Understanding LSI via the Truncated Term-term Matrix

Diplomarbeit

am Fachbereich Informatik an der Universität des Saarlandes

von
Régis Newo Kenmogne

Mai 2005

Diese Arbeit wurde nach einem Thema von Dr. Holger Bast am Max-Planck Institut für Informatik in Saarbrücken angefertigt.

Abstract

In this thesis, we study the relation between Latent Semantic Indexing (LSI) and the co-occurrence of terms in collections. LSI is a method for automatic indexing and retrieval, which is based on the vector space model and which represents the documents and computes the relevance scores in a reduced, topic-related space. For our study, we view LSI as a document expansion method, i.e. for a pair of terms, the occurrence of one of them in a document increases or decreases the importance of the other term for the document, depending on the respective entry in the expansion matrix. We study the relation between the expansion matrix and the co-occurrence information of the pairs of terms in collections. We find out that the entries of the expansion matrix are influenced by the order of co-occurrence of the pairs of terms. We then show that the retrieval performance of LSI for the optimal choice of parameters can be obtained when the expansion matrix used is a simple linear combination of the first and the second order co-occurrences.

Hiermit erkläre ich an Eides Statt, diese Diplomarbeit selbstständig angefertigt, nur die angegebenen Quellen benutzt und sie noch keinem anderen Prüfungsamt vorgelegt zu haben.

Saarbrücken, den 31. Mai 2005

Régis Newo

Acknowledgements

I would like to thank my supervisor Dr. Holger Bast for his guidance and the many helpful suggestions.
I also thank Ingmar Weber and Debapriyo Majumdar for all the hints in all the time of research for and writing of this thesis.

My special thanks go to my family for their support all the time.

Contents

List of Figures

List of Tables

Chapter

1 Introduction

Most text retrieval methods use straightforward term matching (i.e. lexical match). That is, a document is retrieved if and only if it contains one or more words occurring in the user's query. That leads to the fact that those methods cannot handle two well-known phenomenons which are related to the language usage: polysemy (e.g. surfing the web vs. surfing at a beach) and synonymy (e.g. car vs. automobile). The retrieval method which is the subject of this thesis, called Latent Semantic Indexing (LSI), tries to overcome these problems. It was first presented in [10] and [7].

LSI tries to solve these problems by finding the latent structure in documents and queries, i.e., for a given number of topics, LSI finds which words belong to each topic and also which topics each document treat of. For a given query, the relevance score assigned to each document does not depend on the words it contains, but on the topics handled in the document.
LSI is a so-called unsupervised method, that is, neither training nor explicit input of knowledge is required. It has been shown that LSI has good retrieval performance (see [10, 7]). LSI uses linear algebra techniques (e.g. singular value decomposition), as explained in [5] and also in the next chapter.

Contribution

Even though LSI works well in practice (if properly tuned), it is still not clear why LSI improves the retrieval performance [26]. Many papers address this issue and try to find some explanations (as in [17, 15, 19, 26, 8]). We also aim in this thesis to have a better understanding of the way LSI works. As shown in [2] and also detailed in Section 2.4, LSI can be viewed as a document expansion method. That is, the occurrence of a word in a document increases or decreases the importance of another word or even leads to the insertion of another word in the document, depending on the respective entry in the expansion matrix. In this thesis, we thus focus on the expansion matrix which is also called truncated term-term matrix (in Section 2.4, we will justify this appellation).
In [17, 15, 19], the authors claim that there is a strong relation between the entries of the expansion matrix and the word co-occurrence information for each pair of terms (i.e. how often two words co-occur in a document or how many terms exist which co-occur with both words of the pair and so on). We thus want to find out, to what extent those entries depend on the order of co-occurrences of the respective pairs of words. Our results show that only the first (i.e. how often two words directly co-occur) and the second (i.e. how many words directly co-occur with both words of a pair) order of co-occurrence play an important role for LSI. We take the following two approaches.

First, as already done in [17], we analyse the relation between the order of co-occurrence of pairs of words in documents and LSI (i.e. the entries of the truncated term-term matrix). We find some mistakes in the experiments made by the authors of [17, 15, 19], which lead to the fact that some of their conclusions about the relation between LSI and the co-occurrence of words in documents are wrong. We then correct them and adjust the conclusions they drew. This is the subject in Chapter 3.

In the second approach, we approximate the truncated term-term matrix with other matrices, whose entries represent the first and the second order of co-occurrence of the pairs of terms. We show that a simple combination of those two orders of co-occurrence provides a retrieval performance comparable to that of LSI. Based on these approximations, we will also see that the term co-occurrence information is at the heart of what makes LSI work. We will see in Section 2.4 that the first order co-occurrence information for a collection with m words and n documents is computed in $O(x^{2.376})$ where $x = \max(m, n)$, whereas LSI computes the optimal combination of the first and second order co-occurrence in $O(mk)$ where k is the chosen number of topics and is most of the time much smaller than m and n. LSI thus computes that information in a very efficient way, though it seems in some cases to make some computations which do not improve (or even affect) the retrieval performance. We also approximate the combination of LSI and the basic vector space model, which is well-known retrieval model that we detail in Section 2.2, and we show that it has an even better retrieval performance than the previous approximations. We will deal with this part in chapter 4.

In the next chapter, we will first present some preliminaries.

Chapter 2

Preliminaries

2.1 Information Retrieval

The amount of data stored in computers is growing every day. According to a study [3], there were about 550 billion documents on the web (including the deep web) in 2001. This data needs to be classified and retrieved (preferably fast) whenever needed. Information Retrieval is the part of computer science concerned with retrieving, indexing and structuring documents (e.g. text documents, images, videos) from collections (e.g. Web, corpora).
The retrieval method studied in this thesis is a *text retrieval* method, i.e. it retrieves documents in collections made up of text documents containing words, also called *terms*. The problem addressed here is also called the *ad-hoc retrieval* problem ([23]), i.e. the user gives a *query*, which also consists of terms, and the retrieval method tries to find the documents which fit the query best.

Although different methods have been proposed for the text retrieval problem, most of the methods are based on three main models ([1]):

- the Boolean model,
- the probabilistic model and
- the vector space model.

In the Boolean model the documents are represented as sets of words and the query consists of a Boolean combination of terms. Here, the frequency of a term in a document is not important. The main disadvantage of this model is that the retrieved documents are not ranked. That is, all returned documents are supposed to have the same relevance, which most of the time is not true and does not enable the user to concentrate on the documents that the method consider as the most relevant.

The two other models solve these problems in that the documents are ranked by relevance. There exist many methods for the ranked retrieval problem, and there are many measures in order to compare their retrieval quality. The two most important ones ([4]) are the following:

- Precision: it is a measure for the ability of a method to return only relevant documents. It is defined as
$$\text{precision} = \frac{\#\text{ relevant documents retrieved}}{\#\text{ documents retrieved}} .$$
- Recall: it is a measure for the ability of a method to return all relevant documents. It is defined as
$$\text{recall} = \frac{\#\text{ relevant documents retrieved}}{\#\text{ relevant documents in the collection}} .$$

Thus the higher the precision is, the more retrieved documents are relevant, whereas a high recall indicates that almost all returned documents are relevant.

In the probabilistic model, each document is supposed to be either relevant or not for the query, depending on which terms the document contains. The probability that a document is relevant is (approximately) computed and the documents are ranked with respect to that probability. Bayes' theorem is frequently used to compute these probabilities ([1, 9, 14]).

The retrieval method studied in this thesis is based on the vector space model which will be presented in detail in the next section.

2.2 Vector Space Model

The vector space model is widely used in Information Retrieval ([4, 23]). Here, the documents and the query are represented by vectors. Each entry

of the vector corresponds to a term. Thus for a collection with a total number m of terms the vectors are m-dimensional.
The entries in a vector represent the importance or the *weight* of the terms in the corresponding document or query. The easiest way to represent this weight is to use the term frequencies in the documents. That is, each entry in the vector indicates how often the corresponding term occurs in the document or query.

Yet most of the time there are lots of terms which occur very often in many documents. For example, in a collection containing all the articles by the members of a computer science institute the word ‘computer’ is likely to occur in many articles. Whereas a term like ‘lsi’ is rare and more likely to be specific to the document in which it occurs and should have a higher weight. This problem can be solved by considering the *global* and *local* weights of a term [5, 4]. This is why the so-called *tf-idf formula* is often used. It is based on the following two assumptions:

- the weight of a term should increase with its number of occurrences in the document and
- it should decrease with the number of documents in which it appears.

A possible weight of a term in a document proposed in [23] with the tf-idf formula is

$$d_{ij} = \begin{cases} (1 + \log tf_{ij}) \cdot \log \frac{n}{df_i} & \text{if} \quad tf_{ij} \geq 1 \\ 0 & \text{if} \quad tf_{ij} = 0 \end{cases}$$

where tf_{ij} is the frequency of the term i in the document j, n is the total number of documents and df_i the document frequency of the term i (i.e. the number of documents in which term i occurs).

Another problem is that large documents contain many terms that are very often repeated in these documents. This leads to the fact that their vectors have higher entries for the corresponding terms. A smaller document, in which the same terms might be as important as in a large one, will have lower vector entries, which can lead to differences while computing the relevance scores for a given query. This problem can be solved by normalising the document vectors. That is, each entry d_{ij} of a document d_i is divided by the length $|d_i|$ of the document.

In order to get the relevance scores, the query is compared to each document using a similarity measure which returns a value called the similarity score. The list of scores obtained is then sorted and the documents with the

highest scores are the ones the method considers to be the most relevant. The similarity measures generally return the documents which are geometrically close to the query.
Let q be the query vector and let d_i be the vector representing the ith document in the collection. The weight of the jth term is q_j in the query and d_{ij} in the document.
The most widely used similarity measures are:

- the *scalar product* (or also *dot product*), i.e.

$$sim(q, d_i) = q^T \cdot d_i = \sum_{j=1}^{m} q_j d_{ij} \ .$$

  Here, the largest similarity score is obtained when the document contains all the words of the query.

- The *cosine measure*, i.e. the score computed is the cosine of the angle between both vectors. Here we have

$$sim(q, d_i) = \frac{q^T \cdot d_i}{|q| \cdot |d_i|} = \frac{\sum_{j=1}^{m} q_j d_{ij}}{\sqrt{\sum_{j=1}^{m} q_j^2} \cdot \sqrt{\sum_{j=1}^{m} d_{ij}^2}} \ ,$$

  and the largest similarity score is also obtained when the document contains all the words of the query. This measure is equivalent to the first one when the vectors are normalised.

The *term-document matrix* A is the matrix in which each column represents a document. That is, A is defined as

$$A = [d_1, d_2, \ldots, d_n]$$

and is an $m \times n$-matrix. Usually, we have $m \gg n$, because most of the time, the documents in collections cover various topics, and each covered topic implies many new terms which are specific to the topic.
When the similarity measure used is the dot product, the relevance scores for each document with a query q can be obtained by a simple matrix product $q^T \cdot A$. The resulting n-dimensional vector holds the resulting scores for each document.

In Figure 2.1 we have an example of a simple term-document matrix A representing a collection with 5 documents. The query q just contains the term ‘web’ and with the dot product similarity measure the first and third documents are supposed to be the most relevant.

$$A = \begin{array}{lccccc} & d_1 & d_2 & d_3 & d_4 & d_5 \\ \text{internet} & 1 & 1 & 1 & 0 & 0 \\ \text{web} & 1 & 0 & 1 & 0 & 0 \\ \text{surfing} & 1 & 1 & 0 & 1 & 1 \\ \text{hawaii} & 0 & 0 & 0 & 1 & 0 \\ \text{beach} & 0 & 0 & 0 & 1 & 1 \end{array} \qquad \begin{array}{c} q^T = (0\ 1\ 0\ 0\ 0) \\ \\ q^T \cdot A = (1\ 0\ 1\ 0\ 0) \end{array}$$

Figure 2.1: Example of term-document matrix and the computation of the similarity score with the vector space model

Although the vector space model does not have the same disadvantages as the Boolean model, it still has some drawbacks. In the vector space model, the terms are assumed to be independent, as different terms are treated as different dimensions. That is, the vector space model assumes that no relation exists between the terms (see [7]). In reality this is not the case. The two main relations between terms are:

- synonymy : this means that many terms can be used to express the same thing (e.g. car and automobile). This can lead to a very low relevance score for some relevant documents just because they do not contain exactly the same words which are in the query, but instead synonyms (i.e. it has an impact on the recall).
- Polysemy : this means that a single term can be used to express many things (e.g. surfing on the web and surfing at the beach). This can lead to the fact that some irrelevant documents have high relevance scores just because they share some words (polysems) with the query (i.e. it has an impact on the precision).

The vector space model can to a certain extent deal with the polysemy problem, but not with the synonymy problem. In the example in Figure 2.1 the document d_2 becomes a very low relevance score because it does not contain the term ‘web’ (but internet), although it is also relevant for our query. The retrieval method which is the subject of this thesis is based on the vector space model and tries to overcome the problems mentioned above. It is introduced in the next section.

2.3 Latent Semantic Indexing (LSI)

As we saw in the previous section, the basic vector space model represents the documents and queries in a so-called term vector space (i.e. the entries of the vectors correspond to the terms in the collection) and it assumes that the terms are independent. Instead, LSI tries to represent the documents by taking the relation which can exist between terms into account. LSI does so by representing the query and documents by topics, instead of terms. For a given number of topics (also called concepts), LSI represents the query and the documents as vectors and each entry of the vectors correspond to a topic. The goal is to reduce the noise caused by the synonymy and the polysemy. By representing documents and queries with topics, similar words (e.g., synonyms) are assigned to the same topic, and polysems are assigned to various topics, according to their different meanings. LSI was first presented in [10, 7].

Now we will show how LSI represents the documents with topics (i.e. how the entries of the vectors, which are the weights of the topics for the document, are computed). Let k be the number of topics we wish to have (with $k < n$ and $k < m$). We want to transform the m-dimensional vectors into k-dimensional vectors. As most concept-based retrieval methods, LSI uses a matrix decomposition for this aim. So we have to find an $m \times k$ matrix T and a $k \times n$ matrix D such that the product of these two matrices is a (rank-k) approximation A' of the term-document matrix A. Each column of T corresponds to a topic and its entries are the weights of the terms for that topic, whereas each column of D is a document represented by topics and we have

$$A \approx A' = T \cdot D \ .$$

LSI uses T to express the queries by topics.
In order to compute the rank-k approximation A' and its decomposition, LSI uses a dimensionality reduction technique called *Singular Value Decomposition*. The dimensions of the approximation are chosen in a way that they represent the *axes of greatest variation* of the term-document matrix (see [23]).

2.3.1 Singular Value Decomposition (SVD)

The Singular Value Decomposition (SVD) is used to solve many problems (e.g. pseudo-inverse of matrices, data compression, noise filtering) and is a least squares method. LSI uses it to find a low rank approximation of the term-document matrix.
The SVD theorem states ([12]) that each $m \times n$ matrix can be written as a product of three matrices, i.e.

$$A = U \cdot \Sigma \cdot V^T \ ,$$

where U is an $m \times m$ matrix whose columns are the (normalised) eigenvectors of AA^T, V is an $n \times n$ matrix whose columns are the (normalised) eigenvectors of $A^T A$, and Σ is an $m \times n$ diagonal matrix which contains the sorted singular values[1] $\sigma_1, \ldots, \sigma_r$ on its diagonal. The columns of U (resp. V) are called the left (resp. right) singular vectors and $U^T U = I_m$ and $V^T V = I_n$.
The number of nonzero singular values corresponds to the rank of A. The proof of the existence and uniqueness of this decomposition can be found in [12].
The fact that Σ is not necessarily a square matrix and that A may have some singular values, which are zero, implies that many vectors in U and V will be multiplied by 0. This is why the so called *reduced SVD* will be used most of the time in this thesis. In this reduced SVD the term-document matrix is decomposed as

$$A_{m \times n} = U_{m \times r} \Sigma_{r \times r} (V_{n \times r})^T \ ,$$

where r is the rank of A, U (resp. V) is obtained by dropping the last columns of U (resp. V) from the *full* SVD and Σ is the diagonal square matrix with the nonzero singular values, i.e., $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_\mathrm{r})$.
With this decomposition, we can have a low rank approximation of each matrix, especially of the term-document matrix A. Knowing that A has rank r, we can get a rank-k (with $k < r$) approximation of A by taking the k singular vectors (and the corresponding singular values) of A from which we have the most variations. That is,

$$A \approx A_k = U_k \Sigma_k V_k^T$$

where U_k is an $m \times k$ matrix which contains the first k columns of U, V_k is an $n \times k$ matrix containing the first k columns of V and Σ_k is a $k \times k$ diagonal matrix which contains the first k singular values.
As we mentioned earlier, SVD is a least-squares method. In fact, as shown in [12], the Eckart-Young theorem states that A_k is the best rank-k approximation of A with respect to the Frobenius norm (2-norm for matrices).

Now, we have the decomposition and the low-rank approximation we need in order to map document and query vectors from the term space into the topic space. U_k is the so-called *term-topic similarity matrix*, i.e., the columns of U_k represent the topics in the term space. And V_k is the so-called *document-topic similarity matrix*, i.e., the columns of V_k^T represent the documents in

[1]The singular values are the (positive) square roots of the eigenvalues of AA^T or $A^T A$. Those eigenvalues are positive real numbers, because AA^T is symmetric and positive definite.

the topic space. Σ_k is often used to accentuate the entries of either U_k or V_k.

2.3.2 LSI in practice

The goal is to transform term vectors into topic vectors. This is done in LSI by using a linear transformation L defined by:

$$\begin{array}{rccc} L: & \mathbb{R}^m & \longrightarrow & \mathbb{R}^k \\ & x & \longmapsto & \Sigma_k^\kappa U_k^T x \end{array}$$

where $\kappa \in \mathbb{R}$ (but most of the time, we have $\kappa \in \{-1, 0, 1\}$).
For $\kappa = -1$ for example, the term-document matrix is mapped to:

$$\begin{array}{rcl} A & \longmapsto & \Sigma_k^{-1} U_k^T A \\ & = & \Sigma_k^{-1} U_k^T U \Sigma V^T \\ & = & \Sigma_k^{-1} \underbrace{(I_k \,|\, 0)}_{k \times r} \Sigma V^T \\ & = & \Sigma_k^{-1} (\Sigma_k \,|\, 0)\, V^T \\ & = & (I_k \,|\, 0)\, V^T \\ & = & V_k^T \end{array}$$

and the query q is mapped to $\Sigma_k^{-1} U_k^T q$. The relevance scores for each document are computed in the same way as in Section 2.2.
Let us for example compute the relevance scores of the documents from the example in Figure 2.1 with the query 'web'. For $k = 2$ and $\kappa = 0$ (i.e. $q \longmapsto U_2^T q$ and $A \longmapsto \Sigma_2 V_2^T$) we have with the dot product similarity measure

$$\left(U_2^T q\right) \cdot \left(\Sigma_2 V_2^T\right) = (\, 0.86 \;\; 0.53 \;\; 0.76 \;\; -0.14 \;\; -0.05 \,).$$

We see that the second document becomes a much higher relevance score (than the fourth and the fifth document), although it does not contain the term 'web'.

The best choice for k varies according to the collections and is a difficult task. While k should be small enough to remove much of the noise, it should also be large enough to cover all the topics treated in the collection [5]. This problem is the subject of an ongoing research [22].

2.4 Term Similarities

As we saw earlier, LSI assigns terms to topics and topics to documents. This assignment of terms to topics depends on the *similarity* between terms, i.e. how often two terms co-occur.
The term-document matrix A for a collection with n documents and m terms is an $m \times n$ matrix with each column of the matrix representing a document. A can also be viewed as a matrix with each row representing a term vector, i.e. a vector containing the weight of a term in each document. Thus, similarities between terms (and also between documents) can be computed [7]. A value representing the similarity t_{ij} between two terms i and j (with $i, j \in \{1, \ldots, m\}$) is the dot product of the i-th (say a_i) and the j-th row (say a_j) of the term-document matrix, i.e. $t_{ij} = a_i \cdot a_j$. t_{ij} is nonzero if and only if a document exists, in which both terms i and j occur.
Let T be the square matrix containing all those similarities. T is called the *term-term co-occurrence matrix* and is defined as:

$$\begin{aligned} T &= AA^T \\ &= U\Sigma V^T (U\Sigma V^T)^T \\ &= U\Sigma \underbrace{V^T V}_{=I_r} \Sigma U^T \\ &= U\Sigma^2 U^T \qquad \big(= (U\Sigma)(U\Sigma)^T \,\big) \end{aligned}$$

T can be computed in $O(x^{2.376})$ where $x = \max(m, n)$ with the fast matrix multiplication algorithm by Don Coppersmith and S. Winograd.
Like the terms, documents can also be compared to each other by computing the dot product of two columns. However, that won't be required in this thesis.

We noticed in the last section that LSI first transforms m-dimensional vectors (documents and queries) into k-dimensional vectors before computing the relevance scores. Let q' be the transformed query q and A' the transformed term-document matrix A. We have:

$$q' = \Sigma_k^\kappa U_k^T q \qquad \text{and} \qquad A' = \Sigma_k^\kappa U_k^T A\ .$$

Let us see what happens when the relevance scores for the documents are computed:

First, we have with the dot product:

$$\begin{aligned} q'^T A' &= (\Sigma_k^\kappa U_k^T q)^T \Sigma_k^\kappa U_k^T A \\ &= q^T \underbrace{U_k \Sigma_k^{2\kappa} U_k^T}_{=:T_k} A \\ &= q^T T_k A \qquad \big(= (T_k q)^T A = q^T (T_k A) \big) \end{aligned} \tag{2.1}$$

As we can see in equation 2.1, T_k is a *term expansion* matrix for the queries as well as for the documents. We also see that computing the relevance scores in the m-dimensional space in combination with that term expansion matrix is equivalent to first transforming the m-dimensional vectors into k-dimensional ones and then computing the relevance scores in the k-dimensional space.
Second, when the cosine similarity measure is used, the relevance score for each document is

$$\begin{aligned} \frac{q'^T d_i'}{|q'| \cdot |d_i'|} &= \frac{(\Sigma_k^\kappa U_k^T q)^T (\Sigma_k^\kappa U_k^T d_i)}{|\Sigma_k^\kappa U_k^T q| \cdot |\Sigma_k^\kappa U_k^T d_i|} \\ &= \frac{q^T U_k \Sigma_k^{2\kappa} U_k^T A}{|\Sigma_k^\kappa U_k^T q| \cdot |\Sigma_k^\kappa U_k^T d_i|} \\ &= \frac{q^T T_k A}{|\Sigma_k^\kappa U_k^T q| \cdot |\Sigma_k^\kappa U_k^T d_i|} \,. \end{aligned} \tag{2.2}$$

When we take a look at the denominator of the right part of equation 2.2, we can remark that $|\Sigma_k^\kappa U_k^T q|$ is the same for each document and thus does not influence the ranking for each query.
Furthermore, we have:

$$\begin{aligned} |\Sigma_k^\kappa U_k^T d_i|^2 &= (\Sigma_k^\kappa U_k^T d_i)^T (\Sigma_k^\kappa U_k^T d_i) \\ &= d_i^T U_k \Sigma_k^\kappa \Sigma_k^\kappa U_k^T d_i \\ &= d_i^T U_k \Sigma_k^\kappa \underbrace{U_k^T U_k}_{=I_k} \Sigma_k^\kappa U_k^T d_i \\ &= d_i^T T_k' T_k' d_i \qquad \text{with} \quad T_k' = U_k \Sigma_k^\kappa U_k^T \quad \text{and} \quad T_k'^2 = T_k \\ &= |T_k' d_i|^2 \end{aligned} \tag{2.3}$$

From equation 2.3, we gather that $|\Sigma_k^\kappa U_k^T d_i| = |T_k' d_i|$. The ranking computed by LSI with the cosine similarity measure is the same as the one computed with

$$q \cdot \frac{T_k d_i}{|T_k' d_i|} \qquad \text{for} \quad i \in \{1, \ldots, n\}. \tag{2.4}$$

For $\kappa = 0$, which is the most widely used variant of LSI, we have $T_k = T_k' = U_k U_k^T$. That means, as shown in [2], that the ranking computed with the

$$T_2 = \begin{array}{llllll} & \text{internet} & \text{web} & \text{surfing} & \text{hawaii} & \text{beach} \\ \text{internet} & 0.55 & 0.42 & 0.20 & -0.09 & -0.14 \\ \text{web} & 0.42 & 0.34 & 0.10 & -0.09 & -0.15 \\ \text{surfing} & 0.20 & 0.10 & 0.58 & 0.21 & 0.38 \\ \text{hawaii} & -0.09 & -0.09 & 0.21 & 0.13 & 0.23 \\ \text{beach} & -0.14 & -0.15 & 0.38 & 0.23 & 0.40 \end{array}$$

Figure 2.2: Truncated term-term matrix ($k = 2$ and $\kappa = 0$) of the term-document matrix defined in Figure 2.1

cosine similarity measure can be obtained with

$$q \cdot \frac{T_k d_i}{|T_k d_i|} \qquad \text{for} \quad i \in \{1, \ldots, n\}, \tag{2.5}$$

which shows that the similarities in this case can also be computed in the m-dimensional space in combination with a term expansion matrix (but only for documents).

We see from the equations 2.4 and 2.5 that when the cosine similarity measure is used, T_k is used as a document expansion matrix. T_k is also called *truncated term-term matrix*, because it is computed using the truncated term-topic similarity U_k. Each entry in T_k is a measure for the similarity of two terms in the concept space. When a vector is multiplied by that matrix, the presence of a term in that vector effects the increase or decrease of the weight of other terms in the vector, depending on the respective entry in T_k. In figure 2.2, we see the truncated term-term matrix of the term-document matrix define in Figure 2.1 for $k = 2$ and $\kappa = 0$. We remark from the matrix that terms which belong to the same topic (e.g. internet and web) receive (high) positive similarity value, whereas terms from different topics (e.g. web and beach) have lower (even negative) similarity values.

Now we want to know how LSI works, i.e. how LSI detects the relation which exists between two terms (and assigns the appropriate value in T_k). Understanding the truncated term-term matrix and also the meaning of its entries would help to understand how LSI works. We will see in the next chapter that the term co-occurrence information is at the heart of what makes LSI work.

Chapter

3 Using Term Co-occurrences

In the last chapter, we saw that LSI is a retrieval method which tries to solve the problems caused by synonymy and polysemy. In this chapter, we will see that LSI does so by using the term co-occurrence information. That is, how often or in how many documents terms occur together with other terms and so on.
For a better understanding of this, we will use a graph representation of the term-term matrix.

3.1 Representing the Term-term Matrix as a Graph

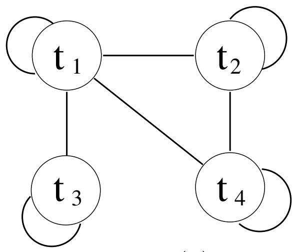
We also saw in the last chapter how two terms can be compared and how the term-term co-occurrence matrix can be obtained. To represent an $m \times m$ term-term matrix T as a graph, we need m nodes, where each node represents a term. There exists an edge between two nodes i and j if and only if the entry t_{ij} of T is nonzero. Knowing that an entry t_{ij} in the term-term matrix is non-zero if and only if there exists a document in which both terms i and j occur, we can then easily see from the graph whether two terms co-occur or not.
The edges of the graph can also be weighted by the value of t_{ij}.

	d_1	d_2	d_3
t_1	1	1	0
t_2	1	0	1
t_3	0	1	0
t_4	1	0	1

(a)

	t_1	t_2	t_3	t_4
t_1	2	1	1	1
t_2	1	2	0	2
t_3	1	0	1	0
t_4	1	2	0	1

(b)



(c)

Figure 3.1: Example of a term-term graph. (a) is the term-document matrix, (b) is the term-term matrix coming from (a) and (c) if is then the term-term graph coming from (b).

This graph is called the *term-term graph*. An example of a term-term graph can be seen in figure 3.1.

For LSI, we are not only interested in terms which directly occur together in documents, but also in the degree of term transitivity. That is, for two terms i and j which may not co-occur, we also want to know if another term k exists such that i and k co-occur and k and j co-occur and so on. In this thesis, the degree of transitivity will also be called order of co-occurrence, which can be defined using the term-term graph.

Definition 3.1 (Order of co-occurrence). *The order co-occurrence of a pair of terms i and j is the number of edges of the shortest path (in the unweighted term-term graph) between node i and j in the term-term graph.*

In figure 3.1 for example, the order of co-occurrence of t_2 and t_3 is 2, because the shortest path between these nodes has 2 edges in the term-term graph. In fact, we can see that t_2 co-occurs with t_3 in document d_1, and t_1 co-occurs with t_3 in d_2 there is no document in which both terms occur. It justifies the order of co-occurrence of that pair of terms.
The following definition is helpful in order to avoid confusions.

Definition 3.2 (nth degree path). *An nth degree path between a pair of terms i and j is a path with exactly n edges between i and j.*

In figure 3.1 again, we can see that there is a third degree path between t_2 and t_3 through t_4 and t_1.

3.2 Some Properties of the (Truncated) Term-term Co-occurrence Matrix

In this section, some mathematical background about the term-term co-occurrence matrix will be presented.
We already know that $T = AA^T = U\Sigma^2 U^T$. With this formula, it is then easy, using inductive proof and the fact that $U^T U = I_r$, to show that for each natural number $n \geq 1$, we have

$$T^n = U\Sigma^{2n}U^T.$$

Due to the fact that Σ_k is a diagonal matrix (i.e. $\Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$), we have for any real number x, $\Sigma_k^x = \mathrm{diag}(\sigma_1^x, \ldots, \sigma_k^x)$. Thus, the entry t_{ij} of T is

$$t_{ij} = \sum_{z=1}^{r} u_{iz}u_{jz}\sigma_z^2,$$

and the entry $t_{ij}^{(n)}$ of T^n is

$$t_{ij}^{(n)} = \sum_{z=1}^{r} u_{iz}u_{jz}\sigma_z^{2n}.$$

For each $n \geq 1$, we also have $t_{ij}^{(n)} = t_{ji}^{(n)}$.

The following lemma shows why the powers of the term-term matrix are important.

Lemma 3.1. *Let i and j be two terms, the ijth entry of T^n is nonzero if and only if there exists a path from i to j in the term-term graph with at most n edges.*

Proof. This lemma can be proven by induction over n.
For $n = 1$, it is clear, because of the definition of the term-term graph, that $t_{ij} \neq 0$ is equivalent to the fact that there is an edge between i and j in the term-term graph.
Now, supposing that the assumption is true for n, let us prove it for $n + 1$.
Since $T^{n+1} = T^n T$, we have $t_{ij}^{(n+1)} = \sum_{x=1}^{m} t_{ix}^{(n)} t_{xj}$.

So,

$$\begin{aligned} t_{ij}^{(n+1)} \neq 0 &\iff \exists\, x \in \{1,\ldots,m\} \quad \text{with} \quad t_{ix}^{(n)} t_{xj} \neq 0 \\ &\iff \exists\, x \text{ with } \quad t_{ix}^{(n)} \neq 0 \quad \text{and} \quad t_{xj} \neq 0 \\ &\iff \exists\, x \text{ such that there is a path between } i \text{ and } x \text{ with at most} \\ &\qquad n \text{ edges and there is an edge between } x \text{ and } j \\ &\iff \text{there is a path with at most } n+1 \text{ edges between } i \text{ and } j. \end{aligned}$$

□

Definition 3.3 (Weight of a path). *The weight of a path between i and j in the weighted term-term graph is the product of the weights of the edges in the path.*

In fact, $t_{ij}^{(n)}$ is the sum of the weights of all possible nth degree paths between i and j.

There are two remarks that should be made here. First, the considered paths may contain cycles. Second, $t_{ii} \neq 0$ for each term i in the collection. Thus, there exists an edge from i to i in the term-term graph for each $i \in \{1,\ldots,m\}$. Both remarks lead to the fact that $t_{ij}^{(n)}$ can be viewed as a linear combination of the weights of the first, second, …, nth order co-occurrence paths between i and j. That is the statement made in Lemma 3.1.

An illustration can be seen in the following calculations. The ij-th entry of T^2 when $i \neq j$ for example is:

$$\begin{aligned} t_{ij}^{(2)} &= \sum_{x=1}^{m} t_{ix} t_{xj} \\ &= \sum_{x \neq i,j} t_{ix} t_{xj} + \underbrace{t_{ii} t_{ij}}_{x=i} + \underbrace{t_{ij} t_{jj}}_{x=j} \\ &= \sum_{x \neq i,j} t_{ix} t_{xj} + t_{ij} \cdot (t_{ii} + t_{jj}) \end{aligned} \tag{3.1}$$

The first summand of equation 3.1 represents the weight of all second degree paths between i and j, and the second summand represents the weight coming from the first degree paths. Thus, we can see that $t_{ij}^{(2)}$ depends on the first and second degree paths.

We also have:

$$\begin{aligned} t_{ii}^{(2)} &= \sum_{x=1}^{m} t_{ix} t_{xi} \\ &= \sum_{x \neq i} t_{ix}^2 + \underbrace{t_{ii}^2}_{x=i} \end{aligned} \tag{3.2}$$

That fact can also be seen when we consider the calculations of the ijth entry of T^3 for $i \neq j$.

$$
\begin{aligned}
t_{ij}^{(3)} &= \sum_{x=1}^{m} t_{ix}^{(2)} t_{xj} \\
&= \sum_{x \neq i,j} \left(\sum_{y \neq i,x} t_{iy} t_{yx} \ + \ t_{ix} \cdot (t_{ii} + t_{xx}) \right) t_{xj} \quad \text{because of equation 3.1} \\
&\quad + \underbrace{\left(\sum_{y \neq i} t_{iy}^2 \ + \ t_{ii}^2 \right) t_{ij}}_{x=i} \qquad \text{according to equation 3.2} \\
&\quad + \underbrace{\left(\sum_{y \neq i,j} t_{iy} t_{yj} \ + \ t_{ij} \cdot (t_{ii} + t_{jj}) \right) t_{jj}}_{x=j} \\
&= \sum_{x \neq i,j} \left(\sum_{y \neq i,x} t_{iy} t_{yx} \right) t_{xj} \ + \ \sum_{x \neq i,j} t_{ix} t_{xj} (t_{ii} + t_{xx}) \\
&\quad + \sum_{y \neq i} t_{iy}^2 t_{ij} \ + \ t_{ii}^2 t_{ij} \\
&\quad + \left(\sum_{y \neq i,j} t_{iy} t_{yj} \right) t_{jj} \ + \ (t_{ii} + t_{jj}) t_{ij} t_{jj} \\
&= \sum_{x \neq i,j} \left(\sum_{y \neq i,j,x} t_{iy} t_{yx} \right) t_{xj} \ + \ \sum_{x \neq i,j} t_{ij} t_{jx} t_{xj} \\
&\quad + \sum_{x \neq i,j} t_{ix} t_{xj} (t_{ii} + t_{xx} + t_{jj}) \\
&\quad + \left(\sum_{y=1}^{m} t_{iy}^2 \ + \ t_{ii}^2 \ + \ t_{jj}^2 \ + \ t_{ii} t_{jj} \right) t_{ij} \\
&= \sum_{x \neq i,j} \left(\sum_{y \neq i,j,x} t_{iy} t_{yx} \right) t_{xj} \\
&\quad + \sum_{x \neq i,j} t_{ix} t_{xj} (t_{ii} + t_{xx} + t_{jj}) \\
&\quad + \left(\sum_{y=1}^{m} t_{iy}^2 \ + \sum_{x \neq i}^{m} t_{jx}^2 \ + \ t_{ii} t_{jj} \right) t_{ij}
\end{aligned} \tag{3.3}
$$

Again, we can see from equation 3.3 that $t_{ij}^{(3)}$ can be seen as a linear combination of the weight of all possible first, second and the third degree paths between i and j.

As we saw in section 2.4, the truncated term-term matrix plays an important role in LSI, and we have $T_k = U_k \Sigma_k^{2\kappa} U_k^T$. The ij-th entry $\tilde{t}_{ij}$ of T_k is

$$\tilde{t}_{ij} = \sum_{z=1}^{k} u_{iz} u_{jz} \sigma_z^{2\kappa}$$

T_k is also a symmetric matrix.
Just as T contains the term-term similarities in the r-dimensional space, T_k contains these in the k-dimensional space (i.e. the topic space). Each row (or column) of T_k represents the similarities of a given term with all other terms. By multiplying T_k with a query or a document vector, each entry of the resulting vector is the dot product of a row of T_k and the query or document vector. This means that the entries $\tilde{t}_{ij}$ of T_k intuitively have the following effects:

- When $\tilde{t}_{ij} > 0$, the weight of term j will be amplified, whenever term i appears in the query or document vector and vice versa.
- When $\tilde{t}_{ij} < 0$, the weight of term j will be decreased, whenever term i appears in the vector and vice versa.
- $\tilde{t}_{ij} \approx 0$ means that there is no significant relation between both terms.

The two following theorems are helpful for a better understanding of the truncated term-term matrix.

Theorem 3.1. *Let r be the rank of the term-document A. If the singular values $(\sigma_i)_{i=1,\dots,r}$ of A are pairwise different (which is almost always the case for 'real' collections), then the truncated term-term matrix can be written as a linear combination of the powers of T. That is,*

$$T_k = \sum_{l=1}^{r} \alpha_l T^l$$

where the α_i are real numbers just depending on $\sigma_1, \dots, \sigma_r$.

Proof. We know that the ij-th entry of T^l is

$$t_{ij}^{(l)} = \sum_{z=1}^{r} u_{iz} u_{jz} \sigma_z^{2n}$$

Let i and j be arbitrary, but fixed and $t^{(l)} = t_{ij}^{(l)}$. We have:

$$
\begin{aligned}
& & t^{(l)} = \textstyle\sum_{z=1}^{r} w_z \sigma_z^{2l} \qquad & \text{with } w_z = u_{iz}u_{jz} \\
\Longrightarrow & & t^{(l)} = \begin{pmatrix} \sigma_1^{2l} \\ \vdots \\ \sigma_r^{2l} \end{pmatrix}^T \begin{pmatrix} w_1 \\ \vdots \\ w_r \end{pmatrix} & \\
\Longrightarrow & & \begin{pmatrix} t^{(1)} \\ \vdots \\ t^{(r)} \end{pmatrix} = \underbrace{\begin{pmatrix} \sigma_1^2 & \cdots & \sigma_r^2 \\ \vdots & \ddots & \vdots \\ \sigma_1^{2r} & \cdots & \sigma_r^{2r} \end{pmatrix}}_{=:M} \begin{pmatrix} w_1 \\ \vdots \\ w_r \end{pmatrix} & \\
\Longrightarrow & & \begin{pmatrix} w_1 \\ \vdots \\ w_r \end{pmatrix} = M^{-1} \begin{pmatrix} t^{(1)} \\ \vdots \\ t^{(r)} \end{pmatrix} &
\end{aligned} \tag{3.4}
$$

We observe that M is a so-called generalised Vandermonde matrix and is thus invertible, as shown in [11], because the singular values are pairwise different. Now, let $\tilde{T} = T_k$ and $\tilde{t} = \tilde{T}_{ij}$.
We have:

$$\tilde{t} = \sum_{z=1}^{k} u_{xz}u_{yz}\sigma_z^{2\kappa} = \sum_{z=1}^{r} w_z \sigma_z^{2\kappa}$$

With the equation 3.4, we also have

$$w_z = \sum_{l=1}^{r} M_{zl}^{-1} t^{(l)}$$

Thus, we conclude that

$$
\begin{aligned}
\tilde{t} &= \sum_{z=1}^{k} \left(\sum_{l=1}^{r} M_{zl}^{-1} t^{(l)} \right) \sigma_z^{2\kappa} \\
&= \sum_{l=1}^{r} \underbrace{\left(\sum_{z=1}^{k} M_{zl}^{-1} \sigma_z^{2\kappa} \right)}_{=:\alpha_l} t^{(l)} \\
&= \sum_{l=1}^{r} \alpha_l t^{(l)} \qquad \text{for arbitrary } i \text{ and } j.
\end{aligned}
$$

And we can see that α_l just depends on the singular values $\sigma_1, \ldots, \sigma_r$. $\square$

Theorem 3.2. *If the ij-th entry of the truncated term-term matrix T_k is nonzero, then there exists a path from i to j in the term-term graph.*

Proof. In [17], the authors provide a mathematical proof which uses the formula for the entries of T_k. Yet this theorem can also be easily proven by using Lemma 3.1 and Theorem 3.1:
Knowing that T_k is a linear combination of the powers of T, an entry in T_k is nonzero if and only if there exists a power of T in which that entry is nonzero. Thus, there exists a path from i to j in the term-term graph whose length, according to Lemma 3.1, is even at most r. □

3.3 Related Work

To our knowledge, [17] is the first work that studied the entries of the truncated term-term matrix. The authors also presented the results of their experiments in [15, 19]. This diploma thesis was inspired by these works.

In the articles, the authors first conducted some experiments with a few collections to find out the number of pairs of terms in each order of co-occurrence. They remarked that for most collections, the maximum co-occurrence order was 3.
Furthermore, they tried to find a relation between the order of co-occurrence of a pair of terms and the value of its entry in the truncated term-term matrix. They found out that for most collections, the absolute value of the T_k-entries of the third order co-occurrence pairs was very low compared to that of the first and second co-occurrence pairs. That means, that such pairs do not have much impact on the retrieval performance of LSI. The authors thus concentrated on the second degree paths (i.e. first and second order co-occurrence pairs) in the term-term graph.
For given intervals for the T_k-values, they computed the number of first and second order pairs, the total number of second degree paths and the average number of second degree paths coming from the first (resp. second) order co-occurrence pairs. An example for a collection can be seen in table 3.1. They then arrived at the following conclusions:

- First order co-occurrence pairs
  - with higher number of second degree paths tend to have negative T_k-values,
  - with few paths tend to have low T_k-values,
  - with a moderate number of paths tend to receive high T_k-values.

Term Term Mat Value	Order 1 Pairs	Order 2 Pairs	Length 2 Paths	Av. No Paths Order 1 pairs	Av. No Paths Order 2 pairs
less than -0.2	68	632	323,952	4,764	513
-0.2 to -0.1	1,026	13,980	5,388,156	5,252	385
-0.1 to 0.0	52,734	7,416,342	598,493,140	11,349	81
0.0 to 0.1	1,256,054	11,292,268	1,500,874,348	1,195	133
0.1 to 0.2	607,618	89,546	298,811,456	492	3,337
0.2 to 0.3	229,274	4,166	135,964,358	593	32,637
0.3 to 0.4	107,380	366	76,171,768	709	208,120
0.4 to 0.5	57,808	46	47,004,230	813	1,021,831
0.5 to 0.6	34,208	8	31,258,040	914	3,907,255
0.6 to 0.7	21,790	2	21,734,644	997	10,867,322
0.7 to 0.8	15,040	-	15,991,638	1,063	-
0.8 to 0.9	10,180	-	11,607,916	1,140	-
0.9 to 1.0	7,466	-	9,000,714	1,206	-
1.0 to 2.0	23,352	-	32,179,792	1,378	-
2.0 to 3.0	3,374	-	5,595,238	1,658	-
3.0 to 4.0	734	-	1,293,838	1,763	-
4.0 to 5.0	250	-	480,136	1,921	-
5.0 to 6.0	82	-	156,506	1,909	-
6.0 to 7.0	48	-	90,186	1,879	-
7.0 to 8.0	20	-	33,366	1,668	-
over 8.0	14	-	33,386	2,385	-

Table 3.1: Average number of paths by T_k value for CRAN, $k = 100$ (from [17])

- Second order co-occurrence pairs
  - with a higher number of second degree paths tend to receive high T_k-values,
  - with a smaller number of paths tend to have low T_k-values.

The authors also proposed in [15] a simple function based on the conclusions for the computation of an approximation of T_k.

When we take a closer look at table 3.1, we notice that they just divide the total number of second degree paths within an interval by the number of first (resp. second) order pairs in order to have the average number of second degree paths coming from the first (resp. second) order co-occurrence pairs within the interval. That means that they did not take into account the fact that the total number of second degree paths contains first as well as second

order co-occurrence pairs. The averages computed are thus incorrect.
Another mistake they made is that they defined LSI with $\kappa = 0$ (as can be read in [19]), but experimented with the truncated term-term matrix for $\kappa = 1$.
These observations lead to the fact that some of their conclusions are wrong.

3.4 Improvements and Experiments

In the previous section, we showed how we think the experiments in [17, 15, 19] should have been done.
We wrote a Java program, in order to reconstruct (in the right way) what the authors observed. That is, the goal was to calculate the average number of second degree paths coming from pairs of terms with first (resp. second) order of co-occurrence in given collections and in given intervals of the truncated term-term value of those collections.

We conducted our experiments with four collections:

- *mpi-abstracts*, a collection of 676 abstracts of publications at the Max-Planck Institut in Saarbrücken with 3,283 terms,
- *MED*, a collection of medical abstracts containing 1,033 documents and 4,250 terms,
- *CRAN*, a collection with 1,379 documents and 4,410 terms and
- *CISI*, a collection with 1,460 documents and 5,753 terms.

The text-files of the collections are transformed into matrices in the Harwell-Boeing format using a program called `text2matrix`[1]. The Harwell-Boeing format was used because the term-document matrices are very sparse.

3.4.1 Java program

In order to complete the experiments, we needed a library which we could use for the computation of the singular value decomposition of a matrix and the computation of a the shortest path in graphs in Java. We used the *Colt library* [6] for the SVD computation and the *Data Structures Library in Java* (JDSL) [13] for the computation of the shortest path.

[1]It can be downloaded at http://www.mpi-sb.mpg.de/~bast/collections/index.html

The program consists of a package containing classes which perform the following:

- read a matrix saved in the Harwell-Boeing format and compute its SVD (using Colt),
- compute the original and the truncated term-term matrix for a given number k of topics,
- compute the order of co-occurrence of each pair of terms. It was done using two methods:
  - *the shortest path method*: this is the method defined in Definition 3.1 and was realized by dint of JDSL,
  - *the T^l method*: this method originates from the knowledge we have from Lemma 3.1. That is, it uses the fact that a pair of terms has the co-occurrence order l if and only if its entry is zero in T^{l-1} and non-zero in T^l,
- compute the statistics needed.

The documentation of the program can be found under http://www.mpi-inf.mpg.de/~regis/lsi/package/.

3.4.2 Results and Interpretation

Two examples of the results we had can be seen in table 3.2 and 3.3. In these experiments, we only considered three cases for the computation of the truncated term-term matrix. Namely for $\kappa \in \{-1, 0, 1\}$, which are the most widely used cases. Although the T_k-values for $\kappa = 0$ are smaller than those for $\kappa = 1$ and even smaller for $\kappa = -1$, we came to the same conclusions for the different values of κ (see table 3.2 and 3.3).

After obtaining the results, we first noticed, as the authors did in [17], that all pairs of terms in the collections have an order of co-occurrence of at most 3. We also noticed that the third order co-occurrence pairs have T_k-values which are not significant compared to other T_k-values.

As we can see from table 3.2 and 3.3, first order co-occurrence pairs of terms with a low number of second order paths tend to have low T_k-values, as was also observed in [17]. However unlike what the authors of [17] stated about first order pairs with many (resp. a moderate number of) paths, we found

Term Term Mat Value	Order 1 Pairs	Order 2 Pairs	Order 3 Pairs	Length 2 Paths	Av. No Paths Order 1 pairs	Av. No Paths Order 2 pairs	Av. number paths
$<$ -2	2,173	6,743	0	3,942,722	650.50	375.08	442.21
-2 to -1.5	862	4,816	0	2,007,115	562.59	316.06	353.49
-1.5 to -1	1,465	11,969	0	4,171,449	525.74	284.17	310.51
-1 to -0.9	429	4,604	0	1,407,614	513.06	257.93	279.68
-0.9 to -0.8	526	6,062	0	1,750,367	489.89	246.24	265.69
-0.8 to -0.7	606	8,338	0	2,276,548	496.13	236.97	254.53
-0.7 to -0.6	759	11,731	0	2,943,794	469.74	220.55	235.69
-0.6 to -0.5	906	17,545	0	4,039,202	453.55	206.80	218.92
-0.5 to -0.4	1,174	27,665	0	5,726,987	437.15	188.46	198.58
-0.4 to -0.3	1,582	47,397	0	8,729,256	408.06	170.55	178.22
-0.3 to -0.2	2,251	93,432	0	15,008,670	394.27	151.14	156.86
-0.2 to -0.	3,583	247,066	0	32,019,266	348.22	124.55	127.75
-0.1 to 0	7,571	2,965,118	1	161,183,000	270.68	53.67	54.22
0 to 0.1	31,179	3,750,062	0	227,459,536	162.68	59.30	60.15
0.1 to 0.2	43,089	594,296	0	84,593,051	182.17	129.13	132.72
0.2 to 0.3	41,659	279,655	0	54,160,445	209.32	162.49	168.56
0.3 to 0.4	37,979	163,859	0	39,992,422	233.47	189.95	198.14
0.4 to 0.5	34,867	105,958	0	31,456,982	253.30	213.53	223.38
0.5 to 0.6	32,898	72,590	0	25,982,161	270.33	235.42	246.30
0.6 to 0.7	29,699	53,259	0	21,979,871	284.11	254.27	264.95
0.7 to 0.8	28,329	39,880	0	19,165,799	292.48	272.82	280.99
0.8 to 0.9	27,177	30,540	0	16,947,385	299.18	288.69	293.63
0.9 to 1	25,735	24,109	0	15,061,425	302.78	301.53	302.17
1 to 1.5	98,540	67,573	0	57,070,447	348.58	336.24	343.56
1.5 to 2	72,255	29,060	0	39,695,955	395.33	383.05	391.81
2 to 2.5	55,081	14,529	0	30,015,268	435.43	415.12	431.19
2.5 to 3	43,408	7,787	0	23,718,379	468.49	434.33	463.29
3 to 3.5	34,986	4,754	0	19,550,432	497.45	451.53	491.96
3.5 to 4	29,202	2,886	0	16,396,071	515.49	465.28	510.97
4 to 4.5	24,922	1,885	0	14,204,393	535.00	462.10	529.88
4.5 to 5	21,075	1,189	0	12,387,863	561.17	471.96	556.41
5 to 6	34,184	1,423	0	20,661,194	584.56	476.89	580.26
6 to 7	26,420	714	0	16,568,838	614.46	468.99	610.63
7 to 8	21,193	392	0	13,827,703	643.76	470.42	640.62
8 to 9	17,290	206	0	11,619,018	666.38	472.02	664.10
9 to 10	14,663	121	0	10,225,312	693.30	491.93	691.65
10 to 11	12,366	96	0	8,814,232	708.90	500.17	707.29
11 to 12	10,614	48	0	7,769,493	730.02	439.38	728.71
12 to 13	9,323	33	0	6,996,487	748.69	498.45	747.81
13 to 14	8,138	28	0	6,245,432	765.84	465.71	764.81
14 to 15	7,134	20	0	5,623,067	787.05	411.60	786.00
15 to 16	6,449	11	0	5,140,804	796.52	369.27	795.79
16 to 17	5,826	2	0	4,765,596	817.80	548.00	817.71
17 to 18	5,179	6	0	4,296,258	828.89	572.50	828.59
18 to 19	4,719	3	0	3,914,436	829.40	173.00	828.98
19 to 20	4,451	3	0	3,817,716	857.41	456.67	857.14
20 to 25	17,206	1	0	15,080,668	876.45	499.00	876.43
25 to 30	12,197	2	0	11,264,615	923.54	110.50	923.40
$\geq$ 30	69,059	0	0	83,269,847	1,205.78	0.00	1,205.78

Table 3.2: Average number of paths by T_k value for CRAN, $k = 100$ and $\kappa = 1$

Term Term Mat Value	Order 1 Pairs	Order 2 Pairs	Order 3 Pairs	Length 2 Paths	Av. No Paths Order 1 pairs	Av. No Paths Order 2 pairs	Av. number paths
$<$ -0.002	96,991	36,361	0	100,745,680	891.70	392.15	755.49
-0.002 to -0.0015	23,002	21,806	0	22,364,660	660.18	329.24	499.12
-0.0015 to -0.001	34,797	49,447	0	35,355,748	603.79	290.12	419.68
-0.001 to -0.0009	9,334	17,630	0	9,974,464	563.94	267.19	369.92
-0.0009 to -0.0008	10,267	21,678	0	11,313,455	553.47	259.75	354.15
-0.0008 to -0.0007	11,566	28,190	0	13,178,283	536.39	247.41	331.48
-0.0007 to -0.0006	13,326	37,372	0	15,614,471	511.08	235.57	307.99
-0.0006 to -0.0005	15,566	51,148	0	19,175,902	494.96	224.28	287.43
-0.0005 to -0.0004	18,023	73,612	0	24,008,273	477.89	209.14	262.00
-0.0004 to -0.0003	21,633	113,833	0	31,774,811	451.89	193.26	234.56
-0.0003 to -0.0002	26,732	197,340	0	45,071,122	419.94	171.51	201.15
-0.0002 to -0.0001	34,217	442,856	0	75,056,354	383.98	139.81	157.33
-0.0001 to 0	49,876	3,591,518	1	232,530,095	326.05	60.22	63.86
0 to 0.0001	84,928	3,326,963	0	222,152,303	248.53	60.43	65.11
0.0001 to 0.0002	78,521	347,766	0	68,942,075	267.64	137.81	161.73
0.0002 to 0.0003	61,282	135,403	0	41,243,217	299.55	169.02	209.69
0.0003 to 0.0004	48,408	69,365	0	28,930,127	323.77	191.12	245.64
0.0004 to 0.0005	39,463	39,832	0	21,826,822	341.06	210.07	275.26
0.0005 to 0.0006	33,641	24,962	0	17,674,232	359.96	222.93	301.59
0.0006 to 0.0007	27,085	16,660	0	14,194,374	379.49	235.04	324.48
0.0007 to 0.0008	22,939	11,983	0	12,030,287	395.50	246.84	344.49
0.0008 to 0.0009	19,937	8,503	0	10,382,877	411.85	255.41	365.08
0.0009 to 0.001	17,040	6,173	0	8,932,758	427.37	267.34	384.82
0.001 to 0.0015	60,220	15,959	0	32,381,664	463.61	279.67	425.07
0.0015 to 0.002	35,593	5,955	0	20,255,795	517.61	307.72	487.53
0.002 to 0.0025	24,309	2,762	0	14,663,550	567.31	316.04	541.67
0.0025 to 0.003	17,192	1,483	0	11,058,843	614.81	329.79	592.17
0.003 to 0.0035	12,539	876	0	8,444,113	649.85	337.43	629.45
0.0035 to 0.004	9,819	552	0	6,998,190	693.56	340.88	674.78
0.004 to 0.0045	7,904	335	0	5,771,548	715.34	350.73	700.52
0.0045 to 0.005	6,416	268	0	4,861,567	742.93	354.31	727.34
0.005 to 0.006	9,703	337	0	7,778,390	788.77	370.74	774.74
0.006 to 0.007	6,999	179	0	5,958,003	842.09	358.70	830.04
0.007 to 0.008	5,276	120	0	4,692,478	880.97	370.53	869.62
0.008 to 0.009	4,271	72	0	3,968,815	922.96	372.67	913.84
0.009 to 0.01	3,353	36	0	3,185,366	946.01	372.31	939.91
0.01 to 0.011	2,559	40	0	2,522,592	979.49	401.70	970.60
0.011 to 0.012	2,155	19	0	2,180,624	1,007.87	456.05	1,003.05
0.012 to 0.013	1,738	18	0	1,834,369	1,050.28	498.61	1,044.63
0.013 to 0.014	1,532	14	0	1,560,520	1,014.61	438.43	1,009.39
0.014 to 0.015	1,342	6	0	1,473,674	1,096.33	399.67	1,093.23
0.015 to 0.016	1,092	3	0	1,194,209	1,092.98	225.67	1,090.60
0.016 to 0.017	984	6	0	1,077,567	1,091.41	603.00	1,088.45
0.017 to 0.018	871	6	0	967,508	1,107.96	411.83	1,103.20
0.018 to 0.019	758	0	0	871,450	1,149.67	0.00	1,149.67
0.019 to 0.02	613	3	0	696,109	1,134.57	206.00	1,130.05
0.02 to 0.025	2,351	11	0	2,794,672	1,187.11	343.36	1,183.18
0.025 to 0.03	1,307	4	0	1,592,272	1,216.41	606.25	1,214.55
$\geq$ 0.03	2,908	1	0	3,688,313	1,268.15	547.00	1,267.90

Table 3.3: Average number of paths by T_k value for CRAN, $k = 100$ and $\kappa = 0$

out that first order pairs with a higher average number of paths tend to receive high T_k-values. And first order pairs with a moderate number of paths can have positive as well as negative T_k-values.
Second order co-occurrence pairs of terms with a smaller number of second order paths have a very low T_k-value, as remarked in [17]. We also observed, as in [17], that second order pairs with a higher number of paths have higher T_k-values. Yet second order pairs with a moderate number of paths do not necessarily tend to have negative T_k-values , as the authors stated in [17], but can also have positive values as well.

3.5 Conclusion

In this chapter, we studied the importance of term co-occurrences for LSI (i.e. T_k) based on what the authors investigated in [17].
We first noticed that in [17], the authors made some mistakes whilst trying to detect the relation between the entries of the truncated term-term matrix and the term co-occurrences. We corrected those mistakes and thus had slightly different conclusions.
However concerning some cases, we are not able to say why the entries of T_k are positive or negative. Thus, a further task would be necessary to be able to formulate more precisely and with sound arguments how the truncated term-term matrix uses the term co-occurrences.
In the next chapter, we will use some approximations of the truncated term-term matrix in order to have a better understanding of the relation between T_k and the term co-occurrence information.

Chapter

4 Approximating the Truncated Term-term Matrix

In this chapter, we try to find an *approximation* of the truncated term-term matrix. More precisely, we want to compute matrices using the term co-occurrence information (without computing the SVD) which gives us comparable or even better retrieval performances than T_k. We will then see that it is possible to have good results with the approximations of T_k, even though we are not already able to compute them automatically yet. The approximations we have in this chapter will help us to have a better understanding of how LSI actually works.

4.1 Idea

According to the last chapter, the first and second co-occurrence pairs of terms play an important role for the value of the entries of T_k. From Lemma 3.1, we know that T and T^2 are the important matrices for these two kinds of pairs of terms, because they are the only ones which contain just those two orders of co-occurrences. The idea here is to approximate T_k by using those two matrices.

Before approximating T_k, we first have to try to detect which kind of relation exists between T_k and T and also between T_k and T^2 (i.e. between the entries of the matrices). That would then be our starting point for finding

Variants of LSI		$\kappa = 0$	$\kappa = 1$	$\kappa = -1$
Med (rank = 1033)	best average precision	0.4872	0.4629	0.4370
	value of k	126	685	119
Time (rank = 425)	best average precision	0.4165	0.3437	0.3617
	value of k	201	422	81
Cran (rank = 1400)	best average precision	0.3255	0.1736	0.2766
	value of k	1314	492	151

Table 4.1: Best average precision with our collections for three variants of LSI.

an approximation of T_k.

4.2 Experiments

All the programs used for the experiments were written in Matlab. For the experiments, we had the following collections which were used because the queries and the relevance judgement sets were already available:

- MED, a collection of medical abstracts containing 1,033 documents and 4,250 terms. There were 30 queries available for this collection.

- CRAN, a collection with 1,379 documents and 4,410 terms. Here, we had 225 queries.

- TIME, a collection with 1,460 documents and 5,753 terms. There were 83 queries available.

For each collection and for each value of $\kappa \in \{-1, 0, 1\}$, we computed the value of $k \in \{1, \ldots, r\}$ (where r is the rank of the term-document matrix) for which T_k had the best *average precision*. Here, the average precision is the average over the averages of the precisions of all the queries at the recalls 5%, 10%, 15%, ...,100%. In Table 4.1, we can see the values of k for which we had the best average precision for each value of κ and the respective average precisions.
In order to detect a relation between T_k and T (resp. T^2) or to approximate T_k, we always used the truncated term-term matrix computed with the value of k for which the average precision is maximised.

4.2.1 Detecting a Relation between T_k and T (resp. T^2)

In order to detect a relation between T_k and T, we used two methods.

In the first method, we plotted clouds of a set of points. Each point of those clouds corresponded to a randomly chosen pair of terms and had the coordinates (x, y), such that x is the entry of the pair in T (resp. T^2) and y the entry of the pair in T_k. We then tried to recognise any kind of relation from the plots.

In the second method, we computed the correlation coefficient between the T_k-values of a set of randomly chosen pairs of terms and their T-values (resp. T^2-values). The formula used for this purpose is the standard one for the correlation coefficient and is also explained in [20]. So, let X and Y be two random variables which represent two lists of numbers, the correlation coefficient of X and Y is defined as:

$$\rho_{XY} = \frac{\mathrm{Cov}\,(\mathrm{X}, \mathrm{Y})}{\sigma_X \sigma_Y},$$

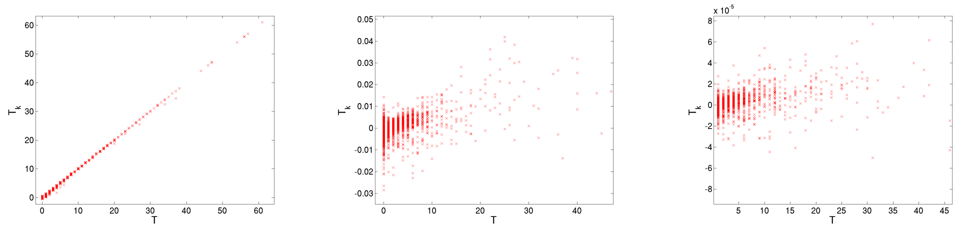
where $\mathrm{Cov}\,(\mathrm{X}, \mathrm{Y})$ is the covariance of X and Y, σ_X (resp. σ_Y) is the standard deviation of X (resp. Y). The value of ρ_{XY} is always between -1 and 1. And a value close to -1 or 1 implies that X and Y are well correlated, a value close to 0 means that there is no correlation between X and Y.
This method was also applied for $\kappa \in \{-1, 0, 1\}$.

For both methods, the sets of pairs of terms were chosen randomly, because the total number of the pairs is very large[1]. For each method, the sets were chosen independently, such that we could assume that a detected relation does in fact exist.

We first wanted to know whether a relation exists between the entries of T_k and those of T. Due to the fact that $T_k = U_k \Sigma_k^2 U_k^T$ for $\kappa = 1$ and $T = U \Sigma^2 U^T$, there was a strong linear relation between T_k and T for $\kappa = 1$ (see Figure 4.1 (a), (b) and (c)). The correlation coefficient between the entries of T_k and those of T was also fairly high (it was always above 0.95). But for other values of κ, we couldn't detect any kind of relation, as can be seen in Figure 4.1, and the correlation coefficients were low (i.e. between 0.1 and 0.4).

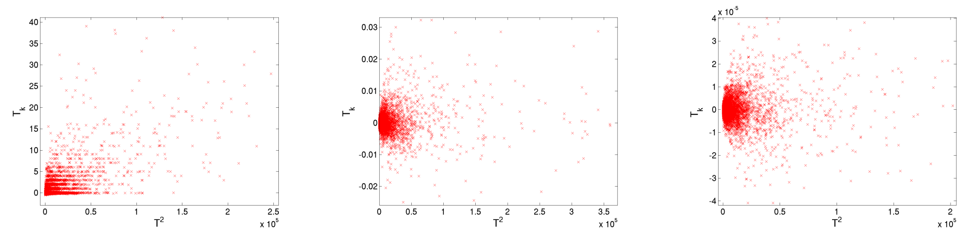
We then wanted to know if a relation exists between the entries of T_k and

[1] For a $m \times n$ term-document matrix, the total number of pairs of terms is $\frac{m(m+1)}{2}$



(a) (b) (c)

Figure 4.1: Plots T_k-T for Med. In (a) T_k was computed with $\kappa = 1$, in (b) $\kappa = 0$ and in (c) $\kappa = -1$.



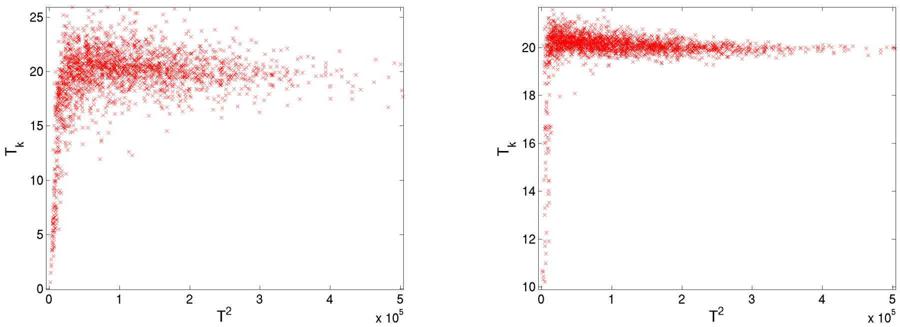
(a) (b) (c)

Figure 4.2: Plots T_k-T^2 for Med. In (a) T_k was computed with $\kappa = 1$, in (b) $\kappa = 0$ and in (c) $\kappa = -1$.

those of T^2. However we obtained the same result as with T. That is, there was a slightly linear relation between the entries of T_k and those of T^2 for $\kappa = 1$ and no apparent relation for $\kappa \in \{0, -1\}$ as can be seen in Figure 4.2. The correlation coefficients also confirm these observations.

Thereupon, we wanted to find a relation between T_k and both T and T^2. For that purpose, we made a restriction on the original term-term value of the pairs of terms and then tried to find a relation between the entries of T_k and those of T^2 for the pairs which satisfied the restriction. Here, the restriction means, that all plotted pairs must have a particular T-value t, or have to be between two values t_1 and t_2.
The clouds obtained in this case were more interesting than those obtained before. These clouds were influenced by one or two lines, depending on the values of κ and k used to compute T_k.

When $\kappa = 1$, the clouds are always influenced by two lines. The first line always has a positive slope, and that slope increases the higher k becomes. The second line is always horizontal, and located at the T_k-value t (the T-



(a) (b)

Figure 4.3: Plots T_k-T^2 for Med ($\kappa = 1$) with a restriction on the T-value. For both plots, the T-value of each pair is 20. In (a) T_k was computed with $k = 200$ and in (b), $k = 500$.

value of all pairs in the cloud). An example can be seen in Figure 4.3.
But when $\kappa \in \{0, -1\}$, the form of the clouds depends on the value of k as we can see in Figure 4.4. When k is small (i.e. about 1/10-th or 2/10-th of the rank), the cloud is affected by two lines, the first one having a positive slope and the second one a negative slope. For moderate values of k (i.e. about 4/10-th of the rank), the first line (with the positive slope) seems to disappear. The cloud is thus influenced by just one line, which has a negative slope. And for high values of k (i.e. about 8/10-th of the rank or above), the cloud is then clearly influenced by two lines again. The first line has a negative slope, and the second one is horizontal and located at the T_k-value 0.

Furthermore, when we computed the correlation coefficient between the entries of T_k and those of T^2 which have a given T-value, we first noticed that the value obtained is very small, which is coherent since we saw that most of the clouds were influenced by two lines. Yet, when we also made a restriction on the T^2-values of the pairs of terms, the correlation coefficients obtained confirmed the observations from the clouds.

With $\kappa = 0$ for example (see Table 4.1), the optimal value of k, according to the average precision, for the collection Med is small compared to the rank of the term-document matrix. Thus, the clouds obtained in this case are influenced by two lines, the first one having a positive slope and the second one a negative slope (as in Figure 4.4(a)). Since for each T-value the first line always starts from the origin (i.e., the point for which we have

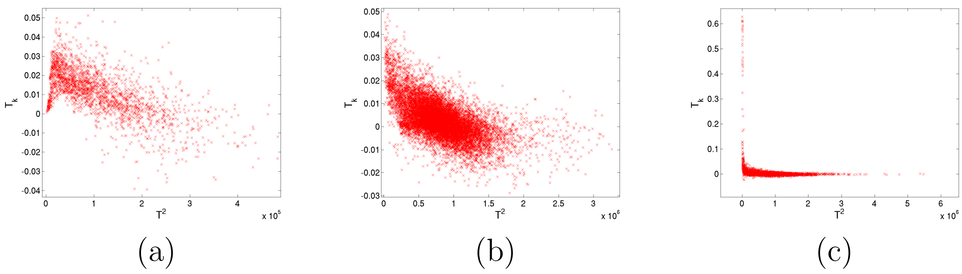


Figure 4.4: Plots T_k-T^2 ($\kappa = 0$) with a restriction on the T-value. For the three plots, the T-value of each pair is 20. In (a) we used the collection Med and T_k was computed with $k = 126$. Time was used for (b) $k = 201$ and in (c), we used Cran and $k = 1314$.

$\tilde{t}_{ij} = t_{ij}^2 = 0$) and seems to have the same slope, we assumed that the T-value does not have an impact on it. We could thus see from the plots that a possible approximation of T_k could be

$$\tilde{T} \approx \min\bigl(\beta_1 T^2, \alpha_1 T - \beta_2 T^2\bigr) . \tag{4.1}$$

Thus finding the appropriate α_1, β_1 and β_2 would be a starting point in order to find an approximation of that T_k.
Knowing that the ranking is not affected when the truncated term-term matrix used is multiplied by a factor, and also that for each $x \in \mathbb{R}$ and two matrices A and B, we have $\min(c \cdot A, c \cdot B) = c \cdot \min(A, B)$, the number of variables we have to find can be reduced when we use the formula:

$$\tilde{T} \approx \min\left(T^2, \alpha T - \beta T^2\right) , \tag{4.2}$$

with $\alpha = \frac{\alpha_1}{\beta_1}$ and $\beta = \frac{\beta_2}{\beta_1}$.

For $\kappa = 0$ again, the optimal value of k for the collection Time is moderate, compared to the rank of the term-document matrix. The clouds obtained here are thus influenced by just one line, which has a negative slope (as in Figure 4.4(b)). Thus, we tried here to approximate the truncated term-term value with

$$\tilde{T} \approx \alpha T + \beta T^2, \tag{4.3}$$

where α and β have to be determined.

With the collection CRAN and also $\kappa = 0$, the optimal value of k is very high (see again Table 4.1). the clouds obtained here practically consist of two lines, the first having a very high negative slope (almost $-\infty$), and the second being horizontal and located at the T_k-value 0 (as in Figure 4.4(c)). Since we couldn't deduce a reasonable formula from the clouds, we tried to approximate T_k with the formula seen in equation 4.2 and equation 4.3.

4.2.2 Approximation of T_k

Here, we tried to find some matrices which can be used in order to approximate T_k, i.e., we tried to find the optimal values of α and β for which we have the best average precision when $\tilde{T}$ from equation 4.2 and equation 4.3 is used as the document expansion matrix. For this purpose, we used the optimisation method of Matlab which is defined in [21]. That method is called *Nelder-Mead Simplex algorithm* and was first presented in [25]. The method requires initial values in order to compute the optimum.
As in the previous section, we tried to approximate T_k for $\kappa \in \{-1, 0, 1\}$. The approximations were compared to T_k with respect to the average precisions. We also computed the average precision for the case in which the identity matrix is used as the document expansion matrix (which is in fact the basic vector space model with the cosine similarity measure) in order to compare it with the approximations.

4.2.2.1 Approximation of T_k for $\kappa = 0$

We started with $\kappa = 0$, because the ranking can be easily computed, as seen in equation 2.5. For each collection, we tried to approximate the best truncated term-term matrix (i.e. the one with the best average precision as seen in Table 4.1) using the two formulas (i.e. equations 4.2 and 4.3) we derived in Section 4.2.1.

Approximation with $\min(T^2, \alpha T - \beta T^2)$
Our task here was to compute $\tilde{T}$ from equation 4.2 for which the average precision is maximised. We first had to compute the initial values for α and β.

For Med, we first tried 'manually' to calculate α_1, β_1 and β_2 from equation 4.1 (i.e., we chose some points from the plots in order to determine those values). We then calculated α and β from equation 4.2 using the formulas $\alpha = \frac{\alpha_1}{\beta_1}$ and $\beta = \frac{\beta_2}{\beta_1}$. Once we had the desired values, we then optimised them in order to have the values for which the average precision of $\tilde{T}$ is maximised. The average precision obtained was close to that of LSI (i.e. T_k).

We also computed the optimal $\tilde{T}$ with $\alpha = 0$ and $\beta = 0$ as initial values. The result we obtained here was better than the previous one and was much closer to that of LSI.
For Time and Cran, we used $\alpha = 0$ and $\beta = 0$ as initial values, because this formula is not derived from their $T^2 - T_k$ plots. For neither of the collections, the average precision was as close to that of LSI as with Med.

Approximation with $\alpha T + \beta T^2$
Here, we can first notice that this case is a kind of simplification of the one before.
This formula was derived from the plots of Time. So for Time, we manually computed the initial values of α and β as in the previous case. The average precision obtained after optimising was then exactly the same we had with the formula $\min(T^2, \alpha T - \beta T^2)$, yet with other values of α and β. In addition, when we used $\alpha = 0$ and $\beta = 0$ as initial values, we obtained the same average precision.

For Med and Cran we used $\alpha = 0$ and $\beta = 0$ as initial values. And even for these collections, the average precisions obtained were exactly the same we had with $\min(T^2, \alpha T - \beta T^2)$ (with $\alpha = 0$ and $\beta = 0$ as initial values).

An overlook of all the average precisions for the approximations can be seen in Table 4.2.

4.2.2.2 Approximation for $\kappa = 1$ and $\kappa = -1$

Here, we used the formula of equation 2.4 in order to compute the rankings, and thus the average precisions. We first noticed that the average precisions obtained with LSI using these two values of κ were slightly worse than those obtained with $\kappa = 0$, as already remarked in [27] (see Table 4.1). We also computed the best average precisions of the two approximations already seen in Section 4.2.2.1. However the results we had were not as good as those obtained in the previous section.

4.2.2.3 Discussion

Let us now analyse the results.
We should first of all remark that for each approximation, the entries of the matrices αT and those of βT^2 had approximately the same size. Thus, neither of them was dominant in the approximations $\alpha T + \beta T^2$ and $\min(T^2, \alpha T - \beta T^2)$.

	Approximations	LSI ($\kappa = 0$)	$\min\left(T^2, \alpha T - \beta T^2\right)$		$\alpha T + \beta T^2$	
Med	best average precision	0.4872	0.4793	0.4831	0.4831	0.4831
	optimal α	–	510.36	$5.11 \cdot 10^{-4}$	1675.4	$5.11 \cdot 10^{-4}$
	optimal β	–	0.0473	$4.66 \cdot 10^{-8}$	-0.1526	$-4.66 \cdot 10^{-8}$
	initial α	–	1600	0	1600	0
	initial β	–	-0.16	0	-0.16	0
Time	best average precision	0.4165	0.3729	0.3729	0.3729	0.3729
	optimal α	–	$7.59 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$	$7.59 \cdot 10^{-4}$	$2.5 \cdot 10^{-4}$
	optimal β	–	$1.6 \cdot 10^{-8}$	$5.28 \cdot 10^{-9}$	$-1.6 \cdot 10^{-8}$	$-5.28 \cdot 10^{-9}$
	initial α	–	$7.5 \cdot 10^{-4}$	0	$7.5 \cdot 10^{-4}$	0
	initial β	–	$-1.5 \cdot 10^{-8}$	0	$-1.5 \cdot 10^{-8}$	0
Cran	best average precision	0.3255	0.2355		0.2355	
	optimal α	–	$3.8 \cdot 10^{-4}$		$3.8 \cdot 10^{-4}$	
	optimal β	–	$7.5 \cdot 10^{-9}$		$-7.5 \cdot 10^{-9}$	
	initial α	–	0		0	
	initial β	–	0		0	

Table 4.2: Average precisions of the approximations of the truncated term-term matrices for all collections.

For each collections, we always had the same optimal average precision for both approximations (see Table 4.2). We even had the same optimal values for α and β when both approximations where optimised with $(0, 0)$ as initial value for (α, β).
The formula $\min\left(T^2, \alpha T - \beta T^2\right)$ was derived from the collection Med (see Figure 4.4(a)). We also know that the $T^2 - T_k$ plot from Figure 4.4(a) is representative for every collection with small optimal value k for the truncated term-term matrix. In this case, the cloud with the positive slope in the plot is not significant and does not improve the retrieval performance. T_k can thus be approximated by only using the formula $\alpha T + \beta T^2$, which represents the second cloud in the plot. LSI thus here seems to make some computations which are not relevant in order to have a better retrieval performance.
For Time, which is the collection from which the formula $\alpha T + \beta T^2$ was derived (see Figure 4.4(b)), we also observed that the optimal values for α and β were even the same for both approximations. This also means that the first argument of $\min\left(T^2, \alpha T - \beta T^2\right)$ is not important.
For Cran, which represents collections having a large optimal value of k for T_k, we also had the same average precision and optimal values for α and β for both approximations.
With the approximation $\alpha T + \beta T^2$, the optimal α was always positive and

β negative for all collections. It thus represents the clouds with the negative slope that can be seen on the plots for Med and Time (see Figure 4.4(a) and (b)). On Figure 4.4(c), we can see that such a cloud does not exist for Cran. That explains the fact that the average precisions obtained for Cran are not good. Other attempts to approximate the truncated term-term matrix of Cran were not successful.

We also observed that for the collections Med and Time, we had different optimal values of α and β, but the same average precisions with the approximation $\alpha T + \beta T^2$. This fact can be easily explained, when we notice that for those collections, the ratio $\frac{\alpha}{\beta}$ is always the same for the different optimal pairs. That means that the first approximation can be written as a factor multiplied by the other approximation. Let T_1 be the matrix obtained from the first optimal pair (α_1, β_1) (i.e. $T_1 = \alpha_1 T + \beta_1 T^2$), and T_2 the matrix obtained with the second one (α_2, β_2) ($T_2 = \alpha_2 T + \beta_2 T^2$). Because of the fact that

$$\frac{\alpha_1}{\beta_1} = \frac{\alpha_2}{\beta_2} \Longleftrightarrow \alpha_1 \beta_2 = \alpha_2 \beta_1 \Longleftrightarrow \frac{\alpha_1}{\alpha_2} = \frac{\beta_1}{\beta_2}$$

we have

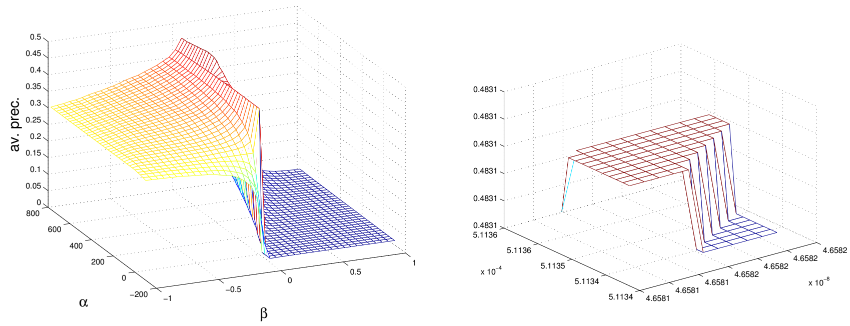
$$\frac{\alpha_2}{\alpha_1} T_1 = \alpha_2 T + \underbrace{\frac{\alpha_2 \beta_1}{\alpha_1}}_{=\beta_2} T^2 = T_2$$

The equation above justifies the same average precisions for both optimal pairs, knowing that $\frac{\alpha_2}{\alpha_1} > 0$ and $\frac{\beta_1}{\beta_2} > 0$. That fact can also be seen on Figure 4.6. The points on the ridge on that figure represents the values for which the average precision is maximal. We can also remark that the value of the average precision drops to 0 when α becomes negative. This is due to fact that the ranking computed in that case is almost the reverse of the one computed with a positive α.

With the formula $\min(T^2, \alpha T - \beta T^2)$, we also have such a ridge (see Figure 4.5). However, the ridge in this case is not straight line, because the higher α and β becomes, the influence of the first argument of the formula (i.e. T^2) also grows. The average precision then drops very slowly, such that it cannot be noticed on the plot. When we analyse the formula $\alpha T + \beta T^2$ using what we calculated in equation 3.1, we have for $i, j \in \{1, \ldots, m\}$ with $i \neq j$:

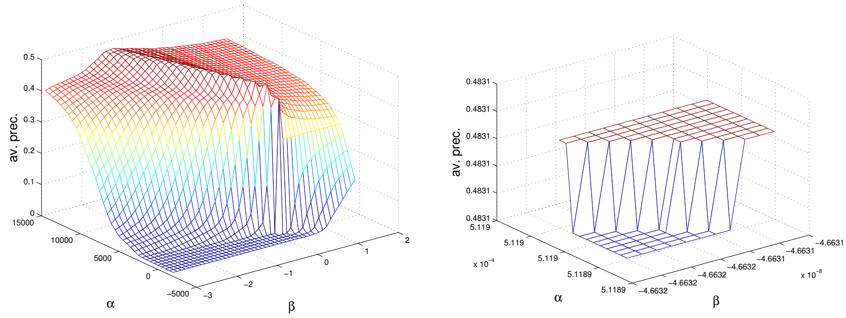
$$\begin{aligned} \tilde{t}_{ij} &= \alpha t_{ij} + \beta t_{ij}^{(2)} \\ &= \alpha t_{ij} + \beta \left(\sum t_{ix} t_{xj} + (t_{ii} + t_{jj}) t_{ij} \right) \\ &= (\alpha + \beta t_{ii} + \beta t_{jj}) t_{ij} + \beta \sum t_{ix} t_{xj} \end{aligned} \tag{4.4}$$

We can see from equation 4.4, that the ijth entry in the approximation is a combination of the weights of the first and the second degree co-occurrence



(a) (b)

Figure 4.5: Average precision with $\min(T^2, \alpha T - \beta T^2)$ for different α and β. (a) is a global view and (b) is a zoomed view at the optimum.



(a) (b)

Figure 4.6: Average precision with $\alpha T + \beta T^2$ for different α and β. (a) is a global view and (b) is a zoomed view at the optimum.

paths between i and j. Thus, the goal of the optimisations that we made in this chapter was to find which amount of first and second degree paths is needed for a good retrieval performance. LSI actually does that automatically for each pair of terms (by just using the singular value decomposition). We see for example on Figure 4.5 and 4.6 the behaviour of the average precisions, when we vary those two parameters. With $\min(T^2, \alpha T - \beta T^2)$, the average precision drops very fast for positive values of β. In that case, the expression $\alpha T - \beta T^2$ is negative and the first argument of the approximation is not important. That then leads to a reverse ranking.

4.2.3 Combining LSI (for $\kappa = 0$) with the basic vector space model

Here we wanted to combine LSI with the raw vector space in order to obtain better rankings than LSI, and thus better average precisions. We know that computing the rankings in the basic vector space model is the same as computing the rankings with equation 2.5 using the identity matrix I as the document expansion matrix. Thus combining it with LSI means that we use a $\tilde{T}$ as the expansion matrix, such that

$$\tilde{T} = \lambda I + (1 - \lambda)\, T_k, \tag{4.5}$$

for $\lambda \in [0, 1]$ and an appropriate T_k.

For each collection, we started by computing the value of λ and k for which the average precision is maximised. We did so by optimising λ for each possible value of k. The average precision computed with the matrices obtained was always greater than the approximation we had in the section before, and even greater or equal the average precision from LSI.
We then tried to approximate that combination. We did so by using the identity matrix I and $\alpha T + \beta T^2$. We chose $\alpha T + \beta T^2$ for this approximation, because its formula is simpler than $\min(T^2, \alpha T - \beta T^2)$ and mainly because we had the same results with both formulas. Thus, our goal was to find the values of α and β for which the matrix

$$\tilde{T} = I + \alpha T + \beta T^2 \tag{4.6}$$

had the best average precision. For this purpose also, we used the formula in equation 2.5 and $(\alpha, \beta) = (0, 0)$ as initial value.

Discussion

The average precisions obtained here were always better than that of LSI. For Med and Cran, it was even better than the one obtained with $\lambda I + (1 - \lambda)\, T_k$. The results can be seen on Table 4.3. We also remarked here that the optimal α is positive and β negative for Med and Time, as for the approximations in the section before. For Cran though, it was quite the the opposite, and we did not find any explanation for that fact.

A comparison of all the approximations for each collection can be seen in Figures 4.8, 4.9 and 4.10. The case $\min(T^2, \alpha T - \beta T^2)$ was removed from

Collections		Med	Time	Cran
Vector space model	average precision	0.4574	0.4136	0.3250
LSI ($\kappa = 0$)	average precision	0.4872	0.4165	0.3255
Combination $\lambda I + (1-\lambda)\,T_k$	average precision	0.5007	0.4361	0.3255
	value of k	107	50	1314
	value of λ	0.4125	0.4561	0
$I + \alpha T + \beta T^2$	average precision	0.5020	0.4210	0.3300
	value of α	$1.7 \cdot 10^{-3}$	$5.1 \cdot 10^{-5}$	$-9.5 \cdot 10^{-4}$
	value of β	$-1.5 \cdot 10^{-7}$	$-5.4 \cdot 10^{-10}$	$2.4 \cdot 10^{-9}$

Table 4.3: Average precisions of the approximations of the combination of LSI and the vector space model

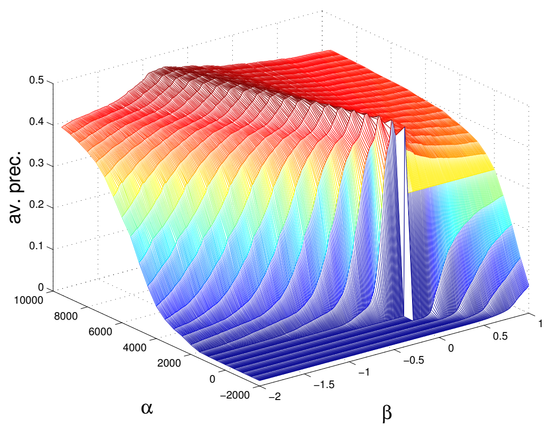


Figure 4.7: Behaviour of the average precision with $I + \alpha T + \beta T^2$ (Med).

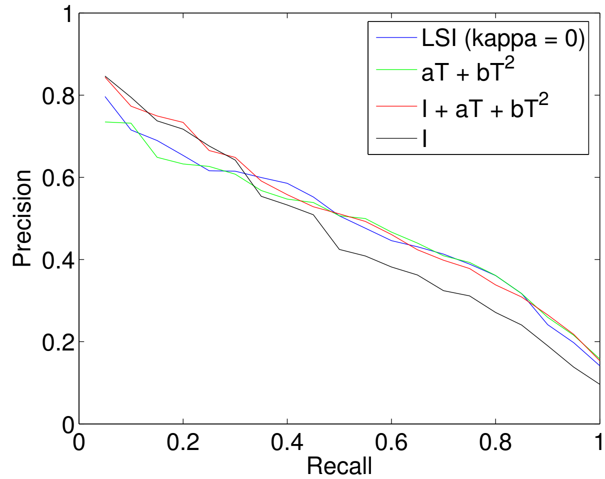


Figure 4.8: Comparison of the approximations of T_k for $\kappa = 0$ for Med

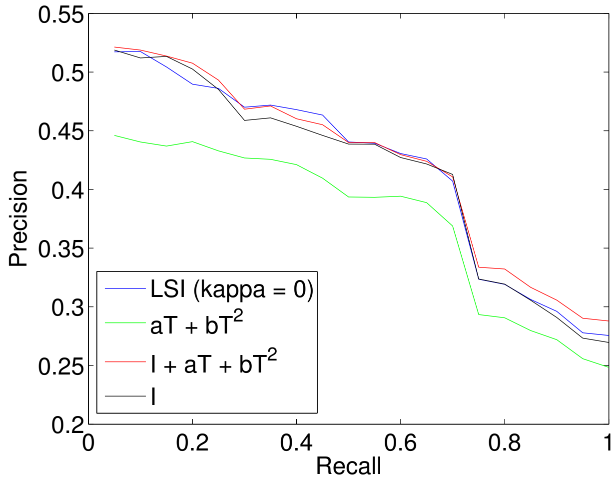


Figure 4.9: Comparison of the approximations of T_k for $\kappa = 0$ for Time

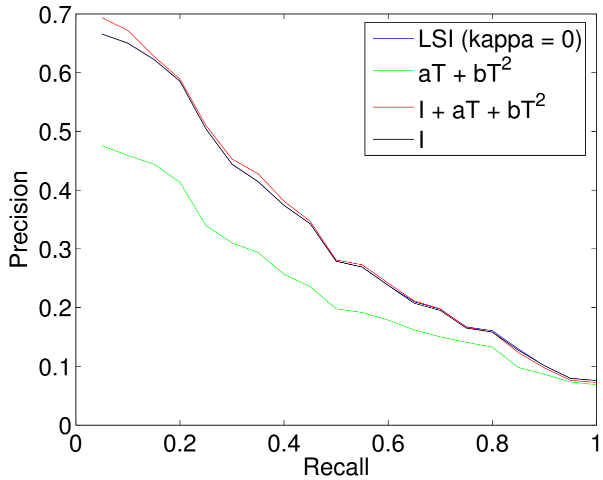


Figure 4.10: Comparison of the approximations of T_k for $\kappa = 0$ for Cran

the comparison because the results were the same as for the approximation $\alpha T + \beta T^2$.

For the collection Med, we can first notice that LSI and the first approximation (i.e. $\alpha T + \beta T^2$) have comparable retrieval performances. Second, we can see that although the identity matrix (i.e., the basic vector space model) provides better precisions for small recalls, its precision curve drops faster than the other curves. And third, we can also see that the approximation $I + \alpha T + \beta T^2$ combines the best precision of both the vector space model and LSI.

For Time and Cran, the precision-recall curve of $\alpha T + \beta T^2$ is not as good as the ones of LSI or $I + \alpha T + \beta T^2$. Yet we can see that the approximation $I + \alpha T + \beta T^2$ is always better than LSI and the basic vector space.

Each entry in $I + \alpha T + \beta T^2$ is a combination of the first and the second order co-occurrences of the respective pair of terms, with an additional weight for the pairs (t, t) for a term t.

4.3 Conclusion

In this chapter, we computed some approximations of the truncated term-term matrix and analysed them. Due to the fact that T and T^2 were the only matrices used for the computations of the approximations, and knowing that the entries of these matrices consist of the first and second order co-occurrences, we also see here (as in chapter 3), that LSI essentially uses the term-term relationships based on co-occurrence information.

In [2], the authors showed, using the so-called *synonymy graph*, that the entries of the truncated term-term matrix (for $\kappa = 0$) strongly depends on the term-term relationships of the pairs of terms. Thus, we see that LSI (in fact the singular value decomposition) computes those relationships in a very efficient way, although some of the computations do not improve its retrieval performance. An explicit computation of these relationships would be very expensive, because we would need to compute them for each pair of terms. And we know that there exists $\frac{m(m+1)}{2}$ pairs of terms for an $m \times n$ term-document matrix.

Chapter 5

Conclusion

LSI is a retrieval technique based on the dimension reduction of the term-document matrix from the vector space model, which can cope with polysemy and synonymy, two well-known problem in information retrieval. Experimentally it has been shown that, for many collections, the retrieval performance obtained with LSI is better than that obtained with the basic vector space model.

In this thesis, we study the relation between the entries of the truncated term-term matrix, which is the document expansion matrix produced by LSI, and the term co-occurrence information of the respective pairs of terms. We first corrected what the authors carried out in [17, 15]. In contrast to what they claim, we found out that it is not possible to predict the sign of an entry of the truncated term-term matrix when only the degree of co-occurrence of the respective pair of terms is known.

Second, we computed some approximations of the truncated term-term matrix by using the first and second co-occurrence information provided by the term co-occurrence matrices T and T^2. We then found out that a simple combination of both co-occurrence information has a comparable retrieval performance to that of LSI. Furthermore, we also provided an insight into the way LSI works. We saw that LSI computes the co-occurrence information in a much more efficient way than a straightforward method would. However, we also found out that the LSI sometimes seems to make some computations which are not needed in order to improve its retrieval performance. We also remark that the combination of LSI and the basic vector space model is an improvement of LSI in the sense of its retrieval performance.

Future Work

In this thesis, we made a step for a better understanding of LSI via its expansion matrix. However, many things still have to be done.

In order to compute the approximations, we had to optimise the values of the variables α and β. It would be better to have strong theoretical foundations which could permit us to compute those factors automatically.

We would also like to have better approximations (e.g. for the Cran collection), which would provide us a better insight into the way LSI works. Although we saw that LSI computes the term co-occurrence information in a very efficient way, it would interesting to search for other expansion matrices whose computations would be more efficient.

As mentioned in [2], the truncated term-term matrix is symmetric, as its approximations in this thesis are. However, it would be sometimes better to have an asymmetric expansion matrix. Further research in that direction would thus be helpful.

Bibliography

[1] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[2] Holger Bast and Debapriyo Majumdar. Understanding spectral retrieval via the synonymy graph. In *28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*, pages ?–?, Salvador, Brazil, August 2005. ACM.

[3] Michael K. Bergman. The deep web: Surfacing hidden value, September 2001.

[4] Michael W. Berry, Zlatko Drmac, and Elizabeth R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Review*, 41(2):335–362, April 1999.

[5] Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, pages 573–595, December 1994.

[6] The Colt Distribution. http://hoschek.home.cern.ch/hoschek/colt/, 2002.

[7] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis.

Journal of the American Society of Information Science, 41(6):391–407, 1990.

[8] Georges Dupret. Latent concepts and the number orthogonal factors in latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 221–226, New York, NY, USA, 2003. ACM Press.

[9] Norbert Fuhr. Probabilistic models in information retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[10] 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 *SIGIR '88: Proceedings of the 11th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 465–480. ACM Press, 1988.

[11] F. R. Gantmacher. *Matrizenrechnung II*, page 87. VEB Deutscher Verlag der Wissenschaften, Berlin, 1959.

[12] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*, chapter 2, pages 48–86. Johns Hopkins University Press, Baltimore, Maryland, third edition, 1996.

[13] The Data Structures Library in Java (JDSL). http://www.jdsl.org/, 2004.

[14] Karen Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 2. *Information Processing and Management*, 36(6):809–840, 2000.

[15] April Kontostathis and William M. Pottenger. Detecting patterns in the LSI term-term matrix. In IEEE International Conference on Data Mining (ICDM'02), editor, *Proceedings of the Workshop on Foundations of Data Mining and Discovery*, December 2002.

[16] April Kontostathis and William M. Pottenger. Improving retrieval performance with positive and negative equivalence classes of terms, 2002.

[17] April Kontostathis and William M. Pottenger. A mathematical view of latent semantic indexing: Tracing term co-occurences, 2002.

[18] April Kontostathis and William M. Pottenger. A framework for understanding LSI performance. In *Proceedings of ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval (ACMSIGIRMF/IR '03)*, 2003.

[19] April Kontostathis and William M. Pottenger. A framework for understanding latent semantic indexing (LSI) performance, 2004. Information Processing and Management. Preprint.

[20] Ulrich Krengel. *Einführung in die Wahrscheinlichkeitstheorie und Statistik*. Vieweg Verlag, 7 edition, August 2003.

[21] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9(1):112–147, 1998.

[22] Debapriyo Majumdar. *The Optimal Dimension in Latent Semantic Analysis*. PhD thesis, Max-Planck Institut Informatik Saarbrücken, 2004. ongoing work.

[23] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*, chapter 15, pages 529–574. The MIT Press, Cambridge, Massachusetts, 1999.

[24] William Mill and April Kontostathis. Analysis of the values in the LSI term-term matrix. Technical report, Ursinus College, 2004.

[25] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[26] Christos H. 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, June 1-3, 1998, Seattle, Washington*, pages 159–168. ACM Press, 1998.

[27] Josiane Xavier Parreira. Information retrieval by dimension reduction - a comparative study. Master's thesis, Saarland University, 2003.

[28] Hinrich Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.