UDC: 025.4
Preliminary communication

INFORMATION RETRIEVAL USING LATENT SEMANTIC INDEXING

Jasminka Dobša
University of Zagreb,
Faculty of Organization and Informatics, Varaždin, Croatia
jdobsa@foi.hr

Abstract: *Our capabilities for collecting and storing data of all kinds are greater then ever. On the other side analyzing, summarizing and extracting information from this data is harder than ever. That's why there is a growing need for the fast and efficient algorithms for information retrieval.In this paper we present some mathematical models based on linear algebra used to extract the relevant documents for some subjects out of a large set of text document. This is a typical problem of a search engine on the World Wide Web. We use vector space model, which is based on literal matching of terms in the documents and the queries. The vector space model is implemented by creating the term-document matrix. Literal matching of terms does not necessarily retrieve all relevant documents. Synonymy (multiple words having the same meaning) and polysemy (words having multiple meaning) are two major obstacles for efficient information retrieval. Latent Semantic Indexing represents documents by approximations and tends to cluster documents on similar topics even if their term profiles are somewhat different. This approximate representation is accomplished using a low-rank singular value decomposition (SVD) approximation of the term-document matrix. In this paper we compare the precision of information retrieval for different ranks of SVD representation of term-document matrix.*

Keywords: *information retrieval, singular value decomposition, vector space model, low-rank approximation, latent semantic indexing.*

1. INTRODUCTION

In this paper we present some mathematical models based on linear algebra used to extract relevant documents for some subjects out of a large set of text documents. More specifically, we want to match user information requests, or queries, with relevant information items, or documents. This is a typical problem of a search engine on the World Wide Web.

In the first section, the vector space model for information retrieval is presented [3,14]. The SMART (System for the Mechanical Analysis and Retrieval of Text) introduced in 1983[3], was one of the first to use vector space model. The vector space model is implemented by creating term-document matrix. The first step to create term-document matrix is to create a list of relevant terms for used collection. The term-document matrix is a $m \times n$ matrix, where m is number of terms and n is number of documents. It is formed

on the base of frequency of terms from the list in a specific document. We can create few types of the term-document matrix for the same collection of documents according to weight formulas we will use. On the other hand queries are represented as m-dimensional vectors and a matrix-vector product produces a n-dimensional vector of scores that is used to rank documents in relevance.

The newer method of Latent semantic indexing (LSI) is a variant of the vector space model and it is presented in the second section. LSI has been introduced in 1990 [6] and improved in 1995 [5]. In the vector space model similarity between query and document is measured using just the information about term matching between them. Literal matching of terms does not necessarily retrieve all relevant documents. Synonymy (multiple words having the same meaning) and polysemy (words having multiple meaning) are two major obstacles for efficient information retrieval. Latent Semantic Indexing represents documents by approximations and tends to cluster documents on similar topics even if their term profiles are somewhat different. This approximate representation is accomplished using a low-rank singular value decomposition (SVD) approximation of the term-document matrix. The approximation results in great savings in storage and retrieval time and improvement in information retrieval [6,8]. Probabilistic analysis of effectiveness of Latent semantic indexing is provided in [2,7,13,15].

In the third section, an application of LSI on standard text collection MEDLINE is given. A precision of the retrieval is compared for term matching vector space model and LSI method for different weight formulas and different ranks of SVD approximation.

2. THE VECTOR SPACE MODEL

The vector space model is implemented by creating the term-document matrix and a vector of query. Let the list of terms be numbered from 1 to m and documents be numbered from 1 to n. The term-document matrix is $m \times n$ matrix $A = [a_{ij}]$ where a_{ij} represents the weight of term i in document j. The simplest case is to set that a_{ij} is frequency of the term i in the document j. In the general case, the term weight has tree components: local, global and normalization, so we can write

$$a_{ij} = g_i t_{ij} d_j \qquad (1)$$

where tij is the term component (based on information in the j th document only), g_i is the global component (based on information about the use of the i th term thought the collection), and d_j is the normalization component. Formulas commonly used for local, global and normalization components are given in Tables 1-3. χ represents the signum function:

$$\chi(t) = \begin{cases} 1 & \text{if } t > 0, \\ 0 & \text{if } t = 0, \\ -1 & \text{if } t < 0. \end{cases} \qquad (2)$$

The weight formula is specified by three letters where the first letter represents local component, the second global component and the third normalization component.

Example 2.1 Following the notation specified in Tables 1-3 lets calculate the weight of term i in document j for formula cxn, where c specifies augmented normalized term

frequency (Table 1), letter x specifies that there no global weights (Table 2) and letter n specifies a column normalization (Table 3). So weight of term i in document j is

$$a_{ij} = \frac{0.5\chi(f_{ij}) + 0.5\left(\frac{f_{ij}}{\max_k f_{kj}}\right)}{\sqrt{\sum_{k=1}^{m}\left(0.5\chi(f_{ij}) + 0.5\left(\frac{f_{ij}}{\max_k f_{kj}}\right)\right)^2}} \qquad (3)$$

From numerical point of view it is important that the term-document matrix is sparse, that means that most of its entries are zero. That is because a document usually contains only few terms from the list of terms.

Table 1: Local term weight formulas

Symbol	Formula for t_{ij}	Description
b	$\chi(f_{ij})$	Binary
t	f_{ij}	Term frequency
c	$0.5\chi(f_{ij}) + 0.5\left(\frac{f_{ij}}{\max_k f_{kj}}\right)$	Augmented normalized term frequency
l	$\log(f_{ij}+1)$	Logarithmic

Table 2: Global term weight formulas

Symbol	Formula for g_i	Description
x	1	No change
f	$\log\left(\frac{n}{\sum_j \chi(f_{ij})}\right)$	Inverse document frequency (IDF)
p	$\log\left(\frac{n-\sum_j \chi(f_{ij})}{\sum_j \chi(f_{ij})}\right)$	Probabilistic inverse

Table 3: Normalization formulas

Symbol	Formula for d_j	Description
x	1	No normalization
n	$\left(\sum_{i=1}^{m}\left(g_i t_{ij}\right)^2\right)^{-\frac{1}{2}}$	Normal

In the term-document matrix the documents are represented as a columns. A query has the same shape as a document; it is a vector, which on i th place has the weight of the i th term. Let query be $q = [q_i]$. We will also specify term weighting for query. Here

$$q_i = g_i \hat{t}_i, \tag{4}$$

where g_i is computed based on the frequencies of terms in the document collection by one of three formulas from Table 2, and $\hat{t}_i$ is computed using the same formulas as for t_{ij} given in Table 1 with f_{ij} replaced by $\hat{f}_i$, the frequency of term i in the query. The normalizing component of query weighting is always x. We never normalize the vector of query because it has no effect on document ranking.

Example 2.2 In the weight formula for query tpx letter t means that local weight of terms in documents is simply term frequency, letter p means global component is probabilistic inverse, and letter x means that there is no normalizing. So,

$$q_i = \log\left(\frac{n - \sum_j \chi(f_{ij})}{\sum_j \chi(f_{ij})}\right)\hat{f}_i \tag{5}$$

The weight formula cxn.tpx means cxn term weighting for documents and tpx term weighting for queries.

A common measure of similarity between the query and the document is the cosine of the angle between them. Let a_j, $j = 1, 2, ..., n$ be the columns in the term-document matrix. They represent the documents. Then we calculate similarity between query q and document a_j using a formula

$$\cos\theta_j = \frac{a_j^T q}{\|a_j\|_2 \|q\|_2} \tag{6}$$

If we use column normalization in term weighting then dividing by $\|a_j\|_2$ in formula (6) doesn't have any impact because vectors of documents are of unit norm. Further, dividing by $\|q\|_2$ doesn't have impact on ranking of documents according their relevance to the query. In order to rank documents, we compute

$$s = q^T A, \tag{7}$$

where the j th entry in s represents the score of document j.

3. THE LSI METHOD

Theoretical results supporting LSI are given in [4,9,10]. For any matrix A exists singular value decomposition

$$A = U\Sigma V^T \tag{8}$$

where U is the $m \times m$ orthogonal matrix, V is the $n \times n$ orthogonal matrix and Σ is $m \times n$ diagonal matrix

$$\Sigma = \text{diag}(\sigma_1, ..., \sigma_p), \qquad (9)$$

where $p = \min\{m, n\}$ and $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_p \geq 0$. The σ_i are the singular values and u_i and v_i are the i th left singular vector and the i th right singular vector respectively. Let $\text{r}(A)$ be the rank of matrix A. From the orthogonal invariance of rank it follows that

$$\text{r}(A) = \text{r}(U\Sigma V^T) = \text{r}(\Sigma), \qquad (10)$$

so the rank of the matrix A is equal to the number of its nonzero singular values.

Theorem by Eckart and Young [9] states that the distance in Frobenious norm between A and its k-rank approximation is minimized by the approximation A_k. Here

$$A_k = U_k \Sigma_k V_k^T, \qquad (11)$$

where U_k is $m \times k$ matrix whose columns are the first k columns of U, V_k is the $n \times k$ matrix whose columns are the first k columns of V, and Σ_k is $k \times k$ diagonal matrix whose diagonal elements are the k largest singular values of A. More precisely,

$$\|A - A_k\|_F = \min_{\text{rank}(X) \leq k} \|A - X\|_F = \sqrt{\sigma_{k+1}^2 + ... + \sigma_{r(A)}^2}. \qquad (12)$$

So, matrix A_k is the best k-rank approximations for term-document matrix A.

Algorithm for LSI is as follows: first we create the term-document matrix and the vectors of queries and then we use the singular value decomposition on the term-document matrix and create a k-rank approximation. We rank the documents according to their relevance to the queries by calculating the score vector $s = \left(q^T U_k\right)\left(\Sigma_k V_k^T\right)$.

In score vector a query is presented as $q^T U_k$ and documents are presented in $k \times n$ matrix $\Sigma_k V_k^T$. As $k << m$ (where k is rank of approximation and m is number of terms) storage of documents using LSI needs much less memory space than storage of documents using ordinary vector space model.

Table 4 : Comparison: Straightforward term matching method and LSI method (rank -30 and rank -50 approximation) for different term weighting

Weight	Mean average precision			Median average precision		
	Term matching method	LSI, k=30	LSI, k=50	Term matching method	LSI, k=30	LSI,k=50
txn.txx	77,21	78,88	82,74	**90,91**	86,62	90,91
bxn.bxx	78,19	76,67	83,37	84,44	85,60	93,39
txn.bxx	78,22	79,25	82,90	**90,91**	86,62	89,55
lxn.bfx	79,60	85,75	86,26	89,70	93,90	94,55

lxn.lfx	80,09	85,20	85,41	89,70	93,90	92,73
lxn.tfx	79,87	85,27	86,03	89,70	93,94	94,54
lxn.bpx	80,15	85,87	85,87	**90,91**	93,90	93,39
lxnlpx	80,73	85,51	85,53	**90,91**	93,90	92,73
cxn.bfx	**80,75**	88,14	85,69	87,88	97,73	90,91
cxn.tfx	80,33	90,58	86,58	89,39	**97,98**	94,95
cfn.tfx	80,36	**90,67**	**86,69**	89,39	**97,98**	**95,96**
cxx.bpx	79,59	74,04	78,48	82,12	83,97	89,92
cxn.bpx	80,50	88,25	85,69	87,88	**97,73**	90,91
cxn.tpx	80,19	90,58	86,57	89,39	**97,98**	94,95

4. APPLICATION ON A STANDARD TEXT COLLECTION

Experiments are conducted on the part of standard text collection MEDLINE that consists of 270 medical documents (titles and abstracts). Also, the queries and relevance judgments were available. List of terms is formed by extracting all terms out of the documents and then ejecting terms that occur in only one document and terms on a stop list of 493 common words used by SMART ('and', 'the'...). Terms were not steamed or variations of words were not mapped to the same root form. The list of terms consists of 2153 terms.

For the comparison of the methods two standard measures are used: mean average precision and median average precision [1]. When we evaluate a query, we return a ranked list of documents. Let r_i denote the number of relevant documents among the top i documents and r_n total number of relevant documents in collection. The precision for the top i documents, p_i, is then defined as

$$p_i = \frac{r_i}{i}, \qquad (13)$$

i.e., the fraction of the retrieved i documents that are relevant. The recall is defined as r_i/r_n, i.e. the fraction of relevant documents that has been retrieved.

The N - point (interpolated) average precision is defined as

$$\frac{1}{N}\sum_{i=0}^{N-1}\tilde{p}\left(\frac{i}{N-1}\right) \qquad (14)$$

where

$$\tilde{p}(x) = \max_{\frac{r_i}{r_n}\geq x} p_i \qquad (15)$$

is pseudoprecision at recall level $x \in [0,1]$.

Here 11-point interpolated average precision is used (recall levels $0, 0.1, \ldots, 1$). We have 17 queries, so we compute mean average precision and median average precision, expressed as percentages.

In the Table 4 we compare straightforward term matching vector space method and LSI method for different term weighting. In Tables 5,6 and 7 comparisons of LSI method for different k-rank approximations and fixed term weighting are done.

All the experiments are done in MATLAB 5.0.

5. CONCLUSIONS AND DISCUSSION

From the Table 4 we can see that median average precision is usually higher then mean average precision. That means that most of the queries have good average precision (more than 90%), but there are some queries with match the worst average precision. However, the results for mean average precision are more stable. That's way we will concentrate in our discussion on mean average precision, which is more standard measure in information retrieval. The best performances for every method (term matching, LSI $k=30$, LSI $k=50$) in Table 4 are bolded.

It is obviously that LSI has better performance than straightforward term matching method. Using LSI method, documents are projected in means of least squares on space spread by first k left singular vectors of term-document matrix. First k components capture the major associational structure in the term-document matrix and throw out the noise. That's way minor differences in terminology used in documents are ignored and closeness of objects (queries and documents) is determined by the overall pattern of term usage, so it is context based. LSI method addresses the problems of synonymy and polysemy: documents which contain synonyms are closer after projection than in original space; documents which contain polysemy in different context are farther after the projection than in original space.

We can see that term weighting has influence on choice of the best k-rank approximation. Greatest influence has first letter of term weighting, i.e. the term component. If the term component is t, i.e. term frequency, then the performance is the best for LSI method, $k=50$. For the logarithmic term component the best performance is again for LSI method, $k=50$, but improvement in relation to LSI method, $k=30$ is smaller then before. For augmented normalized term frequency, we can notice a drop of mean average precision for LSI method, $k=50$ in relation to LSI method, $k=30$. In Tables 5,6 and 7 we can see the effect of raising the rank of term-document matrix approximation for different term weightings. The best performances for every weight are bolded. Generally, the best performance is achieved for ranks of approximation of 30-40. We can notice that for higher ranks of approximation mean average precision is falling down, so is makes sense to choose the rank of approximation between 30 and 40 for this collection. As the total number of relevant terms here was 2153 and the total number of documents 270, the original term-document matrix was stored in 2153×270 matrix. If we chose the rank of approximation to be 40, then SVD approximations of documents a stored in 40×270 matrix which is a great saving of memory space.

Table 5: LSI method for cfn.tfx and cxn.tpx term weighting

K	mean average precision		median average precision	
	cfn.tfx	cxn.tpx	cfn.tfx	cxn.tpx
10	70,75	70,93	81,82	85,45
15	80,80	81,02	90,91	90,91
20	78,13	78,83	90,91	90,91
25	86,43	86,56	95,96	95,96
30	**90,67**	**90,58**	**97,98**	**97,98**
35	89,11	89,18	96,59	96,59
40	89,25	89,20	95,96	96,59
45	86,61	89,55	95,96	97,73
50	86,69	86,59	95,96	94,95
55	85,80	84,79	96,59	94,95
60	84,71	86,32	96,59	96,59
65	83,64	83,56	96,59	96,59
70	83,88	84,08	96,59	96,59

Table 6: LSI method for lxn.bpx and lxn.lfx term weighting

k	mean average precision		median average precision	
	lxn.bpx	lxn.lfx	lxn.bpx	lxn.lfx
10	73,05	73,08	87,88	90,76
15	77,23	77,13	92,09	91,12
20	83,53	85,13	**96,59**	**96,59**
25	84,66	84,37	94,66	94,66
30	85,87	85,20	93,90	93,90
35	87,01	86,88	92,73	94,55
40	**87,85**	87,73	93,39	94,66
45	87,19	**87,93**	94,66	95,96
50	85,87	85,41	93,39	92,73
55	86,10	85,18	93,94	92,73
60	85,63	85,02	93,94	93,18
65	85,33	85,28	92,27	94,81
70	85,66	85,28	94,95	94,81

Table 7: LSI method for txn.txx term weighting

k	mean average precision	median average precision
10	60,03	83,33
15	73,37	83,84
20	77,88	86,36
25	77,51	87,73
30	77,21	90,91
35	83,08	86,36
40	**83,80**	86,36
45	82,26	87,12
50	82,74	90,91
55	80,94	90,91
60	80,16	89,70
65	80,31	89,61
70	82,12	**90,63**

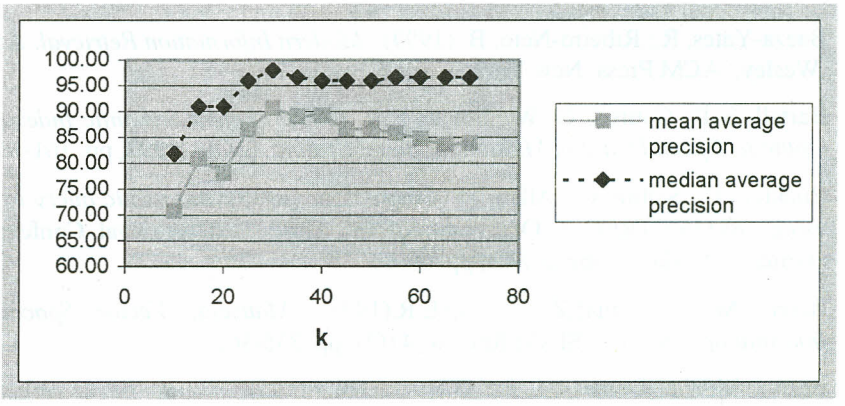

Figure 1: Mean average precision and median average precision for cfx.tfx weighting

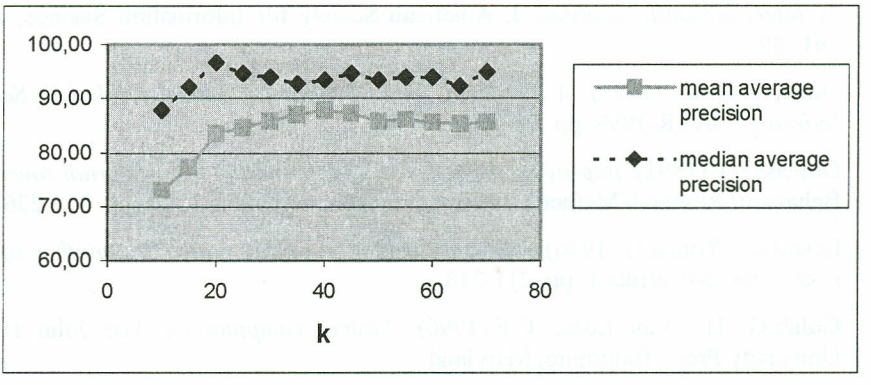

Figure 2: Mean average precision and median average precision for lxn.bpx weighting

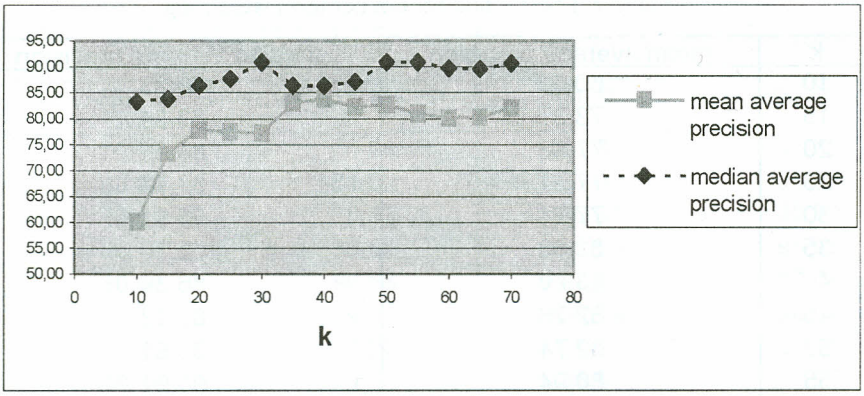

Figure 3: Mean average precision and median average precision for txn.txx weighting

REFERENCES

[1] Baeza-Yates, R.; Ribeiro-Neto, B. (1999) : *Modern Information Retrieval,* Addison-Wesley, ACM Press, New York

[2] Bartell, B.T.; Cottrell, G. W.; Belew, R.K. (1992): *Latent Semantic Indexing is an Optimal Special Case of Multidimensional Scaling*, SIGIR-1992, pp. 161-167.

[3] Buckley, C.; Salton, G.; Allan, J.; Singhanl,A. (1995): *Automatic query expansion using SMART:TREC 3,* Overview of the Third Text Retrieval Conference, D. Harman, ed., Gaithersburg, MD, pp. 69-80

[4] Berry, M.W.; Drmač,Z.; Jessup,E.R.(1999): *Matrices, Vector Spaces and Information Retrieval,* SIAM Review, 41(2), pp. 335-362.

[5] Berry, M.W.; Dumas, S.T.; O'Brien,G.W.(1995): *Using linear algebra for intelligent information retrieval,* SIAM Rewiev, 37(4), pp. 573-595.

[6] Deerwester,S.; Dumas, S.; Furnas, G. ; Landauer, T.; Harsman,R. (1990): *Indexing by latent semantic analysis,* J. American Society for Information Science, 41, pp. 391- 407

[7] Ding, C. H. Q. (1999): *A Similarity-based Probability Model for Latent Semantic Indexing*, SIGIR-1999, pp. 58-65.

[8] Dumas, S.T.(1991): *Improving the retrieval of information from external sources,* Behaviour Research Methods, Instruments and Computers, 23(2), pp. 229-236.

[9] Eckart,C.; Young,G.(1936): *The approximation of one matrix by another of lower rank,* Psychometrika,1, pp. 211-218.

[10] Golub,G. H.; Van Loan, C.F.(1996): *Matrix computation,* The John Hopkins University Press, Baltimore, Maryland

[11] Kolda,T.; O'Leary,D.(1998): *A semi-discrete matrix decomposition for latent semantic indexing in information retrieval,* ACM Trans. Inform. Systems, 16, pp. 322-346.

[12] MEDLINE(1998) , ftp>//ftp.cs.cornell.edu/pub/smart/med/

[13] Papadimitriou, C.; Raghavan, P.; Tamaki, H.; Vempala, S.(1998): *Latent sematnic Indexing: Probabilistic Analysis,* Procedings of Seventeenth ACM-SIGACT-SIGMOD-SIGARD Symp. Principles of Database Systems, Seattle, Washington, pp. 159-168.

[14] Salton,G.; McGlill, M. (1983): *Introduction to Modern Information Retrieval,* McGraw- Hill, New York

[15] Story, R. E.(1996): *An Explanation of the Effectiveness of Latent Semantic Indexing by Means of a Bayesian Regression Model*, Information Processing & Management, 2(3), pp. 329-344

Received: 9 January 2003
Accepted: 9 October 2003