Alcom-FT Technical Report Series
ALCOMFT-TR-03-185

Query Transformation through approximated LSI computation*

Luigi Laura† Umberto Nanni* Fabiano Sarracco*

December 2, 2003

Abstract

The Latent Semantic Indexing (LSI) proved to be an effective technique in the field of Information Retrieval. Its drawbacks are the time needed to compute the SVD decomposition and to answer queries, since the query must be compared against each document in the collection. In this paper we present a technique that uses the information computed by the traditional LSI to provide a fast online answer to the users. Informally, if compared to the traditional LSI, we can say that instead of returning the documents that are related to the concepts in the query, we return the documents that contain terms that are related to the concepts in your query.

1 Introduction

The Latent Semantic Indexing (LSI) has been introduced by Deerwester *et al.* in [10, 6]. At high level, the information retrieval process is done by taking advantage of the *latent semantic structure* hidden in the document, i.e., a higher order structure in the association of terms with documents. Using the singular-values decomposition (SVD), the original term-document matrix is approximated by a low rank one that is obtained considering only the k largest singular values; the other singular values are considered to be some sort of "noise" in the semantic structure of the collection of documents and are therefore omitted. Both terms and documents are projected into this k dimensional space, that we can think as the "semantic" or "concept" space. Here a term can be adjacent to a document that does not include it if this is consistent to the higher order structure in the collection of documents. For example, some webpages often refer to Microsoft as the "evil empire", and the noun Microsoft is not present in that pages.

LSI proved to be a very effective retrieval technique; on the other side its drawbacks are the computational time needed by SVD and query answer since for each query the distance between the query and *each* document in the collection is computed. For huge collections this is unfeasible, if we want to provide a fast online query answer. In this paper we present a technique that tries to capture the information in the reduced matrix A_k without explicitly computing the distance between the query q and every document in the collection.

Our approach is very close to the intuition behind LSI, i.e., that, citing from [8], "there is a *latent* structure in the pattern of word usage that is partially obscured by word choice". As we said

*Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

†Dip. di Informatica e Sistemistica Università di Roma "La Sapienza" Via Salaria 113 00198 Roma Italy. E-mail: {laura,nanni,sarracco}@dis.uniroma1.it

above, LSI works in the reduced "concepts" space, and answers to query by returning the documents whose concepts are more similar to the concepts of the query. We propose an "approximated" LSI, in the sense that we return the documents that are *more pertinent to the concepts behind the query.* To do this we project the query in the reduced concepts space; here the components of the vector are the single concepts, and we emphasize the concepts that have a high absolute value, that are the "important" concepts embedded in the query, and at the same time we set to value zero the components (concepts) that have a small (absolute) value , i.e., are not relevant to the query. The modified vector is projected back in the terms space, where now we can say that these terms are representative of the most important concepts in the query. We use usual *text matching* to retrieve documents that, as we said above, contain terms related to the concepts in the query.

1.1 Organization of the paper

The rest of the paper is organized as follows. The following section discuss related work in the literature. Section 3 recall how *LSI* works. In Section 4 we present our algorithm We conclude with experimental results (Section 5) and some final remarks are presented in Section 6.

2 Related Work

LSI has been introduced by Deerwester *et al.* in [6], extending their original work presented in [10]. We detail it in the following section. In [14] Landauer and Littman present an extension of LSI that is shown to be effective for automatic cross-language IR: the authors start from a corpus containing the same documents in different languages; this corpus serves to "train the automatic interlingual indexing system" that is then used to represent any document in one or more of the languages "known" to the system (see also [11, 12]).

Papadimitriou *et al.* in [17] presented a probabilistic model of document generation (on some topics). They formally prove, for documents generated according to their model, the effectiveness of LSI as a retrieval technique. More recently Azar *et al.* [2], starting from a more general theoretical framework than the one by Papadimitriou *et al.* , proved that LSI works in a wider range of settings and gave also theoretical justification of its effectiveness against the problem of polysemy.

Hence LSI proved, from both the theoretical and experimental point of view, to be an effective technique, but has as drawbacks the computational time needed by SVD and query answer. In the literature appeared various papers describing techniques to speed up the LSI computation. Among them, Achlioptas and McSherry [1] propose a technique to speed up the computation of low rank matrix approximation A_k. The idea is to compute the SVD not on the original term-document matrix A, but on a matrix $\widehat{A}$ that is obtained from the original A by reducing the number of non-zero entries and therefore speeding up the computation process by taking advantage of the sparse structure of the matrix. A different approach is due to Dumais [7, 8, 9]; the idea is to take a reduced set of documents as a sample of the entire collection, and to compute the SVD decomposition on the term-document matrix that we build from this reduced set of documents. This corresponds, of course, to the removal of colums (documents) and rows (terms) from the original input matrix A. A work that deals with the problem of query answer has been done by Letsche and Berry in [15], where the authors show that query answer is a highly parallelisable task that can be speeded up if multiple processors are available.

So far we have discussed only IR problems with a fixed collections of documents. This is usually not a realistic scenario, it should be possible to add and/or remove documents from the set. But if we allow the collection to vary over time, this of course changes the terms-documents matrix A and

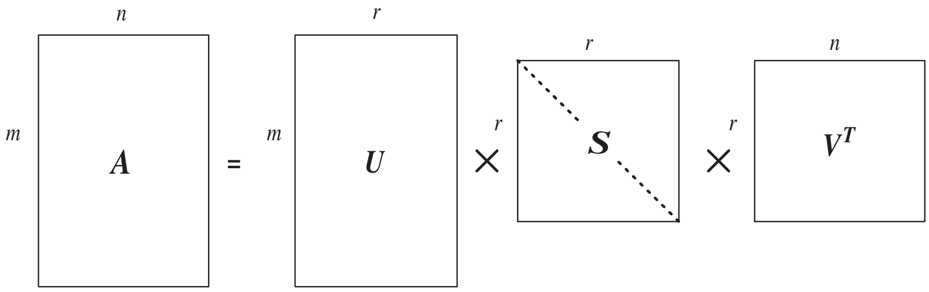

Figure 1: Singular Value Decomposition

therefore its singular value decomposition. In his Ph.D thesis O'Brien [16] considered this problem and proposed an algorithm, called SVD-updating, that does not need to compute the SVD from scratch when new documents are added. The author also proved a bound on the error given by his technique.

Up-to-date Web references related to LSI papers, software and corpora can be found in the webpages maintened by Berry [4], Landauer [13] and Telcordia Technologies, Inc. [19].

We can focus our approach among the query operations based on automatic global analysis; examples of these techniques are the similarity thesaurus proposed by Qiu and Frei [18], the statistical thesaurus presented by Crouch and Yang in [5] and the clustering techniques used by Vorhees [20].

3 Preliminaries

Consider a collection of heterogenous documents. Every document can be seen as a collection of the terms that appear in the document. The *term-document* matrix A is the $n \times m$ matrix in which the rows represent terms and colums represent documents.

Information retrieval via LSI works essentially in four phases:

- Offline phases, in which the documents (and A) are processed:
    1. The *Singular Value Decomposition* (SVD) is applied to the matrix A which is expressed as the product of three distinct matrices $A = USV^T$ (see fig.1).
    2. The matrix A is approximated with A_k (see fig.2).
- Online phases, that take place every time a user presents a query q:
    3. The query q is considered a pseudo-document $\tilde{D}^q$;
    4. The closest documents are returned to the user.

We now explain the four phases in detail.

Phase 1: SVD. SVD is a technique used in linear algebra to estimate the rank and the correlation of a matrix. Eigenvalues are not defined for a rectangular matrix. In this case it is possible to define some sort of generalized eigenvalues: the *singular values*.

Consider the matrix $A_{(m\times n)}$ (we assume, without loss of generality, that $m > n$), whose rank is r; then, the square matrix $C_{(m\times m)} = A \cdot A^T$ has:

- r positive eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r > 0$;

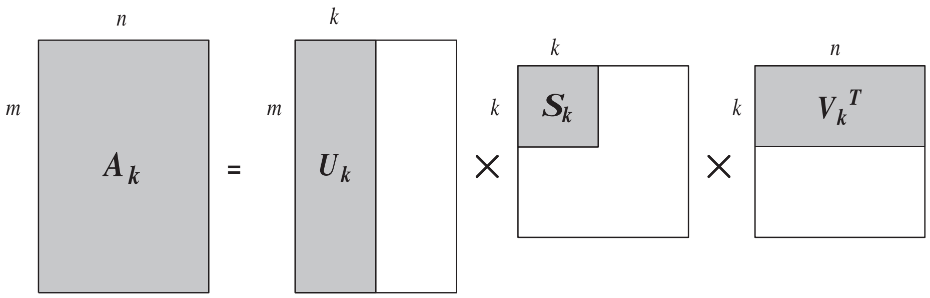

Figure 2: Approximating A with A_k

- one null eigenvalue $\lambda_0 = 0$ with multiplicity $m - r$.

The values $s_1 \geq s_2 \geq \ldots \geq s_r > 0$, obtained as $\sigma_i = \sqrt{\lambda_i}$ are, by definition, the **singular values** of matrix A.

The SVD-decomposition of A is a way to rewrite A as a product of three matrices:

$$A = U' \cdot S' \cdot V'^T$$

where:

U' is a $(m \times m)$ matrix, whose column vectors $[u_1 \ldots u_m]$ are the eigenvectors of $C = A \cdot A^T$, and therefore are orthonormal [1];

V' is a $(n \times n)$ matrix whose column vectors $[v_1 \ldots v_n]$ are the eigenvectors of $D = A^T \cdot A$, and therefore are orthonormal too.

S' is a matrix $(m \times n)$ made as follows:

$$S' = \begin{bmatrix} S & 0_{(r\times(n-r))} \\ 0_{((m-r)\times r)} & 0_{((m-r)\times(n-r))} \end{bmatrix}$$

where $S = diag(s_1, \ldots, s_n)$ is the diagonal matrix whose entries are the singular values of matrix A and $0_{(k\times l)}$ is the $(k \times l)$ matrix whose entries are all equal to 0; we note that, coherently with Figure 1 and the above formulas, U is the matrix whose columns are the first r columns of U'; similarly V is the matrix whose columns are the first r columns of V' (and therefore V^T is the matrix whose rows are the first r rows of V'^T).

Phase 2: k-rank approximation. We recall that the r elements of the diagonal of S are ordered; we consider the k largest ones; the other $(r-k)$ are fixed to zero. That means that the $(r-k)$ columns of U and V are not considered, as in fig. 2)
So we define the rank k matrix A_k $(m \times n)$ as follows:

$$A_k = U_k S_k V_k^T$$

where:

[1] A square matrix M is orthonormal if it is regular ($\det M \neq 0$) and if $M^T = M^{-1}$; so, for a orthonormal matrix M it holds $M^{-1} \cdot M = M^T \cdot M = I$.
Also $\det M = \pm 1$.

- U_k is the $m \times k$ matrix made by the first k columns of U;
- V_k is the $n \times k$ matrix made by the first k columns of V (therefore V_k^T is made by the first k rows of V^T);
- S_k is the diagonal matrix $k \times k$ whose elements are the k largest singular values.

How good is the approximation? The following, well-known theorem answers this question:

Theorem 3.1 (Eckart and Young) *Among all the $(m \times n)$ matrices M of rank at most k, A_k is the one that minimizes $\| A - M \|_F^2 = \sum_{i,j}(A_{i,j} - M_{i,j})^2$.*

From another point of view, this means that the columns of A (that are our documents), are projected in the k-dimensional space that is spanned by the column vectors of U_k. Every document of our corpus is mapped into a vector in this (reduced) space, where we can easily compute distances between documents in this reduced space. The most commonly used metric to measure distances is the cosine of the angle between two vectors (documents).

Phase 3: Projecting the query in the k-dimensional space. The query q made by the user is, again, a collection of terms and we can represent it as a $(m \times 1)$ vector q_v. As all the documents in our corpus, we can project it onto the k-dimensional space:

$$\underbrace{q}_{k \times 1} = \underbrace{S_k^{-1}}_{k \times k} \cdot \underbrace{U_k^T}_{k \times m} \cdot \underbrace{q_v}_{m \times 1} \tag{1}$$

Phase 4: Returning documents. Once that we have a representation of the query q in our k-dimensional space (q_k), we compute the distances between the query and all the documents, returning the closest ones. As we said before, we do this by computing the cosine of the angle between query and all the documents.

The last phase is probably the weak point of LSI (from the point of view of speed): the fact that we have to compare the query with *all* the documents in the collection. This means, of course, that we can't return the closest documents in constant time, as we would like to do in online applications like a search engine available via the web.

4 Our work: approximated LSI computation

In this section we propose a novel approach that is based on what we call "approximated" LSI computation. We recall that LSI works by projecting your query in the reduced "concepts" space, where the documents that are similar (close) to your query are returned; in our system we do an "approximated" version of this, because we return the documents that contain words that are related to the concepts expressed by your query.

To be more precise: we take the query q and project it into the reduced ("concepts") space computed by LSI; the vector q' that we obtain is representative of the (latent) concepts of the query; here this vector is modified in the following way: we cut away the minor concepts, i.e., the concepts whose components have a small absolute value and we emphasize the major concepts; the modified vector we obtain is projected back in the terms space, and these are the (weighted) terms that are the counterimage of the concepts of the query. We then use *text matching* to retrieve the documents.

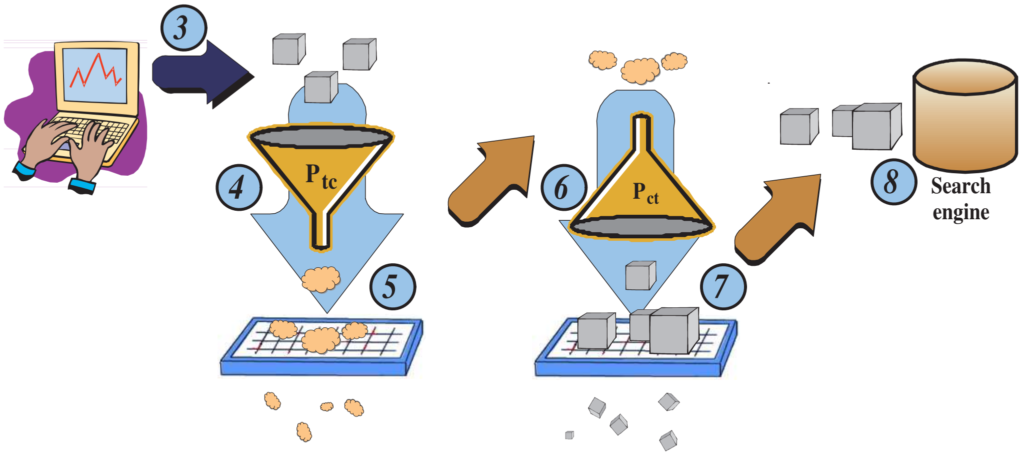

Figure 3: A schematic view of the algorithm.

4.1 The algorithm

Our approach can be summarized in the following phases:

- Offline phases, in which the documents (and A) are processed:

  1. It applies the *Singular Value Decomposition* to the matrix A which is expressed as the product of three distinct matrices $A = USV^T$ (see fig.1).
  2. It approximates the matrix A with A_k (see fig.2).

- Online phases, every time a user presents a query q:

  3. The query q is mapped into a vector $\vec{q}$ in the term space;
  4. The query vector $\vec{q}$ is projected in the reduced space to obtain $\vec{r}$;
  5. From $\vec{r}$ we calculate the vector $\vec{r}'$ by filtering the components that are below the *concepts cutoff value* c_{cv};
  6. The vector $\vec{r}'$ is then mapped into the vector $\vec{q}'$ in the m-dimensional space of terms;
  7. The minor components from the vector $\vec{q}'$ are filtered out by removing the ones that are below the *terms cutoff value* t_{cv} and we get $\vec{q}''$;
  8. We use $\vec{q}''$ to retrieve documents;

Please note that the offline phases (number one and two) are exactly the same as in *LSI* (see section 3). Now we detail the online phases:

Phase 3. In the more general case our query q will be a set of weighted terms. We can express it as a vector $\vec{q} = (q_1, q_2, \ldots, q_m)$ in the m dimensional space of terms (TVS - term vector space) defined as

$$q_j = \begin{cases} \overline{w} & \text{if the } j\text{-term is in } q \text{ with weight } \overline{w} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Phase 4. We now project $\vec{q}$ in the concepts k-dimensional space. We use the transformation matrix $P_{tc} = \Sigma_k^{-1} \cdot U_k^T$ to obtain the vector

$$\underbrace{\vec{r}}_{(k\times 1)} = \underbrace{P_{tc}}_{(k\times m)} \cdot \underbrace{\vec{q}}_{(m\times 1)} = \underbrace{\Sigma_k^{-1}}_{(k\times k)} \cdot \underbrace{U_k^T}_{(k\times m)} \cdot \underbrace{\vec{q}}_{(m\times 1)} \tag{3}$$

that represents the query q in the concepts space.

Phase 5. In this phase we filter out all the minor components of $\vec{r}$. Our aim is to get an idea of the concepts that are more relevant for the query q, thus we filter out the "noise" given by the minor components. More formally, let $c_{cv} \in (0,1)$ be the *concepts cutoff value*, the components of the filtered vector $\vec{r}'$ are given by

$$r'_i = \begin{cases} r_i & \text{if} |r_i| \geq \max_j |r_j| \cdot t_c \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Phase 6. In this phase we project back the r' to the terms space to get the vector $\vec{q}'$.

$$\vec{q}' = P_{ct} \cdot r' \tag{5}$$

where $P_{ct} = P_{tc}^{-1} = U_k \cdot \Sigma_k$.

Phase 7. Analogously as phase 4, the minor components of the vector $\vec{q}'$ are filtered away. Let $t_{cv} \in (0,1)$ be the *terms cutoff value*, then the components of the vector $\vec{q}''$ are given by

$$q''_i = \begin{cases} q'_i & \text{if} |q'_i| \geq \max_j |q'_j| \cdot t_t \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Phase 8. In this phase we use our transformed query vector $\vec{q}''$ to retrieve documents. The usual strategies from IR can be applied now to retrieve documents.

We note that while the process done in phase 4 is aimed to improve our retrieval effectiveness by leaving only the relevant concepts in the query, the one in phase 6 is done in order to speedup the retrieval process, since the terms that have little weights can slow down the process, because there can be many of them.

5 Experimental results

In this section we present the results of the experimental study. We compared the behavior of the following strategies:

- *TM* The simple text matching;
- *AL* Our query expansion strategy based on the approximated LSI computation as seen in previous section;
- *LSI* The full *LSI* computation as described in section 3;

Our data set are 3 books from 0'Reilly (www.oreilly.com). They are downloadable for free (in HTML version) and are an interesting (and difficult) dataset because they are quite specific (they all are related with computer science), and the correlation between them is high if compared to more heterogeneous collections of documents. We considered every HTML page as different documents; of course we removed hyperlinks between pages. The total number of documents was more than 3000. Of those 3000 documents, 1500 were from the first book, 1000 from the second and the remaining 500 was from the last one.

To measure the retrieval effectiveness of the algorithms we have used the usual precision at 11 standard recall levels (see [3] for a more detailed description). We recall some definitions briefly.

Consider a document collection C, and a request of information I (usually is a query q from the user of the system). Let R be the set of *relevant* documents related to I, and let $|R|$ be the number

User Query	Weighted Terms in $\vec{q}''$			
java beans	amp :	0.071578	bean :	0.543920
	java :	0.581357	jdk :	0.249543
	job :	0.071189	nbsp :	0.085341
	program :	0.062061	property :	0.094435
	value :	0.058189		
job control	background :	0.129848	control :	0.069291
	ctrl :	0.068826	echo :	0.107537
	filename :	0.058268	job :	0.427626
	line :	0.058270	list :	0.078579
	number :	0.099742	object :	0.066689
shell logout	bourne :	0.118177	login :	0.094966
	perl :	0.091234	prompt :	0.098514
	shell :	0.687393		
zip file	checksum :	0.097757	file :	0.715479
	util :	0.162579	zip :	0.150560

Table 1: Examples of query and returned terms

of documents in this set. The system (that we are evaluating) returns the set A.
Let $Ra = R \cap A$ be the intersection of the sets A and R, i.e. let Ra be the set of the relevant documents in the answer given by the system. The recall and precision measures are defined as follows:

- **Recall** is the fraction of the relevant documents (the set R) which has been retrieved,

$$Recall = \frac{|Ra|}{|R|}$$

- **Precision** is the fraction of the retrieved documents (the set R) which is relevant,

$$Precision = \frac{|Ra|}{|A|}$$

Terms and Concepts

To get an idea of how our approximated LSI works, in Table 1 we can see some examples of queries and returned terms. We notice that in some cases the system works as we would like to, for example when it adds the terms "bourne" and "prompt" to the query "shell logout"; sometimes, however, it adds misleading terms, like "ctrl" to the query "job control". We also notice that in our system the weight of the terms in the original query can be less than the one of the added terms; see for example the weight of "control" in the query q="job control".

Retrieval and time performance

The average precision at eleven standard recall levels plot is shown in Figure 4; we note that our AL system performs better than TM, and LSI outperforms both. But, as we can see in Figure 5, LSI is much slower than TM and AL. Here it is shown the time (in seconds) needed by each algorithm to return an answer to a query q from a user. We notice that there are two different y-axis; the left one, whose range is from 0 to 60, measures the LSI while the right one, ranged 0 to 6, measures TM and AL. We see that TM and AL, as we expected, are much faster than LSI. We note that our IR system has been implemented in order to study and evaluate different techniques; therefore it is not a high performance IR system the answer time can be reduced. The plot confirms that LSI has an answer time linear in the number of documents in the collection, while both TM and AL have a constant answer time.

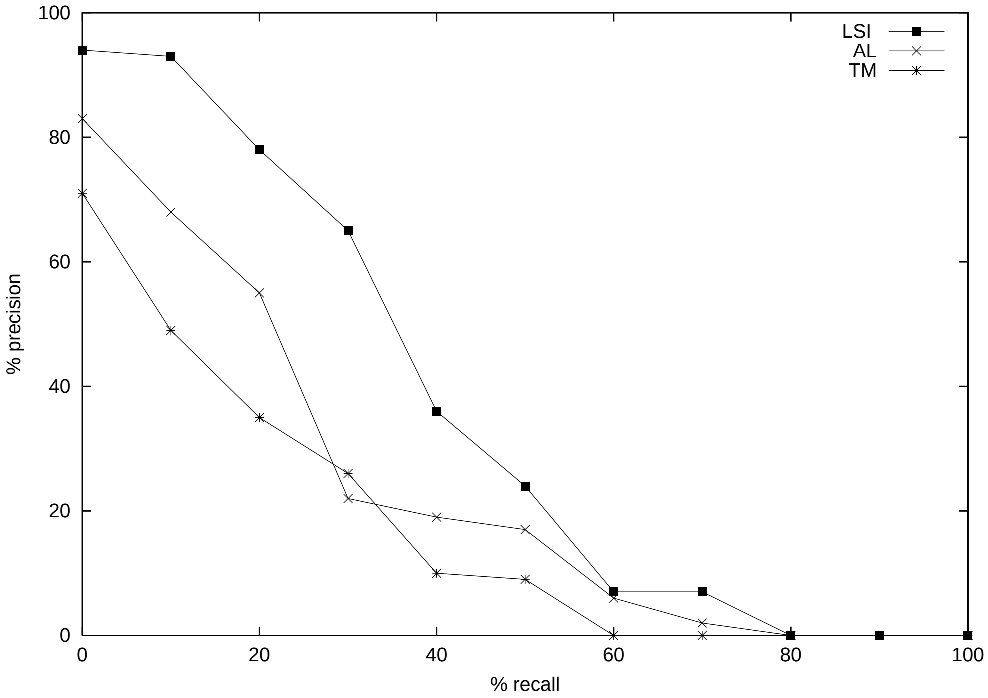

Figure 4: Average precision at eleven standard recall levels

Time Performance

Choosing the number of dimensions k

We do not report plots for the different choices of k, the number of dimensions, because these plots look very similar to each other. Further investigations are needed to understand whether it is a natural consequence of the "approximation" technique or can be due to the particular dataset. The results shown are with $k = 70$.

6 Conclusion and further works

At the moment we are performing new experimental tests over various data collections. The results show the possibilities of this new technique, but also remark its twofold behaviour. In cases where the query terms correctly express the concept behind the query, this algorithm is able to outline

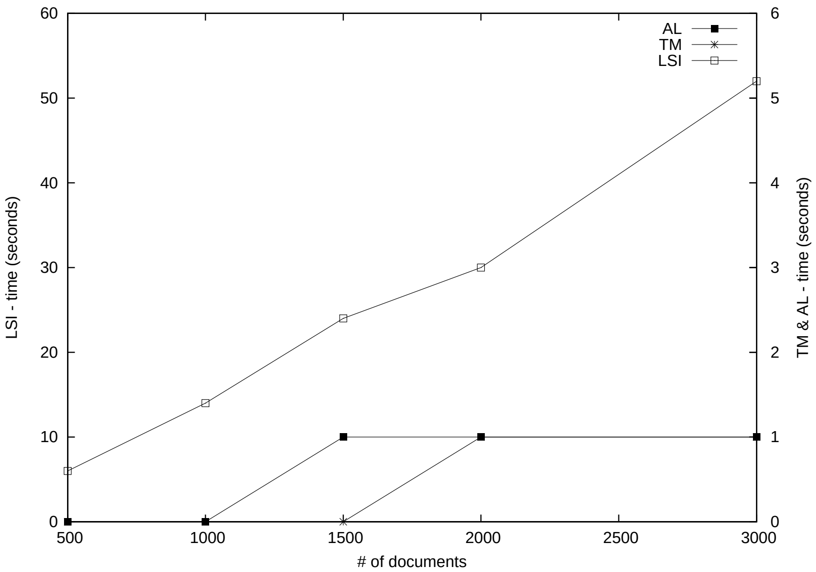

Figure 5: Time comparison chart – left y-axis is for LSI, right y-axis for TM and AL

and discard concepts of no interest, providing more suited terms for its search.
Sometimes (very often) query terms are still ambiguous and don't catch the real concept of interest. In these cases, we are not able to provide a correct retrieval; furthermore, the concept filtering can cut-off the concept of interest (with no high ranking), providing absolutely incorrect terms. The performances, in these cases, are vary bad, worse than simple text-matching.
That's why in some cases the average graphics show for out algorithm performances comparable to simple text-matching search.

This technique could be more effective if an appropriate user relevance feedback helps to discriminate relevant concepts in cases of ambiguous queries.
We are studying a new version of the system, able to select from the collection a set of terms with high discriminating power above concepts. In this way, every retrieved document is presented with all the discriminating terms it contains. Furthermore, everytime there is an abiguity in the query above two or more concepts, the system is able to prompt the user for selection (/deselection) of the discriminating terms associated with ambiguous concepts.
By selecting (/deselecting) one or more of these terms, or one or more of the retrieved documents, the user implicitly choices the concept of interest, allowing this way a more precise retrieval.

References

[1] Achlioptas, D., and McSherry, F. Fast computation of low rank matrix approximations. In *STOC: ACM Symposium on Theory of Computing (STOC)* (2001).

[2] Azar, Y., Fiat, A., Karlin, A., McSherry, F., and Saia, J. Spectral analysis of data. In *Proc. of STOC 2001* (2001).

[3] Baeza-Yates, R., and Ribeiro-Neto, R. *Modern Information Retrieval.* Addison Wesley, 1999.

[4] Mike berry's lsi page at tennessee. url="http://www.cs.utk.edu/ lsi/".

[5] Crouch, C., and Yang, B. Experiments in automatical statistical thesaurus construction. In *Proc. of the 15th ACM SIGIR Conference on Research and Development in Information Retrieval* (1992).

[6] Deerwester, S., Dumais, S., Landauer, T., Furnas, G., and Harshman, R. Indexing by latent semantic analysis. *J.Soc.Inform.Sci. 41*, 6 (1990), 391–407.

[7] Dumais, S. Lsi meets trec: A status report. In *The First Text REtrieval Conference (TREC1)* (1993), pp. 137–152.

[8] Dumais, S. Latent semantic indexing (lsi) and trec-2. In *The Second Text REtrieval Conference (TREC2)* (1994), pp. 105–116.

[9] Dumais, S. Using lsi for information filtering: Trec-3 experiments. In *The Third Text REtrieval Conference (TREC3)* (1995).

[10] Dumais, S., Furnas, G., Landauer, T., and Deerwester, S. Using latent semantic analysis improve information retrieval. In *Proceedings of CHI'88: Conference Human Factors in Computing* (1988), pp. 281–285.

[11] Dumais, S., Landauer, T., and Littman, M. Automatic cross-linguistic information retrieval using latent semantic indexing. In *SIGIR'96 - Workshop on Cross-Linguistic Information Retrieval* (1996), pp. 16–23.

[12] Dumais, S., Letsche, T., Littman, M., and Landauer, T. Automatic cross-language retrieval using latent semantic indexing. In *AAAI Spring Symposuim on Cross-Language Text and Speech Retrieval* (1997).

[13] Tom landauer's lsa page at colorado. url="http://lsa.colorado.edu/".

[14] Landauer, T., and Littman, M. Fully automatic cross-language document retrieval using latent semantic indexing. In *In Proceedings of the Sixth Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research* (1990), pp. 31–38.

[15] Letsche, T., and Berry, M. Large-scale information retrieval with latent semantic indexing. *Information Sciences - Applications*, 100 (1997), 105–137.

[16] O'Brien, G. Information managemente tools for updating an svd-encoded indexing scheme. Master's thesis, University of ennesse, Knoxville, 1994.

[17] Papadimitriou, C., Raghavan, P., Tamaki, H., and Vempala, S. Latent semantic indexing: A probabilistic analysis. *J. Comput. Systems Sciences*, 61 (2000), 217–235.

[18] Qiu, Y., and Frei, H. Improving the retrieval effectiveness by a similarity thesaurus. Tech. Rep. 225, ETH Zurich, 1994.

[19] Telcordia technologies - references to papers on lsi.
url="http://lsi.research.telcordia.com/lsi/LSIpapers.html".

[20] Vorhees, E. *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval.* PhD thesis, Cornell University, 1986.