

Straightforward Feature Selection for Scalable Latent Semantic Indexing

Jun Yan¹ Shuicheng Yan² Ning Liu¹ Zheng Chen¹

¹Microsoft Research Asia, Sigma Center, 49 Zhichun Road, Beijing, China, 100080
{junyan, ningl, zhengc}@microsoft.com

²Department of Electrical and Computer Engineering, National University of Singapore, Singapore
eleyans@nus.edu.sg

Abstract

Latent Semantic Indexing (LSI) has been validated to be effective on many small scale text collections. However, little evidence has shown its effectiveness on unsampled large scale text corpus due to its high computational complexity. In this paper, we propose a straightforward feature selection strategy, which is named as Feature Selection for Latent Semantic Indexing (FSLSI), as a preprocessing step such that LSI can be efficiently approximated on large scale text corpus. We formulate LSI as a continuous optimization problem and propose to optimize its objective function in terms of discrete optimization, which leads to the FSLSI algorithm. We show that the closed form solution of this optimization is as simple as scoring each feature by Frobenius norm and filter out the ones with small scores. Theoretical analysis guarantees the loss of the features filtered out by FSLSI algorithm is minimized for approximating LSI. Thus we offer a general way for studying and applying LSI on large scale corpus. The large scale study on more than 1 million TREC documents shows the effectiveness of FSLSI in Information Retrieval (IR) tasks.

1. Introduction

Latent Semantic Indexing (LSI) [5] was originally proposed for dealing with the synonymy and polysemy problems in text analysis. It has been successfully applied to the Information Retrieval (IR) tasks on various small and middle scale text collections and its solution is based on the Singular Value Decomposition (SVD) [5]. The high computational complexity of SVD greatly limits the study of LSI on large scale text corpus. Little evidence showed the effectiveness of LSI on unsampled large scale text corpus. A recent study [13] argues that it is hard for LSI to enhance IR performance on large text collections though it generally works well on small and middle scale collections. To judge the effectiveness of LSI on large

scale text collection and make it usable if its effectiveness can be verified, a pressing task is to make the LSI computable on large scale text corpus.

In this paper, we propose a straightforward and scalable data preprocessing approach to make LSI applicable on large text collections. We preprocess the data by proposing a feature selection algorithm, which is named as Feature Selection for LSI (FSLSI), to reduce the size of feature set such that SVD only need to be computed on the much smaller scale reduced data. Theoretically, the proposed feature selection algorithm guarantees that the LSI implemented on the feature reduced dataset can closely approximate its counterpart without feature selection. Through this way, we can overcome the bottleneck of SVD and apply LSI on larger scale data than many previous studies [6, 7, 8, 9] with closely approximated solution. In this work, we name the approach to use FSLSI as data preprocessing step for approximating classical LSI as the Approximated LSI, i.e. ALSI in short.

In details, LSI aims to project documents from the term space to another much lower-dimensional latent semantic space, and in total LSI can be considered as a special linear dimensionality reduction process. We firstly discover the backend objective function of LSI and reformulate it as a continuous optimization problem under the dimensionality reduction framework [22]. Motivated by the observation that many feature selection algorithms can be generally considered as discrete optimization problems under the same dimensionality reduction framework [22], in this paper we propose to optimize the objective function of LSI in terms of discrete optimization, which consequently leads to a novel feature selection algorithm, *i.e.* the FSLSI. We show that the closed form solution of this optimization problem is as straightforward as scoring each feature by Frobenius norm and filter out the ones with small scores.

The selected features are consistent with the objective of LSI since they are selected by optimizing the same objective function as the original LSI. In addition, the dimensionality reduction framework allows us defining Energy function to help determine the number of features to be selected by FLSLI. To guarantee the loss led by the reduced features is minimized in approximating the large scale LSI, we require the bases of the latent semantic space, which are the left singular vectors calculated by SVD on the original data, have the minimum changes after the features are removed. However, the loss of the latent semantic space is only known after the SVD has been computed on the original data. In this work, we theoretically prove that, though without the real SVD computation on original data, our proposed FLSLI algorithm can filter out exactly the features which lead to the minimum loss. Thus our proposed approach is theoretically guaranteed to well approximate LSI on large scale dataset.

Experimental results on the TREC2 and TREC3 [21] show that our proposed approach (ALSI) can efficiently and closely approximate LSI in IR tasks. Thus our proposed approach offers a general way for studying and applying LSI on large scale corpus. The large scale study on TIPSTER [23] in IR task, which includes more than 1 million TREC documents and about 0.8 million terms, shows that IR performance evaluated by different metrics can be effectively improved by using ALSI. As an example, the precision at 5 can be improved 20% in contrast to the classical Cosine similarity [11] without LSI. It verifies that LSI can truly improve the IR performance on large scale text collections. This observation is different from some previous studies on large scale LSI [12]. The reason for the difference is that the reduced data by FLSLI allows us to compute more than 2,000 eigenvectors through SVD, while it is almost impossible to compute more than 1,000 eigenvectors on the same but unreduced dataset by classical LSI without *super* computers.

As a summary, the main contributions of this paper are: (1) we propose a general way, i.e. a straightforward feature selection algorithm as preprocessing of LSI, to make LSI implementable on large scale text corpus; (2) we theoretically guarantee the proposed algorithm is optimal for approximating classical LSI; and (3) we draw different conclusions from some previous large scale LSI studies through experiments. We point out that the reason why some previous LSI studies fail on the large scale data is that they cannot solve a large number of eigenvectors in SVD computation. Our proposed approach makes

solving a much larger number of eigenvectors possible without super computers.

The rest of this paper is organized as follows. In Section 2, we review the related works. Section 3 introduces the details of the algorithm for approximating LSI through reformulating it under the linear dimensionality reduction framework. The theoretical analysis is given in Section 4. In Section 5, the experimental results on large scale datasets are presented. Finally we conclude this paper and discuss the future work in Section 6.

2. Related works

Latent Semantic Indexing (LSI) was originally proposed for dealing with the problem of *synonymy* and *polysemy* [5]. A variety of tests and applications have been developed to validate its power in text representation [8]. Special interests have been paid to investigate its ability in improving the IR performance [6, 7, 8, 9]. Besides general IR tasks, LSI has also been successfully applied for the cross-language retrieval [10] and distributed information retrieval tasks [19]. However, classical LSI suffers from the high computational cost involved in the Singular Value Decomposition (SVD), especially when applied to large scale text corpus. To avoid the costly computation, it has been proposed to use other strategies such as Semi-Discrete matrix Decomposition (SDD) [16] and Concept Indexing (CI) [15] instead of LSI for text processing. However, the substituted strategies cannot judge the effectiveness of classical LSI, and throw away the advantages of LSI in text understanding.

The complexity issue of LSI may be alleviated by some traditional techniques, e.g., the parallel computation [1] and the incremental matrix decomposition [2]. However, little evidence showed the effectiveness of LSI on large scale text corpus by using these approaches since they either require special machine environment or cannot well approximate the original SVD. The SVD algorithm designed for sparse matrix [3] is widely used in LSI study. However, it is still insufficient in computing large number of eigenvectors due to its limited efficiency improvement for SVD computation. The data reduction approaches, such as document sampling, document clustering, term reduction or combination of document reduction and term reduction [6,7,8,19], were used as the preprocessing for LSI in many previous studies. They all aim to reduce the data scale before SVD computation. However, the results on the reduced data will highly depend on the specific data reduction strategy. To the extent of our knowledge, few previous

studies by data reduction have been proved be able to well approximate the original LSI. Our proposed feature reduction algorithm is proposed for this purpose and guarantees the minimum loss in approximating the original LSI.

There exist many works, such as term norms [12] and fold in [2], which can be used to further improve the performance of LSI. They can also be used to improve the algorithmic performance of our proposed approximate LSI, but it will not be the focus of this paper. To our best knowledge, the large scale study of LSI was introduced in [12], yet negative results were reported. In this paper, we will however give a contrary conclusion regarding the performance of LSI on large scale corpus through seeking for more eigenvectors in extensive studies.

3. Approximate large scale LSI

In this section, we first give a brief overview on the classical LSI and then introduce its reformulation. Then we propose our feature selection algorithm as the preprocessing step for LSI, and its objective function is the same as that for the original LSI, yet with additional discrete constraints on the projection matrix. Finally we summarize the entire algorithm for approximating classical LSI and introduce the energy function for determining the number of selected features.

3.1. Latent Semantic Indexing

In this paper, we assume all text documents and queries are represented in the classical Vector Space Model (VSM) [11] by the Term Frequency Inversed Document Frequency (TFIDF) indexing [11]. Thus a corpus of text documents are represented by a $d \times n$ term-by-document matrix $X \in R^{d \times n}$, where n is the number of documents, and d is the number of features (terms). Each document is denoted by a column vector $x_i \in R^d$, $i = 1, 2, \dots, n$. Let x_{ki} , $k = 1, 2, \dots, d$ stands for the k^{th} entry of x_i , X^T stands for the transpose of the matrix X and I stands for the identity matrix of arbitrary size.

In the classical LSI algorithm, the vector space model is improved by replacing the original term-by-document matrix with a low-rank approximation derived from the SVD. The SVD of matrix X can be described as $X = U\Sigma V^T$, where $\Sigma \in R^{d \times n}$ is a diagonal matrix. The diagonal elements of Σ , $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = 0$, are the singular values of the matrix X . The matrices $U = \{u_1, u_2, \dots, u_d\} \in R^{d \times d}$ and $V = \{v_1, v_2, \dots, v_n\} \in R^{n \times n}$ consist of the left singular vectors and right singular vectors of

matrix X respectively. Thus $UU^T = I$ and $VV^T = I$. LSI only retains the leading singular vectors, *i.e.* the first p columns of U and V (usually $p \ll d$), as well as the upper-left sub-matrix of Σ , such that $X_p = U_p \Sigma_p V_p^T$, where the matrices $U_p = \{u_1, u_2, \dots, u_p\}$, $V_p = \{v_1, v_2, \dots, v_p\}$ and $\Sigma_p = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_p)$. It is known that X_p gives an optimal rank- p approximation of the original matrix X . After SVD, LSI essentially projects a document x_i into a lower dimensional feature space as

$$x'_i = \Sigma_p^{-1} U_p^T x_i \in R^p, \quad (1)$$

where $x'_i \in R^p$ is the latent semantic indexing of the document x_i , $i = 1, 2, \dots, n$. The space spanned by the columns of U_p , *i.e.* $\text{span}\{u_1, u_2, \dots, u_p\}$ is known as the *latent semantic space*. U_p is the projection matrix to project the documents into the latent semantic space. Since Σ_p^{-1} is a diagonal matrix, it is used for rescaling the features of documents in latent semantic space. For a given query q , it is projected into the latent semantic space by,

$$q' = \Sigma_p^{-1} U_p^T q \in R^p. \quad (2)$$

In traditional IR tasks, the documents are ranked according to their similarities to the query q in the latent semantic space. A commonly used similarity measurement is the Cosine similarity,

$$\text{Sim}(q', x'_i) = \langle q', x'_i \rangle / \|q'\| \|x'_i\|, \quad (3)$$

where $\langle \cdot, \cdot \rangle$ stands for the inner product between vectors and the $\|\cdot\|$ is the Frobenius norm of a vector. Theorem-1 below reformulates the classical LSI as an optimization problem.

Theorem-1: Let $C = XX^T$, classical LSI can be reformulated as an optimization problem as

$$W^* = \text{argmax}_W \text{tr}\{W^T C W\}, \text{ s.t. } W^T W = I, \quad W \in R^{d \times p}. \quad (4)$$

Here $\text{tr}\{W^T C W\}$ indicates the summation of all the diagonal elements, namely the trace, of the matrix $W^T C W$. The proof of Theorem 1 is given in the appendix.

3.2. Feature selection for LSI

The linear dimensionality reduction problem can be generally defined as to search for an optimal linear function $f: R^d \rightarrow R^p$ (usually $p \ll d$) such that a vector $x_i \in R^d$ is projected into a lower dimensional feature space through $x'_i = f(x_i) = W^T x_i \in R^p$, where $W \in R^{d \times p}$ is the projection matrix. The traditional dimensionality reduction algorithms can be roughly

classified into two categories: feature extraction and feature selection algorithms [20]. Under the general dimensionality reduction framework [22], both categories can be formulated as to search for the optimal W according to certain objective function $J(W)$. Here, we define

$$H_{fe} = \{W \in R^{d \times p}, W^T W = I\}. \quad (5)$$

If $J_{fe}(W)$ stands for the objective function of certain feature extraction algorithm, then the feature extraction algorithm can be generally formulated as solving the optimization problem¹,

$$W^* = \operatorname{argmax}_{W \in H_{fe}} J_{fe}(W). \quad (6)$$

As some instances, the feature extraction algorithms, such as Principal Component Analysis (PCA), Linear Discriminate Analysis (LDA), and Maximum Margin Criterion (MMC), can all be formulated as solving the optimization problem in Eqn. (6) over the solution space defined in Eqn. (5). More details of these algorithms under the dimensionality reduction framework are referred to [22]. Theorem 1 shows that LSI is a feature extraction algorithm under the general dimensionality reduction framework and it can be embedded into this framework through

$$\begin{aligned} W^* &= \operatorname{argmax}_{W \in H_{fe}} J_{LSI}(W) \\ &= \operatorname{argmax}_{W \in H_{fe}} \operatorname{tr}\{W^T C W\}. \end{aligned} \quad (7)$$

On the other hand, the feature selection algorithms require $W \in R^{d \times p}$ being a binary matrix whose entries can only be either 0 or 1. In sum, the matrix should satisfy two constraints: (a) each column of W has one and only one non-zero element of 1; (b) each row of W has at most one non-zero element. Here we define the solution space for the feature selection problem as,

$$H_{fs} = \{W \in B^{d \times p}, W \text{ satisfies the constraints (a) and (b)}\}. \quad (8)$$

If $J_{fs}(W)$ stands for the objective function of certain feature selection algorithm, the feature selection problem can be generally formulated as solving the optimization problem as

$$W^* = \operatorname{argmax}_{W \in H_{fs}} J_{fs}(W). \quad (9)$$

It has been proved [22] that the commonly used text feature selection algorithms such as Information Gain (IG) and CHI are all special cases of solving the optimization problem in Eqn. (9) over the solution space in Eqn. (8) according to different objective

functions [22]. The feature extraction algorithms and feature selection algorithms are usually studied separately. However, the general dimensionality reduction framework shows that the major difference between these two categories lies in the different solution spaces. The former takes continuous real matrix as solution while the latter takes binary matrix as solution. Within each algorithm category, the objective function finally determines the algorithmic details.

In this work, we propose to optimize $J_{LSI}(W)$ in the discrete space H_{fs} instead of the continuous one H_{fe} , and consequently a novel feature selection algorithm is derived. In the dimensionality reduction framework, the novel feature selection algorithm can be formulated as solving the optimization problem,

$$W^* = \operatorname{argmax}_{W \in H_{fs}} J_{LSI}(W). \quad (10)$$

Since Eqn. (10) optimizes the same objective function as LSI does, the selected features are consistent with the objective of LSI in H_{fe} . We use it as the preprocessing of classical LSI such that we only preserve a small subset of features for the consequent SVD computation, which allows us to conduct LSI on large scale dataset. The derived algorithm to solve the problem defined in Eqn. (10) is called *Feature Selection for LSI* (FSLSI) hereafter.

Suppose $W = \{w_1, w_2, \dots, w_p\} \in H_{fs}$, then each column vector of W has one and only one nonzero element. Without loss of generality, we use k_l to represent the index of the nonzero entry in $w_l, l = 1, 2, \dots, p$. Then we have

$$\begin{aligned} J_{LSI}(W) &= \operatorname{tr}\{W^T C W\} \\ &= \sum_{l=1}^p w_l^T X X^T w_l \\ &= \sum_{l=1}^p (w_l^T X)(w_l^T X)^T = \sum_{l=1}^p \sum_{i=1}^n (x_{k_i l})^2. \end{aligned} \quad (11)$$

Defining the score of the k^{th} feature as,

$$\operatorname{Score}(k) = \sum_{i=1}^n (x_{k_i})^2, k = 1, 2, \dots, d. \quad (12)$$

The objective function in Eqn. (10) is transformed into

$$J_{LSI}(W) = \sum_{l=1}^p \operatorname{Score}(k_l). \quad (13)$$

Note that $\operatorname{Score}(k) \geq 0$ for any k , and the problem of maximizing $J_{LSI}(W)$ can be simply solved by selecting the p largest ones from $\sum_{i=1}^n (x_{k_i})^2, k = 1, 2, \dots, d$, since they naturally maximize the objective function in Eqn. (13). Without loss of generality, suppose the p selected features are indexed by k_l^* ,

¹ Since minimizing $J(W)$ equals to maximizing $-J(W)$, in this work all optimization problems are represented by maximization problems.

$l = 1, 2, \dots, p$. We can construct the matrix $W^* = (w_1^*, w_2^*, \dots, w_p^*) \in H_{fs}$ by

$$w_{kl}^* = \begin{cases} 1 & k = k_l^* \\ 0 & \text{otherwise} \end{cases}. \quad (14)$$

Then the FLSLI algorithm is to select p features with largest scores, where the scores are computed as in Eqn. (12) (squared Frobenius norm). These features guarantee to optimally maximize $J_{LSI}(W)$ in H_{fs} .

3.3. Using FLSLI to approximate LSI

As a summary, optimizing the objective function of LSI in terms of discrete optimization leads to the feature selection algorithm FLSLI. Its selected features are optimal for optimizing $J_{LSI}(W)$ in H_{fs} . We use this algorithm as the preprocessing of LSI to only retain a subset of features such that SVD can be applied on a relatively smaller scale matrix even though the original data scale is extremely large. From equation (12), we can see that the features are scored by their Frobenius norm. Thus the algorithm is as straightforward as reserve the features with large Frobenius norms.

We summarize our algorithm of approximate LSI using FLSLI in Table 1. In Step-1, each feature score requires computing the square of n real values. Thus the time complexity of FLSLI is proportional to the matrix size, *i.e.* $O(dn)$. In the Step-2, the complexity of SVD on Y is $O(p_1^3)$. Thus the total complexity of approximate LSI using FLSLI equals to $O(dn) + O(p_1^3)$. In contrast to the direct SVD approach time complexity of which is as high as $O(d^3)$, the complexity will be significantly reduced when $p_1 \ll d$.

Table 1. Algorithm for approximate LSI (ALSI) using FLSLI.

<p>Step-1 Optimize $J_{LSI}(W)$ in discrete solution space H_{fs},</p> $W_1^* = \operatorname{argmax}_{W \in H_{fs}} \operatorname{tr}\{W^T X X^T W\}.$ <p>1.1 Using Eqn. (12) to compute the feature scores; 1.2 Select the p_1 number of features with the largest scores and filter out others by $Y = W_1^{*T} X$.</p> <p>Step-2 Optimize $J_{LSI}(W)$ on reduced data Y in the continuous space H_{fe},</p> $W_2^* = \operatorname{argmax}_{W \in H_{fe}} \operatorname{tr}\{W^T Y Y^T W\}.$ <p>2.1 Calculate $W_2^* \in R^{p_1 \times p}$ by SVD on reduced Y; 2.2 Project documents to latent semantic space by $X' = W_2^{*T} Y$.</p>

3.4. Feature number selection

For the algorithm introduced in Table 1, a questionable issue is how to determine the number of features to select, namely p_1 . It is still an open

problem for many state-of-the-art feature selection algorithms. In this work, we take the additional benefit from the dimensionality reduction framework, which allows us to define the energy function for determining the optimal feature number. Similar to PCA [14] in the dimensionality reduction framework, the energy of a matrix is defined by the summation of all its eigenvalues. Suppose the eigenvalues of matrix $C = X X^T$ are $\lambda_i, i = 1, 2, \dots, d$, the energy of C is defined as $E = \sum_{i=1}^d \lambda_i$. The Lemma-1 gives another explanation of the matrix energy.

Lemma-1: let $\lambda_i, i = 1, 2, \dots, d$ be the eigenvalues of the matrix $C = X X^T$, then the energy of C is $E = \sum_{i=1}^d \lambda_i = \operatorname{tr}\{C\} = \sum_{i=1}^d c_{ii}$.

Through Lemma-1, we can translate the energy function of feature extraction problem, which is defined by matrix eigenvalues, to the energy function of feature selection problem, which is defined by the matrix trace. Given a matrix $C = \{c_{ij}\} \in R^{d \times d}$, the definition of feature score in Eqn. (12) shows the truth that $\operatorname{Score}(k) = \sum_{i=1}^n (x_{ki})^2 = c_{kk}$. Without loss of generality, we sort the features in a decreasing order according to their scores. Thus the energy of the reduced matrix after selecting p_1 number of features is,

$$E(p_1) = \sum_{l=1}^{p_1} c_{ll} = \sum_{l=1}^{p_1} \operatorname{Score}(l). \quad (15)$$

The percentage of energy preserved after feature selection is,

$$R(p_1) = \frac{E(p_1)}{E} = \frac{\sum_{l=1}^{p_1} \operatorname{Score}(l)}{\sum_{i=1}^d \operatorname{Score}(i)}. \quad (16)$$

Given the user defined energy threshold θ , the p_1 can be optimized by,

$$p_1 = \operatorname{argmin}_r R(r), \text{ s.t. } R(r) \geq \theta. \quad (17)$$

The energy threshold used by many other feature extraction algorithms are generally $\theta = 0.8$ or larger [14]. In this work, we propose to use the matrix energy function for determine feature number of FLSLI algorithm and experimentally show that on the TREC2 and TREC3 datasets, reserving 90% of matrix energy, we can reduce 90% of original features on which the LSI can still be well approximated. In addition, if only 70% energy is reserved, the LSI still can be approximated reasonably well with more than 98% features removed for SVD computation. Details are given in Section 5.

4. Theoretical analysis

The review of LSI in Section 3.1 shows that LSI aims to optimize a rescaled projection matrix such that any document can be projected to the latent semantic

space by the projection matrix U_p . Suppose we have already computed $U_p = \{u_1, u_2, \dots, u_p\}$ by SVD, where $u_j \in R^d$, $j = 1, 2, \dots, p$ are known as the bases of the latent semantics space. If we remove some features (terms) by $W \in H_{fs}$, the u_j after feature selection is $W^T u_j \in R^p$. We can reconstruct it to the d -dimensional space by $u'_j = WW^T u_j$. To minimize the affection of feature selection in approximating LSI, it is expected that the Euclidean distance between the bases of latent semantic space and their reconstruction $\|u_j - WW^T u_j\|$, $j = 1, 2, \dots, p$ can be minimized. Since we only reserve the singular vectors with large singular values as the bases of the latent semantic space, we can weight the importance of the distance $\|u_j - WW^T u_j\|$ by its singular value σ_j . Thus the optimal projection matrix W for feature selection in approximate LSI should minimize $\sum_j (\sigma_j \|u_j - u'_j\|)^2$. It equals to select the features which will have low weights on the bases of the latent semantic space. Thus the optimal features selected for approximating LSI should have the ability to minimize

$$\sum_j \sigma_j^2 \|u_j - u'_j\|^2. \quad (18)$$

However, Eqn. (18) can be minimized only after the SVD has been computed on the original terms by document matrix X since u_j is the j^{th} left singular vector of X . While our algorithm aims to select features before SVD to reduce the data scale. The Theorem-2 below guarantees that the solution of our proposed feature selection algorithm before SVD computation equals to the solution of equation (18) if SVD is implemented on original X . This result makes it possible to select the optimal group of features before the real computation of SVD for approximate LSI.

Theorem-2: The features selected from $\text{argmax}_{W \in H_{fs}} J_{LSI}(W)$ are the same as those from $\text{argmin}_{W \in H_{fs}} \sum_j \sigma_j^2 \|u_j - u'_j\|^2$.

The proof of this theorem is given in the appendices. Theorem 2 guarantees that the features selected by the discrete optimization of LSI's objective function are exactly the ones which are optimal for approximating LSI. Thus besides scalability and easiness to be implemented, another advantage of FLSI is that it theoretically guarantees to minimize the loss in approximation the original LSI.

5. Experiments

In this Section, we evaluate the effectiveness of the approximate LSI on large scale text corpus. Section 5.1 gives the detailed experimental configuration. In Section 5.2, the experiments are introduced to show

that our proposed algorithm can truly approximate classical LSI in IR tasks. In Section 5.3, some extensive study of the approximate LSI in large scale IR tasks are provided. The final subsection is for sensitivity analysis of our proposed algorithm.

5.1. Experimental setup

In this paper, all text documents are indexed in the vector space model through Lemur Language Modeling Toolkit [24]. The porter stemmer is used for stemming. We utilized the SVDLIBC (Doug Rohde's SVD C library version 1.34) [25] to perform the SVD. In the final evaluation stage, we directly used the TREC evaluation tool [21] for result evaluation. Two computers are used for the computation. Machine-I is with 2.4GHz AMD processor and 15GB RAM. Machine-I is used for the computation of our proposed ALSI. Machine-II is with the same processor but 50GB memory. Machine-II is used to run classical LSI on large scale datasets for comparative study.

Toward large scale study, we used the TREC2 and TREC3 datasets, which share the same document collections but have different queries (topics), for the evaluation. For the experiments on a larger scale dataset, we used the TIPSTER dataset, which combines all documents of TREC3 for ad hoc retrieval and routing task, with more than 1 million documents. In addition, to make the readers easy to verify the correctness of our experiments, we also report the experimental results on two toy datasets, which are commonly used benchmark text datasets in SMART collection [18]. Table 2 lists the description of all the datasets used in our experiments.

Table 2. Description of all datasets.

Dataset	# Documents	# Terms	# Queries
TIPSTER	1,078,118	815,398	100
TREC2	742,358	642,889	50
TREC3	742,358	642,889	50
CISI	1,460	7,276	35
CRAN	1,400	6,540	225

Using LSI in the IR task, the documents are generally ranked by the Cosine similarity calculated by Eqn. (3) in the latent semantic space. A commonly used baseline for evaluating the effectiveness of LSI is to rank documents by the Cosine similarity in the unreduced term space [11]. This can be calculated by setting $x_i = x'_i$ and $q = q'$ in Eqn. (3). We refer to this baseline approach as **Cosine** hereafter. We also involve **BM25** [17] with default parameter setting as a baseline algorithm for comparative study. To compare with

other dimension reduction algorithms in IR tasks, the random projection (RP) [4] is also involved as a baseline.

In this work, we use the TREC evaluation tool to evaluate the IR performance of different approaches [21]. They include $P@n$, MAP, g-map and R-prec etc. In addition, the statistical t-test is also involved. For demonstration, we mainly use the precision at n ($P@n$) and Mean Average Precision (MAP) to give some insights. $P@n$ is a measure of how well the system performs in not returning irrelevant documents to given query in the top n results.

$$P@n = \frac{\# \text{ relevant documents in top } n \text{ ranked results}}{n}.$$

The MAP is the mean of the Average Precision (AP) over all queries, which is defined through $P@n$. The AP is given by,

$$AP = \frac{\sum_{i=1}^N P@i \times rel(i)}{\# \text{ relevant documents}}$$

where N is the number of documents and $rel(i)$ is a binary function indicating whether the i^{th} document is relevant or not.

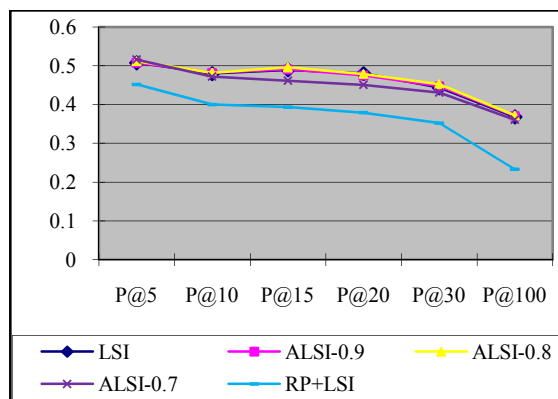
5.2. Results of approximating classical LSI by ALSI in IR task

In this subsection, we experimentally show how well ALSI can approximate the original LSI in IR tasks with much smaller computational cost. For this purpose, Machine-II is used to conduct classical LSI on the full TREC2 document set, which is the same as the TREC3 document set for ad hoc retrieval. And then we conduct the ALSI on the same document dataset by machine-I. All documents and queries are projected into the two latent semantic spaces learned by LSI and ALSI respectively for the final IR evaluation. The difference between their $P@n$ results and MAP are used for measuring their closeness in IR tasks. For comparison purpose, we also involve the Random Projection (RP) as the preprocessing of LSI to compare with our proposed ALSI. Recall Eqn. (17), the smaller the energy threshold θ is, the fewer features will be reserved after feature selection. Since many previous dimension reduction algorithms use 0.8 as the energy threshold for dimension reduction. In this subsection, we show the results of ALSI by setting $\theta = 0.7, 0.8,$ and 0.9 respectively. The results are given in Table 3, where the MAP-2 and MAP-3 are MAP results tested by TREC2 queries (topic 101-150) and TREC3 queries (topic 151-200) respectively.

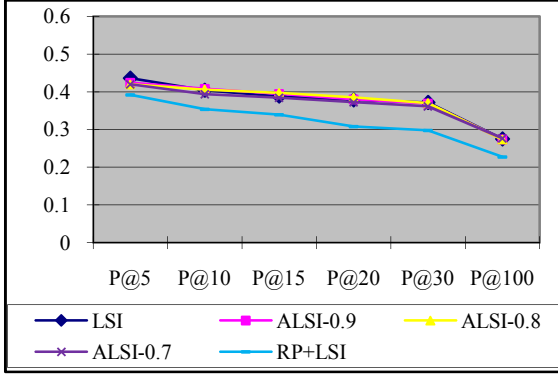
Table 3. Using ALSI to approximate original LSI

	LSI	ALSI $\theta=0.9$	ALSI $\theta=0.8$	ALSI $\theta=0.7$	RP+LSI
# feature	642,889	50,033	18,405	9,356	50,033
Time (s)	153,726	25,021	19,606	3,019	24,987
MAP-2	0.4742	0.4742	0.4737	0.4703	0.383
MAP-3	0.4091	0.4090	0.4046	0.3998	0.326

For the experiments listed in Table 3, the dimension of the projected feature spaces from both LSI and ALSI are set as 1,000. The affection of the projected space dimension will be studied in Section 5.4. The number of features selected for RP is set as 50,033, which is the same as that of ALSI with $\theta=0.9$. Table 3 shows that 50,033 (7.8%) features can retain about 90% of energy after feature reduction. The computational time can be significantly reduced while the MAP almost does not change in contrast to classical LSI. If we retain 70% energy, we only need to reserve less than 10,000 (1.5%) features. Through feature reduction, the computation efficiency can be 50 times faster than classical LSI while the MAP will only change very little. In contrast to using the RP feature reduction as the preprocessing of LSI, our proposed approach can much closely approximate LSI in terms of MAP. To give more insights of how well ALSI can approximate LSI in IR tasks, we show the $P@n$ of ALSI on the same TREC collection in Figure 1. This Figure shows that the $P@n$ of ALSI can well approximate LSI on different n values by setting $\theta=0.9$ and 0.8 since their curves are almost overlapped. If we only reserve 1.5% features by setting $\theta=0.7$, the $P@n$ of LSI can still be closely approximated by ALSI.



(a)TREC2



(b) TREC3

Figure 1. P@n of LSI and ALSI with different θ .

5.3. Results on large scale data

In Section 5.2, we verified that ALSI can well approximate classical LSI in IR tasks. In this subsection, we give the extensive study of ALSI on large datasets to show how the ALSI can improve the IR performance on large scale dataset. The detailed experimental results are summarized in Table 4. Besides P@n and MAP, more evaluation metrics provided by the TREC evaluation tool are used for the comparison. The evaluated results are measured on top 100 results of each query. In Table 4, the ALSI(0.8, 1000) stands for the results of using ALSI algorithm for text representation with $\theta=0.8$ and the dimension of reduced feature space is 1,000. LSI(1000) stands for using LSI to project documents into 1,000-dimensional latent semantic space.

Table 4. Approximate LSI (ALSI) used for IR tasks on different datasets.

Data	Model	P@5	P@10	P@30	P@100	map	g-map	R-prec	bpref	recip_rank
Tipster with topic 101-150	Cosine	0.488	0.502	0.5	0.4438	0.502	0.3929	0.1406	0.1227	0.6803
	BM25	0.54	0.502	0.442	0.3696	0.4769	0.2598	0.1153	0.1013	0.6806
	LSI(1,000)	0.524	0.494	0.464	0.3994	0.4863	0.374	0.127	0.1099	0.6317
	RP(1,000)	0.412	0.36	0.26	0.1548	0.4073	0.0639	0.0432	0.0406	0.5997
	ALSI(0.8, 1000)	0.52	0.498	0.4773	0.4188	0.4849	0.3731	0.1268	0.1099	0.6302
	RP(2500)	0.404	0.37	0.2647	0.1532	0.4317	0.0646	0.0437	0.0403	0.5912
	ALSI(0.8, 2500)	0.588	0.548	0.541	0.5187	0.5307	0.4182	0.1422	0.1263	0.6694
Tipster with topic 151-200	Cosine	0.44	0.42	0.388	0.3028	0.42	0.3415	0.2018	0.1675	0.5663
	BM25	0.352	0.332	0.248	0.1886	0.3066	0.1148	0.1214	0.1083	0.4916
	LSI(1000)	0.430	0.419	0.401	0.2892	0.4262	0.1899	0.1807	0.1551	0.5726
	RP(1000)	0.332	0.302	0.2353	0.126	0.3452	0.059	0.0923	0.0842	0.49
	ALSI(0.8, 1000)	0.432	0.418	0.4007	0.2968	0.4193	0.1895	0.1809	0.1555	0.5725
	RP(2500)	0.404	0.346	0.2453	0.1288	0.3832	0.0433	0.0979	0.091	0.5176
	ALSI(0.8, 2500)	0.504	0.462	0.446	0.3514	0.474	0.2763	0.218	0.1882	0.6394
TREC2	Cosine	0.508	0.52	0.486	0.4264	0.5001	0.3929	0.1955	0.1698	0.6692
	BM25	0.52	0.496	0.418	0.3518	0.4621	0.236	0.1579	0.139	0.6796
	LSI(1000)	0.508	0.48	0.4433	0.3678	0.4742	0.3618	0.1685	0.1456	0.6857
	RP(1000)	0.432	0.368	0.252	0.133	0.423	0.0784	0.0564	0.0528	0.6039
	ALSI(0.8, 1000)	0.508	0.482	0.4467	0.3702	0.4737	0.3628	0.1699	0.1467	0.6739
	RP(2500)	0.396	0.362	0.2407	0.121	0.3955	0.0467	0.0567	0.0538	0.5394
	ALSI(0.8, 2500)	0.58	0.54	0.5047	0.425	0.5212	0.4181	0.1961	0.1735	0.6993
TREC3	Cosine	0.42	0.414	0.3793	0.3086	0.4199	0.3324	0.2011	0.1748	0.5606
	BM25	0.532	0.444	0.3893	0.2728	0.4486	0.2526	0.1836	0.1636	0.6115
	LSI(1000)	0.436	0.404	0.3713	0.2748	0.4091	0.172	0.169	0.1478	0.5668
	RP(1000)	0.392	0.23	0.16	0.0826	0.3733	0.055	0.0973	0.0906	0.5132
	ALSI(0.8, 1000)	0.424	0.408	0.3693	0.274	0.4046	0.1493	0.1676	0.1471	0.5557
	RP(2500)	0.384	0.346	0.2407	0.1304	0.3836	0.0575	0.1008	0.0945	0.5402
	ALSI(0.8, 2500)	0.552	0.478	0.4213	0.3234	0.4504	0.287	0.2033	0.1767	0.6736
CISI	Cosine	0.4105	0.3553	0.2399	0.1496	0.3519	0.2672	0.2402	0.4539	0.6399
	BM25	0.3316	0.2934	0.218	0.1429	0.3009	0.2311	0.2	0.4215	0.5702
	LSI(1000)	0.4395	0.3697	0.2614	0.1539	0.3729	0.2919	0.2448	0.457	0.6506
	RP(1000)	0.2	0.1816	0.1281	0.0886	0.1997	0.0873	0.1074	0.2334	0.3948
	ALSI(0.8, 1000)	0.4316	0.3697	0.2487	0.1517	0.3609	0.2882	0.2547	0.4607	0.6369
CRAN	Cosine	0.4471	0.3098	0.1535	0.0601	0.5137	0.3754	0.3975	0.7738	0.8104
	BM25	0.4409	0.3107	0.1504	0.059	0.5136	0.3606	0.4022	0.7604	0.8052
	LSI(1000)	0.455	0.3251	0.1579	0.0602	0.5221	0.3819	0.4082	0.7893	0.8043
	RP(1000)	0.3867	0.264	0.1234	0.0478	0.4944	0.2929	0.3416	0.6195	0.7537
	ALSI(0.8, 1000)	0.456	0.3227	0.1578	0.0609	0.5268	0.3857	0.4104	0.7818	0.8133

Table 4 shows that if we project the text documents into the same dimensional space by ALSI and LSI respectively, the results of ALSI can approximate LSI in IR tasks over almost all datasets and all evaluation metrics. This enhanced the conclusion that ALSI can closely approximate LSI in IR tasks. In many previous LSI studies, the dimension of the projected space by LSI did not exceed 1,000. Our experimental results on the large scale Tipster, TREC2 and TREC3 datasets show that if using classical LSI to project data to the 1,000-dimensional spaces, the results are worse than Cosine or BM25, by which the documents are ranked without LSI. This verifies the previous conclusions that it is hard for LSI to improve IR performance on large scale dataset [12]. The reason why many previous studies only compute the leading 1,000 eigenvectors is led by high complexity of SVD. Similarly, we cannot compute more than 1,000 eigenvectors by classical LSI. However, our proposed FLSI can greatly reduce the data scale, and thus we can compute more eigenvectors on the reduced data. For example, if we project the data into 2,500-dimensional space by ALSI, the IR performances are improved on almost all large scale datasets. As for the detailed examples, the $P@10$ can be relatively improved about 10% on Tipster with topic 101-150 and 24% with topic 151-200. The experiments show that the ALSI can overcome the bottleneck caused by SVD. To guarantee the performance improvement of ALSI compared with the three baseline algorithms are statistically significant, we utilize the paired t-test (2-tailed) over the results of all datasets and all evaluation metrics. The results $2.3e-6 < 0.01$, $2.6e-11 < 0.01$ and $2.4e-18 < 0.01$ in contrast to Cosine, BM25 and RP respectively show the improvement of ALSI is statistically significant.

5.4. Sensitivity analysis

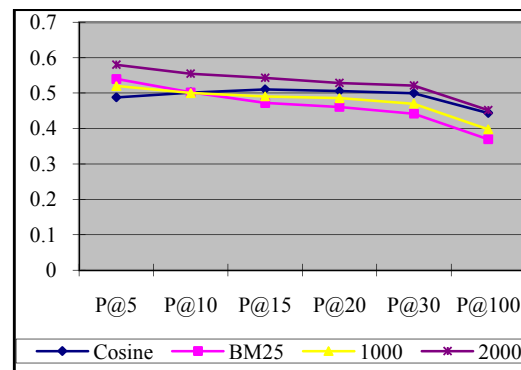
The experimental results reported in Table 4 still leave two questions for us. The first is how the energy threshold θ can affect the performance of ALSI. The second is how the dimension of the space projected by ALSI can affect the performance. Some preliminary study for the first question has been given in Section 5.1. To better answer these two questions, we present the results of $P@n$ on TIPSTER with different parametric values in Figure 2. The numbers 1,000 and 2,000 in Figure 2 stand for the dimension of the projected space. We ignore the baseline RP in Figure 2 since its performance is incomparable with others. From these results, we can have the following observations:

1. If the energy threshold is no smaller than 0.8 and the dimension of the projected space is no smaller than 2,000, ALSI can perform better than Cosine and

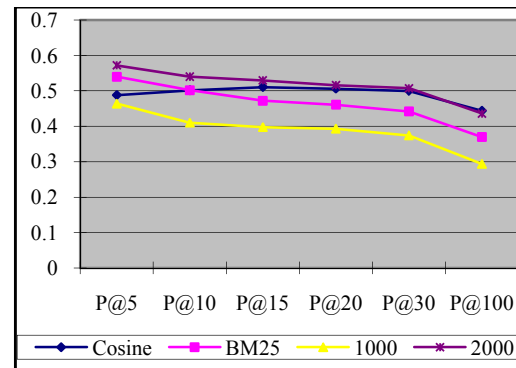
BM25 consistently. Since we have verified that ALSI can closely approximate LSI, this observation means that LSI can truly improve the IR performance on large scale data if the documents are projected into a relatively high dimensional space.

2. The larger the energy threshold is, the better performance can be achieved by ALSI. This can be observed from each row of Figure 2. For the same dimension of projected space, its $P@n$ results can be improved for larger energy threshold. Recall the results showed in Section 5.2, energy threshold 0.7 is high enough for approximating LSI on TREC2, however on the larger scale Tipster dataset, $\theta=0.8$ is required.

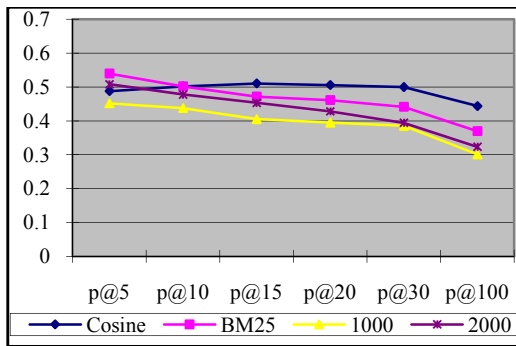
The larger the projected space dimension is, the better performance can be achieved by ALSI. This can be observed from each picture in Figure 2. However, since LSI aims to give the optimal low rank approximation of original term by document matrix, theoretically this observation cannot be always true. If the dimension of the projected space is close to or larger than the rank of the matrix processed by SVD, the IR performance will not be improved any more. Thus the rank of the matrix for SVD computation upper bounds the dimension of the latent semantic space. The IR performance will be improved with the increase of subspace dimension under this upper bound.



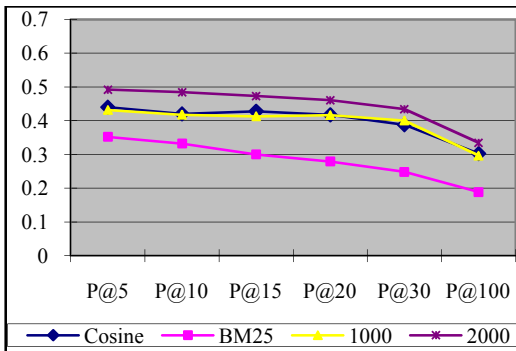
(a) $\theta=0.9$ (Topic101-150)



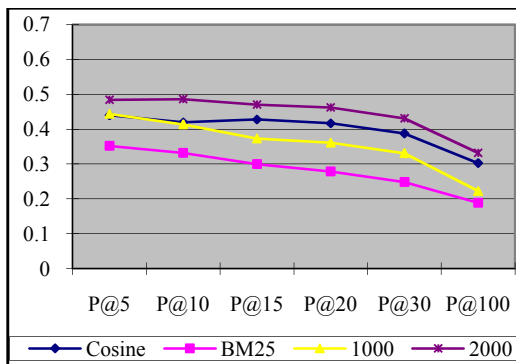
(b) $\theta=0.8$ (Topic101-150)



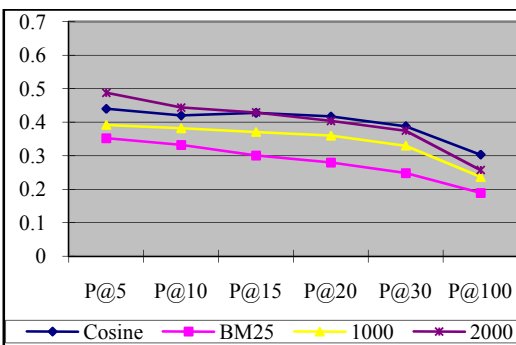
(c) $\theta = 0.7$ (Topic101-150)



(d) $\theta = 0.9$ (Topic151-200)



(e) $\theta = 0.8$ (Topic151-200)



(f) $\theta = 0.7$ (Topic151-200)

Figure 2. Sensitivity of ALSI to projected space dimension and energy threshold on Tipster (measured by $P@n$)

5.5. Discussion

Note the FLSI algorithm scores features as simple as calculating the Frobenius norm of each feature. Experimental results show that it can outperform some other feature reduction algorithms such as random projection. We did not list more baseline feature selection algorithms in this paper due to the space limitation. Through our observation, the features selected by l_1 -norm can give slightly worse but very similar performance to Frobenius norm. Thus for further simplify the FLSI algorithm, we can select features by their l_1 -norm instead, which can give similar performance.

6. Conclusion

The bottleneck of Latent Semantic Indexing in IR tasks is the high complexity of Singular Value Decomposition (SVD). In this work, we propose a scalable straightforward feature selection algorithm, FLSI, as preprocessing of LSI to make it applicable on large scale text dataset. The feature selection can reduce the size of feature set such that SVD can be applied on the much smaller scale reduced data. Theoretically, the proposed feature selection algorithm can guarantee that the LSI implemented on the feature reduced dataset can closely approximate its counterpart without feature selection. In addition, the data reduction allows us to calculate more eigenvectors than classical LSI. The proposed approach is evaluated on several large scale text collections. The experimental results show that our proposed algorithm can closely approximate the classical LSI and the approximate LSI can significantly improve the IR performance. Our proposed algorithm offers a general way for studying and applying LSI on large scale dataset.

7. Acknowledgment

This work is partially supported by NRF/IDM Grant of R-263-000-524-279, Singapore.

8. References

- [1] Berry, M. W. and Martin, D. I. 2000. Parallel SVD for scalable information retrieval. In Proceedings of the International Workshop on Parallel matrix algorithms and applications (Neuchâtel, Switzerland, 2000).
- [2] Berry, M. W., Dumais, S. T. and O'Brien, G. W. 1995. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37, 4 (1995), 575-595.
- [3] Berry, M. W. 1992. Large-scale sparse singular value computations. *Int. J. Supercomp. Appl.*, 6, 1(1992), 13-49.

[4] Bingham, E. and Mannila, H. 2001. Random projection in dimensionality reduction: applications to image and text data, In Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining (San Francisco, California, 2001) SIGKDD'01. ACM Press, New York, NY, 245-250.

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

[6] Dumais, S. T. 1992. LSI meets TREC: A Status Report. In Proceedings of the 1st Text Retrieval Conference (Washington, November 1992) TREC-1. NIST special publication, 137-152.

[7] Dumais, S. T. 1993. Latent Semantic Indexing (LSI) and TREC-2. In Proceedings of the 2nd Text Retrieval Conference (Gaithersburg, MD, 1993) TREC-2. 105-116.

[8] Dumais, S. T. Using LSI for information filtering: TREC-3 experiments. In Proceedings of the 3rd Text Retrieval Conference, TREC-3. Technical Report 500-335.

[9] Dumais, S. T. 1996. Combining evidence for effective information filtering. In Proceedings of the AAAI Spring Symposium on Machine Learning and Information Retrieval (1996). Technical Report SS-96-07.

[10] Dumais, S. T., Letsche, T. A., Littman, M. L. and Landauer, T. K. 1997. Automatic Cross-Language Retrieval Using Latent Semantic Indexing. American Association for Artificial Intelligence (AAAI) Symposium on Cross Language Text and Speech Retrieval, 24-26 (Palo Alto, CA, March, 1997). 15-21.

[11] Greengrass, E. 2000. Information Retrieval: A Survey. Technical Report CS-TR-3514. University of Maryland, MD.

[12] Husbands, P., Simon, H. and Ding, C. 2000. On the use of the Singular Value Decomposition for Text Retrieval. In Proceedings of 1st SIAM Computational Information Retrieval Workshop (Raleigh, NC, 2000).

[13] Husbands, P., Simon, H. and Ding, C. 2001. On the use of the Singular Value Decomposition for Text Retrieval. *Computational Information Retrieval (2001)*, 145-156.

[14] Jolliffe, I.T. *Principal Component Analysis*. New York, Springer Verlag, 1986.

[15] Karypis, G. and Han, E-H. 2000. Concept indexing: a fast dimensionality reduction algorithm

with applications to document retrieval & categorization. Technical Report. University of Minnesota.

[16] Kolda, T. and O'Leary, D. 1998. A Semidiscrete Matrix Decomposition for Latent Semantic Indexing in Information Retrieval. *ACM Trans. Inf. Syst.* 16, (1998), 322-346.

[17] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., and Gatford, M. 1994. Okapi at TREC-3. In Proceedings of the 3rd Text REtrieval Conference (Gaithersburg, USA, November, 1994) TREC-3.

[18] SMART. <ftp://ftp.cs.cornell.edu/pub/smart>

[19] Tang, C., Dwarkadas, S. and Xu, Z. 2004. On scaling latent semantic indexing for large peer-to-peer systems. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval SIGKDD '04. ACM Press, New York, NY, 112-121.

[20] Theodoridis, S. and Koutroumbas, K. *Pattern Recognition, Third Edition*. Academic Press, Inc., Orlando, FL, 2006.

[21] TREC. http://trec.nist.gov/data/docs_eng.html

[22] Yan J, Zhang BY, Liu N, Yan S, Cheng Q, Fan W, Yang Q, Xi W, Chen Z. 2006. Effective and efficient dimensionality reduction for large scale and streaming data preprocessing. *IEEE Trans. Knowl. Data Eng.* 18(2) (2006), 320-333.

[23] http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/

[24] <http://www.lemurproject.org/>

[25] <http://tedlab.mit.edu/~dr/SVDLIBC/>

Appendices

Theorem-1 proof: Let $W = \{w_1, w_2, \dots, w_p\}$, since $W^T W = I$, we have

$$w_i^T w_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Thus $\text{tr}\{W^T C W\} = \sum_{k=1}^p w_k^T C w_k$, the Lagrangian function of the optimization problem with multiplier λ_k is

$$\mathcal{L}(w_k, \lambda_k) = \sum_{k=1}^p w_k^T C w_k - \lambda_k (w_k^T w_k - 1).$$

Maximizing $\mathcal{L}(w_k, \lambda_k)$ leads to zeros at the saddle points, i.e.

$$\frac{\partial \mathcal{L}(w_k, \lambda_k)}{\partial w_k} = (C - \lambda_k I) w_k = 0$$

Thus w_k should satisfy that $Cw_k = \lambda_k w_k$ for $k = 1, 2, \dots, p$. This means that $\lambda_k, k = 1, 2, \dots, p$ are the eigenvalues of C , and w_k 's are the corresponding eigenvectors. Note that

$$\text{tr}\{W^T C W\} = \sum_{k=1}^p w_k^T C w_k = \sum_{k=1}^p \lambda_k w_k^T w_k = \sum_{k=1}^p \lambda_k.$$

Therefore, $\text{tr}\{W^T C W\}$ is maximized when W is composed of the first p leading eigenvectors of C . Suppose the SVD of matrix X is $X = U \Sigma V^T$. It is easy to see that

$$C = X X^T = U \Sigma V^T \cdot V \Sigma^T U^T = U \Sigma^2 U^T$$

So the projection matrix of LSI consists of the leading eigenvectors of the matrix C . The square roots of matrix C 's eigenvalues are the rescaling factors of LSI. Thus the solution of LSI is exactly the solution of the optimization problem defined in Eqn.(4). ■

Theorem-2 proof: The matrix $C = X X^T$ is symmetric and semi-definite. Here we assume that the eigenvalues and corresponding eigenvectors of the matrix C are λ_k and u_k , $k = 1, 2, \dots, d$, respectively. Then we have $C = \sum_{k=1}^d \lambda_k u_k u_k^T$ and

$$J_{LSI}(W) = \text{tr}\{W^T C W\} = \text{tr}\left\{\sum_{k=1}^d \lambda_k W^T u_k u_k^T W\right\}.$$

Thus,

$$\begin{aligned} \arg\max_{W \in H_{fs}} J_{LSI}(W) &= \\ \arg\max_{W \in H_{fs}} \text{tr}\left\{\sum_{k=1}^d \lambda_k W^T u_k u_k^T W\right\}. \end{aligned}$$

On the other hand,

$$\begin{aligned} \sum_j \sigma_j^2 \|u_j - u_j'\|^2 &= \sum_j \sigma_j^2 \|u_j - W W^T u_j\|^2 \\ &= \sum_j \sigma_j^2 (u_j - W W^T u_j)^T (u_j - W W^T u_j) \\ &= \sum_j \sigma_j^2 (1 - u_j^T W W^T u_j). \end{aligned}$$

Suppose the SVD of X is $X = U \Sigma V^T$, then $C = X X^T = U \Sigma^2 U^T$. We have $\lambda_k = \sigma_k^2$, $k = 1, 2, \dots, d$. Thus

$$\begin{aligned} &\arg\min_{W \in H_{fs}} \sum_j \sigma_j^2 \|u_j - u_j'\|^2 \\ &= \arg\min_{W \in H_{fs}} \sum_j \lambda_j (1 - u_j^T W W^T u_j) \\ &= \arg\max_{W \in H_{fs}} \sum_j \lambda_j u_j^T W W^T u_j \\ &= \arg\max_{W \in H_{fs}} \text{tr}\left\{\sum_{k=1}^d \lambda_k W^T u_k u_k^T W\right\}. \\ &= \arg\max_{W \in H_{fs}} J_{LSI}(W) \end{aligned}$$

■