of the database as the V1-115 model; (ii) F32-122, since it combines approximately the same percentage of the database as in Landauer et al. [LL90] and (iii) V1-115, because it is the recommended fine-grained LSI-model.

Figures 4.7 and 4.8 demonstrate that the V1-115 model returns all significant documents with a higher LSI-rank than the F8-119 model and comparable LSI-rank to the F32-122 model. The precision (number of misses) is not necessarily a helpful source of information for these models. There are a limited number of hits available (4, 6, or 8) and the number of related documents is minimal, i.e., one expects large numbers of misses since the number of documents that are either hits or related is much less than the 50 documents returned by LSI.

Another way of indicating how well LSI returns both languages is to record how many Greek documents (which are hits or related) were returned in each of the models. Figure 4.9 illustrates the number of Greek documents returned in each of the three models of interest. The V1-115 model recorded the most Greek documents (related or hits) in the top 50 returned documents. This gives a clear indication that the V1-115 model performs better than the others in incorporating the second language for information retrieval. This fine-grained method, which combines only 7% of the database, is able to achieve high levels of performance with respect to recall, LSI-rank for returned documents, and numbers of Greek documents returned in the top 50. The coarse-grained models, on the other hand, combine up to 34% of the database, and perform worse than the V1-115 LSI-model. When using a large database (several gigabytes), duplicating 34% of the database for the purpose of information retrieval is not an attractive option.

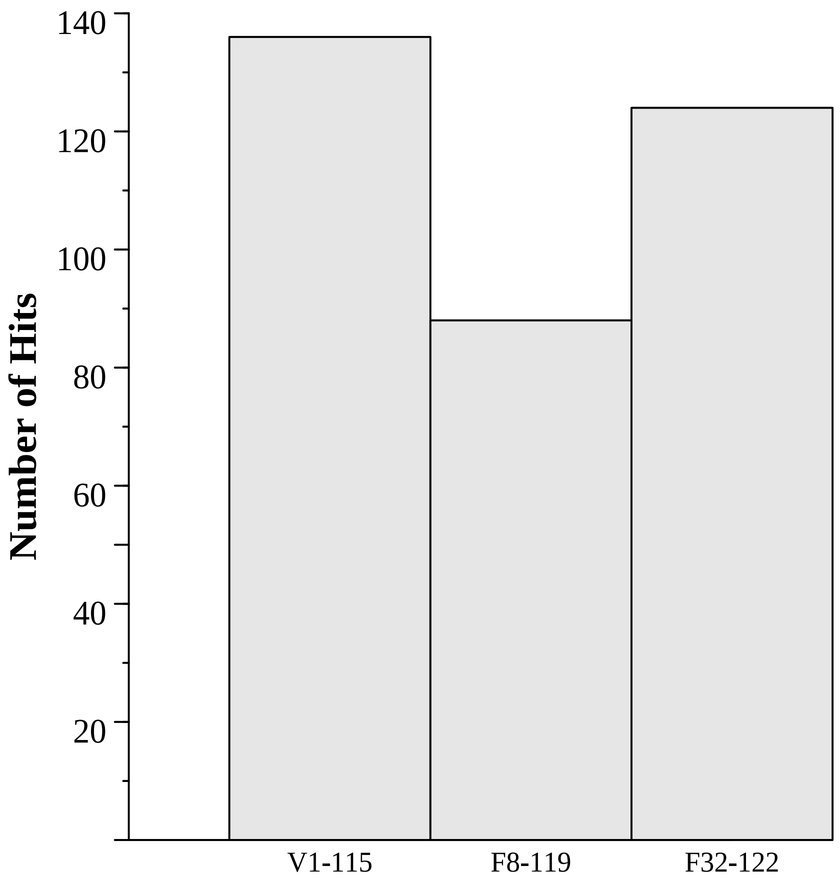

Figure 4.9: Number of Greek documents in top 50 returns.

4.4 Recommendations

Two merging methods for combining cross-language databases (coarse-grain and fine-grain) have been tested. The intent has been to develop a method for combining two languages of a database so that effective information retrieval (recall and precision) can be obtained using LSI. The coarse-grain methods used whole documents when merging the languages and duplicated from about 2% to 34% of the database. The coarse-grain LSI model that performed the best was F32-122 which duplicated more of the database than the other models. Since the coarse-grain methods include whole documents for merging, it is a hit or miss proposition. If the document you are searching for has been included in the merging of the languages, then your query will perform well. If the target document has not been included in the merging of the languages, performance suffers. This implies that for better performance on a given query, more documents should be included in merging. However, Figure 4.8 illustrates that even doubling the amount of documents for merging, the performance of coarse-grain merging models does not improve significantly, unless the target document is included in the merging (see Figure 4.7). LSI model F24-121 includes the target document in the merging and so the LSI-rank of returned *hit* documents improves significantly. If you are willing to take the chance that the target document has been included in merging, (or all documents are very similar in content) then a coarse-grain method might be an option. However, the only way to assure that the target document for your query has been included is to merge 100% of the database (effectively doubling the size of the database).

The only fine-grain method impemented was V1-115, which included under 10% of the database in the merging of the languages. This amounted to about one sentence of each document. Since each document was included in the merging, a greater variety of text from both languages was included. This allowed the languages to merge more effectively so that more Greek documents were returned with the fine-grain model than any of the coarse-grain models. This allowed the recall and precision of the fine-grain model to perform quite well across the whole scope of queries (see Figure 4.8). Even when a coarse-grain method included the target document in merging the languages the fine-grain method did well (see Figure 4.7). The fine-grain method not only had good recall and precision, but it also duplicated a smaller amount of the database. Clearly, the fine-grain method is the method of choice.

Chapter 5

Summary and Future Work

It is clear that the fine-grain method (V1-115) performs better than the coarse-grain methods, but Greek is an unusual language that assumes common words. A common word list was used for eliminating selected words in the English text. While this was not necessary for the Greek text, language-specific common word lists may be needed in other applications. Updating the database is also a consideration that was addressed by *folding-in* documents by Landauer et al. [LL90]. Since the performance of the coarse-grain LSI-models in Landauer et al.'s tests were based upon the *training sets, folding-in* later documents did not affect the performance of their LSI-models. There is no reason to believe that the performance of the fine-grain method LSI-model would be affected by *folding-in* later documents either.

Personal computers (PC's) are the most widely used computers today, and therefore it would be nice if the source code for LSI was written completely in the C language. This would enable other software developers to make use of this information retrieval system. It would also be useful to develop tools (in C) for automatically merging multilingual ASCII text files for use with LSI.

Other than searching the Bible using LSI, there are many other possible uses for a cross-language information retrieval system. For example, a multilanguage thesaurus could be implemented. A search tool for finding files stored on a computer by topic without any organizational overhead would be another useful application. LSI is certainly one plausible technique for effective retrieval from cross-language databases.

Bibliography

Bibliography

[BBS$^+$85] K. Barker, D. Burdick, J. Stek, W. Wessel, and R. Youngblood. *The New International Study Bible*. Zondervan Bible Publishers, Grand Rapids, first edition, 1985.

[BDO95] M. Berry, S. Dumais, and G. O'Brien. The computational complexity of alternative updating approaches for an SVD-encoded indexing scheme. In *Proceedings of the Seventh SIAM Conference on Parallel Processing for Scientific Computing*, Philadelphia, 1995. SIAM.

[Ber92] M. W. Berry. Large scale singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49, 1992.

[DDF$^+$90] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):1–13, 1990.

[Dum91] S. T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers*, 23(2):229–236, 1991.

[GL89] G. Golub and C. Van Loan. *Matrix Computations*. Johns-Hopkins, Baltimore, second edition, 1989.

[Hew93] S. Hewitt. Bible search programs. *Christian Computing*, 5(11):14–24, 1993.

[LL90] T. K. Landauer and M. 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, Waterloo Ontario, October 1990.

[Mir60] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Q. J. Math*, 11(1):50–59, 1960.

Vita

Paul Geoffrey Young was born in Pittsburgh, Pennsylvania on March 23, 1955. He graduated from Montour High School in 1973 and received a Bachelor of Arts degree in Religious Studies from Grove City College in 1978. He received a Masters of Divinity from Gordon-Conwell Theological Seminary in 1984, and after moving to Knoxville, Tennessee, he received his Masters in Computer Science.