Learning Similarity Measures in Non-orthogonal Space*

Ning Liu [1], Benyu Zhang[2], Jun Yan [3], Qiang Yang[4], Shuicheng Yan [2], Zheng Chen[2], Fengshan Bai [1], Wei-Ying Ma[2]

[1] Department of Mathematical Science, Tsinghua University, Beijing, 100084, P.R. China
liun01@mails.tsinghua.edu.cn
fbai@math.tsinghua.edu.cn

[2]Microsoft Research Asia, 49 Zhichun Road, Beijing 100080, P.R. China
{byzhang, zhengc, wyma}@microsoft.com
v-scyan@msrchina.research.microsoft.com

[3] LMAM, Department of Information Science, School of Mathematical Science, Peking University, Beijing 100871, P.R. China
yanjun@math.pku.edu.cn

[4]Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong
qyang@cs.ust.hk

ABSTRACT

Many machine learning and data mining algorithms crucially rely on the similarity metrics. The Cosine similarity, which calculates the inner product of two normalized feature vectors, is one of the most commonly used similarity measures. However, in many practical tasks such as text categorization and document clustering, the Cosine similarity is calculated under the assumption that the input space is an orthogonal space which usually could not be satisfied due to *synonymy* and *polysemy*. Various algorithms such as Latent Semantic Indexing (LSI) were used to solve this problem by projecting the original data into an orthogonal space. However LSI also suffered from the high computational cost and data sparseness. These shortcomings led to increases in computation time and storage requirements for large scale realistic data. In this paper, we propose a novel and effective similarity metric in the non-orthogonal input space. The basic idea of our proposed metric is that the similarity of features should affect the similarity of objects, and vice versa. A novel iterative algorithm for computing non-orthogonal space similarity measures is then proposed. Experimental results on a synthetic data set, a real MSN search click-thru logs, and 20NG dataset show that our algorithm outperforms the traditional Cosine similarity and is superior to LSI.

*This work conducted at Microsoft Research Asia.

Categories and Subject Descriptors
I.5.3 [**Pattern Recognition**]: Clustering and Applications – *similarity measures, text processing.*

General Terms
Algorithms, Measurement.

Keywords
Similarity Measures (SM), Vector Space Model (VSM), Non-Orthogonal Space (NOS), Latent Semantic Indexing (LSI).

1. INTRODUCTION

The performance of many data mining algorithms such as document clustering and text categorization critically depends on a good metric that reflects the relationship between the data objects in the input space [2] [15]. It is therefore important to calculate the similarity as effectively as possible [12]. In the classical Vector Space Model (VSM) [1], queries and documents are represented as vectors of terms. These vectors define an input space where each distinct term represents an axis of that space. Then the similarity of two documents or two queries equals to the cosine of the angle between the high dimension vectors indexed by the terms in corpus [9]. This approach is an effective approximation, but it is nevertheless an oversimplification. The major limitation is that it assumes that the terms are independent, i.e. the dimensions of the input space are orthogonal. However, in text application, the input space is usually non-orthogonal due to the following issues.

There are two common problems with the Vector Space Model [1], [4]. The first is *synonymy*. For example, the word "building" can also be represented by "house" or "construction". The second is *polysemy* which means that most words have more than one meaning. For example, "paper" refers to a material made of cellulose pulp, a formal written composition intended to be published, or even an official document. These facts show that the terms are not independent, i.e. the space is a *non-orthogonal* one. Thus the similarity measured by cosine or inner product based on Euclidean's distance can not exactly describe the relationship between objects.

Recently, attempts have been made to incorporate semantic knowledge with the vector space representation. Latent Semantic Indexing (LSI) [4] [5] [6] [1]is a well-known approach among them. LSI attempts to capture the term-term statistical relationships. In LSI, the document space in which each dimension is an actual term occurring in the collection is replaced by a much lower dimensional document space called the LSI space in which each dimension is a derived concept. Documents are represented by LSI space and vector similarity can be calculated in the same way in LSI space as the traditional VSM. Nevertheless, LSI has its own weakness. For instance, the concept space of LSI is hard to be explained intuitively. Another problem is that the computational complexity of SVD algorithm is too high, which is $O(N^3)$, where N is less value of the number of terms and documents, which makes LSI a time-consuming process. All these problems make the LSI algorithm infeasible for large scale, sparse data sets. Moreover, how to determine the optimal reduced dimensionality is still not solved properly.

Among the various approaches used to deal with non-orthogonal space problem, which gained more interest recently, the distance metric learning approach [11] [13] is based on posing metric learning as a convex optimization problem and other approaches [7] used the Mahalanobis distance to describe the similarity. However, all of them are supervised learning algorithms which learn the similarity metric or distance matrix that critically depends on training samples. Due to this reason, such methods are limited in flexibility due to the lack of enough training data which often occurs in most real tasks.

In this paper, we propose a novel iterative similarity learning approach to measure the similarity among the objects in the non-orthogonal feature space. Our proposed algorithm is based on an intuitive assumption that the similarity of features should affect the similarity of objects and vice versa. Compared with the traditional algorithms such as LSI, our method has the following advantages: (1) it can learn the similarity in the original feature space. Thus, it preserves the sparse structure of the original data and consequently the storage complexity is lower; (2) the time complexity is much lower than SVD. Experimental results show that our algorithm outperforms Cosine similarity, and is no worse and often is better than LSI.

The rest of the paper is organized as follows. In Section 2, we introduce some background knowledge on similarity measurement and learning, such as Cosine similarity, LSI algorithms and supervised algorithms. Following that, we present the problem formulation and the detailed algorithm in Section 3. The experimental results on the synthetic datasets and the real data are demonstrated in Section 4. Conclusion and future work are presented in Section 5.

2. BACKGROUND

Many information mining problems, such as text classification and text clustering are suffering from the problems to understand the representation of documents and to capture the relationship between documents. In this section, we will introduce three popular approaches relative to these problems: Cosine similarity, Latent Semantic Indexing and supervised learning algorithms.

2.1 Cosine Similarity in Vector Space Model

Vector Space Model (VSM) is the first approach to represent documents as a set of terms [3] [9]. This set of terms defines a space such that each distinct term represents the entries in that space. Since we are representing the documents as a set of terms, we can view this space as a "document space". We can then assign a numeric weight to each term in a given document, representing an estimate of the usefulness of the given term as a descriptor of the given documents [8]. The weights assigned to the terms in a given documents can then be interpreted as the coordinates of the document in the documents space. For instance, $B_{t\times d}$ is the matrix of term by documents in the case of text data, where $b_{i,j}$ is the term weighting. Then, a vector similarity function, such as the cosine of the angle between the high dimension vectors indexed by the terms in corpus, can be used to compute the similarity matrix among documents. For two vectors b_i and b_j the Cosine similarity is given by:

$$sim^{\cos}(b_i,b_j)=\cos(\theta)=\frac{<b_i,b_j>}{\|b_i\|\|b_j\|} \qquad (1)$$

where θ is the angle between b_i and b_j . For the term by document matrix B , if the entire column vectors in B have been normalized, the similarity matrix will be:

$$sim^{\cos}(doc,doc)=B^T B\ . \qquad (2)$$

For an ideal similarity measure, the maximum similarity is one, corresponding to the two document vectors being identical (angle between them is zero). The minimum similarity is zero, corresponding to the two vectors having no terms in common (angle between them is 90 degree). Others should between zero and one since any cosine value of an acute angle should between zero and one.

The major limitation of Cosine similarity is that it assumes the terms are independent, orthogonal dimensions of the space, but in fact it is usually not orthogonal. Thus mistakes occur under this assumption. For instance, consider the following two sentences:

C1: Human machine *interface* for Lab ABC *computer* applications.

C5: Relation of *user*-perceived *response time* to error measurement

Using the simplest counting strategy to establish a vector space model, the transposed object matrix B^T could be shown in Table 1.

Table 1. VSM Example

	computer	*Human*	*interface*	*response*	*time*	*user*
C1	1	1	1	0	0	0
C5	0	0	0	1	1	1

Following the example above, c1 and c5 are both talking about the human-computer interaction [4], thus similar in semantic space. However, the Cosine similarity of c1 and c5 by formula (1) are zero which means that they are not similar at all. This mistake occurs due to our assumption that the input space is orthogonal while it is not. The terms are not orthogonal due to there are correlations among different words, for example, 'user' in c5 and 'human' in c1 have strong relationship in semantic space, but we assume that they are orthogonal basis of the vector space. Figure 1 gives the intuitive interpretation of Cosine similarity in non-orthogonal space.

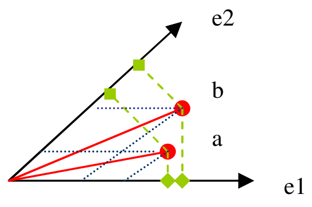

Figure 1 Cosine similarity in non-orthogonal space. Since the basis vectors are not orthogonal, the projections and the components of vectors are not the same.

For simplicity, we consider a two dimensional case. Suppose that "e1" and "e2" are basis vectors (i.e., terms) of a vector space model, "a" and "b" are two vectors (i.e. documents) in this space. Since the basis vectors are not orthogonal, the projections (pictured by broken line with square end) and the components (pictured by dot line) of these vectors are not the same. The Cosine similarity of a and b which considers the *components* as projections will certainly lead to the wrong solution. Thus, the Cosine similarity is less suitable for non-orthogonal vector space.

2.2 Latent Semantic Indexing

Latent Semantic Indexing (LSI) [6] [4] is a information retrieval method designed to overcome two common problems in information retrieval: *synonymy* and *polysemy*. In other words, LSI aims at projecting original data in vector space to an orthogonal space in where the Cosine similarity will not lead to mistakes. From a very high level, LSI tries to take advantage of the conceptual content of documents. A technique known as Singular Value Decomposition (SVD) is used to create this concept space. Below is a simplistic overview of what happens in the preprocessing stage of LSI and how SVD is used [14].

LSI takes the original term by document matrix $B_{m\times n}$ as input. Then, the SVD projection is computed by decomposing matrix B into the product of three matrices $B_{m\times n} = T_{m\times m} S_{m\times n} D_{n\times n}^T$, where $N = \min(m,n)$, T and D have orthonormal columns and S is diagonal. By restricting the matrixes T , D and S to their first $k < n$ rows one obtains the matrix $\tilde{B}_{m\times n} = \tilde{T}_{m\times k} \tilde{S}_{k\times k} \tilde{D}_{k\times n}^T$. $\tilde{B}$ is the best square approximation of B by a matrix of rank *k*.

The inner product between two column vectors of $\tilde{B}$ reflects the extent to which two documents have a similar profile of terms. Thus the matrix $\tilde{B}^T \tilde{B}$ contains the document-to-document similarity:

$$sim^{LSI}(doc, doc) = \tilde{B}^T \tilde{B} = \tilde{D} \tilde{S}^2 \tilde{D}^T \qquad (3)$$

The research into LSI so far has been encouraging. However, LSI has some shortcomings in performance. One is the computational complexity of SVD algorithm is $O(N^3)$ where *N* is the minor value between the number of terms and documents. It makes the LSI algorithm unfeasible for large sparse dataset. Meanwhile, LSI has the storage space problem. After performing an SVD, the approximate matrix $\tilde{B}$ is not a sparse matrix. Furthermore, the problem with LSI is that the concept space is hard to understand by humans and the similarity between documents will appear negative values. Besides, a parameter k under the user's control can affect the information reservation, i.e. the choice of parameter k is still an opening issue and different choice may be affect the final similarity greatly.

2.3 Supervised Similarity Learning

Recently, researchers have considered using distance metrics to measure the similarity of objects. They present some algorithms that can learn a distance metric to increase the accuracy of information retrieval [11] [13]. The Mahalanobis distance is a very considerable way of determining the similarity of a dataset. Mahalanobis distance uses $B^T AB$ to replace the original inner product, where A is a parameterized family of distance metrics. The learning distance algorithms discriminately searches for the parameters that best fulfill the training data. These methods were shown to be very effective; however, measuring the similarity of individual web objects may not be precise when using their algorithms due to no enough training data to learn the parameters.

3. LEARNING SIMLARITY IN NON-ORTHOGONAL SPACE

In this section, we present the problem formulation and give the detailed similarity learning algorithm in non-orthogonal space [10]. We derivate our algorithm from a simple intuitive assumption: "the similarity of features should affect the similarity of objects and vice versa." Following that is the convergence proof of this algorithm. Then, the parameter choosing strategy is given at the end of this section.

3.1 Problem Formulation

Since we are interested in not only the problem of terms and documents, but also other information retrieval problems such as measuring the similarity among queries and pages and so on, in the interest of generality, we use 'object' and 'feature' to represent the detailed problems such as documents and terms or queries and pages in the sequel. The classical Cosine similarity is in fact the inner product of two normalized vectors in an orthogonal vector space. Mistakes occur since we apply it in the non-orthogonal space. This motivates us to give a novel similarity measure which is suitable for a non-orthogonal space.

Suppose that x , y are objects in an *n*-dimensional vector space H^n . The i^{th} entry of x is denoted as x_i which is the value of x on the i^{th} feature. Let x^T be the transpose of vector x . Matrix is represented by capital letter B whose columns are objects in H^n and $b_{i,j}$ is the value of the i^{th} object on the j^{th} feature space. Since these bag of words like vectors have very sparse structure, this sparse structure could save storage requirement greatly and perform quick algorithm so as to save time requirement. As discussed above, our problem is learning the similarity based on the feature-object matrix *B* of which the features may not be independent, i.e. the space is a non-orthogonal one, and preserving the sparse structure of matrix *B*.

3.2 Equations for Similarity Learning

There are many approaches such as kernel based algorithms [7] measure the similarity in non-orthogonal space by,

$$S(o_i, o_j) = o_i^T P o_j ,$$

where o_i and o_j are two objects and P is a semantic proximity matrix satisfies the symmetric and positive semi-definite. Let $B \in R^{m\times n}$ be a feature by object matrix, which could be looked as objects in feature space from columns or features in object space from rows. Suppose S_o and S_f are the similarity matrix between objects and the similarity matrix between features respectively. Firstly, let us interpret our basic assumption, "the similarity of features should affect the similarity of objects and vice versa." In other words, two objects should be more similar if their features are more similar; on the other hand, two features should be more similar if their corresponding objects are more similar, the two factors should affect each other and could not be considered individually until converge.

Therefore, under this assumption, we could adapt the similarity measure in an orthogonal space to a non-orthogonal formulation. In contrast to the Cosine similarity which is more suitable for orthogonal space, we introduce our interactive similarity measurement in a non-orthogonal space,

$$S_o = B^T S_f B \text{ and } S_f = B S_o B^T .$$

We assume that the similarity measurement is normalized. In other words, the obtained similarity values should between zero and one. That is, if two objects are not similar at all, then the similarity between them should be zero; if two objects are the same, then the similarity between them should be one; otherwise, the similarity among objects should not less than zero and not larger than one. In order to satisfy this constrain and for convenience, we normalize our interactive similarity by two positive real parameters λ_1 , λ_2 and solve these similarity iteratively,

$$S_o^{k+1} = \lambda_1 B S_f^k B^T \text{ and } S_f^{k+1} = \lambda_2 B^T S_o^k B .$$

We will prove that if $\lambda_1 < \|B\|_\infty^2$, $\lambda_2 < \|B\|_1^2$, the entries of similarity matrices will between zero and one in the proof of lemma 2 in the appendix. However, since the similarity of the same object should be one, we can assign the diagonal of the similarity matrix a score of 1. Then we rewrite the recursive equations as:

$$S_o^{k+1} = \lambda_1 B^T S_f^k B + L_1^k \qquad (4)$$

$$S_f^{k+1} = \lambda_2 B S_o^k B^T + L_2^k \qquad (5)$$

where $L_1^k = I - diag(\lambda_1 B^T S_f^k B), L_2^k = I - diag(\lambda_2 B S_o^k B^T)$ and λ_1, λ_2 are positive real parameters which satisfy $\lambda_1 < \|B\|_\infty^2$, $\lambda_2 < \|B\|_1^2$. In this paper, we call (4) and (5) the basic Similarity equations in the Non-Orthogonal Space (SNOS). We choose initial value

$$S_{ij}^0 = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

for the interactive iteration process. In other words, we take $S_o^0 = S_f^0 = I$ to initialize this iteration algorithm. In fact, there are many other variations of our algorithm if we relax constrains of the parameters or give other initial values.

From the algorithm equation, it could be seen that the vector space, i.e. matrix *B,* has not been changed from the beginning to the end during the iteration process. In other words, we preserved the sparse structure of matrix *B* and thus save a lot of storage space compare with LSI. Furthermore the time complexity of our algorithm is $O(t \cdot m \cdot n)$, where *m* is the number of features, *n* is the number of objects and *t* is the number of iteration steps. The average iteration steps before converge is about 8 by our experiments. In contrast, in order to get the similarity matrix of features and objects, the time complexity of LSI is $O(n^3 + m^2)$ which is much higher than our proposed approach.

3.3 Convergence Proof

We give a proof summary of the existence and uniqueness for the basic SNOS equations (4) and (5). The detailed proof could be found in the appendix.

Definition 1 suppose matrices $A \in R^{m\times n}$, $B \in R^{p\times q}$, then their *Kronecker Product* $A \otimes B$ is,

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}_{mp\times nq}$$

Definition 2 the *Row-First Vectorization* of a matrix $A \in R^{m\times n}$, denoted as $\vec{A}$, could be represented as $\vec{A} = (a_1, a_2, \cdots, a_m)^T$, where $a_i \in R^n$, $i = 1, 2, \cdots, m$ are row vectors of *A*.

Lemma 1 supposes $A \in R^{m\times n}$ and $B \in R^{n\times n}$ are matrices, then $(\overrightarrow{ABA^T}) \in R^{m^2}$, moreover $(\overrightarrow{ABA^T}) = (A \otimes A)\vec{B}$.

Lemma 2 the similarity matrices S_o and S_f defined in the basic SNOS equations are bounded.

Lemma 3 the entries of similarity matrices S_o and S_f defined in the basic SNOS equations are non-decreasing.

Theorem 1 the interactive iteration basic SNOS equations converge to a unique solution.

Proof: from lemma 2 and 3 we know that S_o and S_f are bounded and non-decreasing, so they converge to some solution. Let's prove the uniqueness of the solution.

Suppose $(S_o, S_f), (S_o^{'}, S_f^{'})$ are two different group of solutions of the basic SNOS equations. Then,

$$\begin{cases} S_o = \lambda_1 B^T S_f B + L_1 \\ S_f = \lambda_2 B S_o B^T + L_2 \end{cases} \quad \text{and} \quad \begin{cases} S_o^{'} = \lambda_1 B^T S_f^{'} B + L_1^{'} \\ S_f^{'} = \lambda_2 B S_o^{'} B^T + L_2^{'} \end{cases}.$$

For all the non-diagonal elements, change the representation by lemma 1. Entries of $\vec{S}_f^k$ and $\vec{S}_o^{k+1}$ could be denoted as $s_f^k(l),\ s_o^{k+1}(g), l = 1,2,\cdots,n^2, g = 1,2,\cdots,m^2$. Suppose the element in S_o correspond to $s_o(l)$ is $s_o(i,j)$, we have

$$\begin{aligned} \left|s_o(l) - s_o^{'}(l)\right| &= \left\|\lambda_1 (b_i \otimes b_j)\vec{S}_f - \lambda_1 (b_i \otimes b_j)\vec{S}_f^{'}\right\| \\ &\le \left\|\lambda_1 (b_i \otimes b_j)\right\| \left\|(\vec{S}_f - \vec{S}_f^{'})\right\| \end{aligned}.$$

For all the diagonal elements

$$\left|s_o(l) - s_o^{'}(l)\right| = |1-1| = 0 \le \left\|\lambda_1 (b_i \otimes b_j)\right\| \left\|(\vec{S}_f - \vec{S}_f^{'})\right\|.$$

Then

$$\left\|(\vec{S}_o - \vec{S}_o^{'})\right\| < \left\|(\vec{S}_f - \vec{S}_f^{'})\right\|.$$

On the other hand

$$\left\|(\vec{S}_f - \vec{S}_f^{'})\right\| < \left\|(\vec{S}_o - \vec{S}_o^{'})\right\|.$$

This leads to the conclusion that $\vec{S}_o = \vec{S}_o^{'}, \vec{S}_f = \vec{S}_f^{'}$.

□

3.4 Parameter Selection

As mentioned above, several parameters are used in the original algorithm. For normalization purposes, we change the diagonal elements of similarity matrices into *1* at each iteration step through matrices L_1^k and L_2^k . Although this rough revision will not affect the convergence of algorithm, intuition tells us that too much of this operation will lead to reduced effectiveness of our algorithm. When choosing a parameter, we wish to minimize the norm of L_1 and L_2 after convergence occurs. In other words, we suggest to solve the parameters through the optimization problems listed bellow:

$$\lambda_1 = \underset{0 \le \lambda_1 < 1/\|B\|_\infty}{\arg\min} \|L_1\| = \underset{0 \le \lambda_1 < 1/\|B\|_\infty}{\arg\min} \left\|I - \lambda_1 diag(B^T S_f B)\right\|$$
$$\lambda_2 = \underset{0 \le \lambda_2 < 1/\|B\|_1}{\arg\min} \|L_2\| = \underset{0 \le \lambda_2 < 1/\|B\|_1}{\arg\min} \left\|I - \lambda_2 diag(B S_o B^T)\right\|$$

Note that the entries of matrices $diag(BS_fB^T)$ and $diag(BS_oB^T)$ are now bounded. Furthermore, $I - diag(BS_fB^T) \ge 0$ and $I - diag(BS_oB^T) \ge 0$, the optimization problem could be changed into,

$$\begin{aligned} \lambda_1 &= \underset{0 \le \lambda_1 < 1/\|B\|_\infty}{\arg\max} \left\|\lambda_1 diag(B^T S_f B)\right\| \\ \lambda_2 &= \underset{0 \le \lambda_2 < 1/\|B\|_1}{\arg\max} \left\|\lambda_2 diag(B S_o B^T)\right\| \end{aligned}.$$

This implicates that we should choose the parameters as large as possible under the constrains $0 \le \lambda_1 < 1/\|B\|_\infty$, $0 \le \lambda_2 < 1/\|B\|_1$.

For convenience, we choose a same parameter in our experiments. It must satisfies $0 \le \lambda < 1/\max\{\|B\|_1, \|B\|_\infty\}$, and as large as possible under this constrain. So we choose:

$$\lambda_1 = \lambda_2 = 0.9/\max\{\|B\|_1, \|B\|_\infty\}$$

in all our experiments.

4. EXPERIMENTS

In this section, we discuss the experimental data set, evaluation metric, and the experimental results based on cosine similarity, LSI, and our proposed SNOS. The first experiment is conducted on a synthetic data to demonstrate the drawbacks of cosine similarity and LSI, which are mentioned in the previous sections. The second experiment is performed on a real MSN click-through log data to find similar queries. LSI failed to finish this experiment due to the large scale of the data set. Our proposed SNOS achieves 80.6% improvement on the precision of similar queries. The third experiment is demonstrated the performance of the proposed algorithm for classification.

4.1 The Synthetic Data

We conduct the first experiment on a sample dataset consisting of the titles of 9 technical memoranda [4]. This dataset comes from the paper in which LSI was proposed. Terms occurring in more than one title are italicized. There are two classes of documents - five about human-computer interaction (c1-c5) and four about graphs (m1-m4). This dataset can be described by means of a term by document matrix B , where each cell entry indicates the frequency with a term in a document (Table 2).

Table 2. Technical Memo Example

c1	*Human* machine *interface* for Lab ABC *computer* applications
c2	A *survey* of *user* opinion of *computer system response time*
c3	The EPS user interface management system
c4	*System* and *human system* engineering testing of *EPS*
c5	Relation of *user*-perceived *response time* to error measurement
m1	The generation of random, binary, unordered *trees*
m2	The intersection *graph* of paths in *trees*
m3	*Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4	Graph *minors*: A *survey*

In this simple intuitive example, the Cosine similarity in VSM, LSI ($k = 2,3,\cdots 8$) and SNOS are used to compute the similarity between documents. Without lose of generality, the similarity between document *a* and all the collection is denoted as $sim(a,doc)$. We use $sim(c1,doc)$ to show the solution of different approaches, where the bold entries are abnormal. (Eg. the *5th* entry of vector $sim^{COS}(c1,doc)$ denotes the cosine similarity between *c1* and *c5*)

$$\begin{aligned}
sim^{COS}(c1,doc) &= (1 \quad 0.2 \quad 0.3 \quad 0.2 \quad \mathbf{0} \quad 0 \quad 0 \quad 0 \quad 0)\\
sim^{LSI(k=1)}(c1,doc) &= (1 \quad 1.0 \quad 1.0 \quad 1.0 \quad 1 \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{1} \quad \mathbf{1})\\
sim^{LSI(k=2)}(c1,doc) &= (1 \quad 0.9 \quad 1.0 \quad 1.0 \quad 0.9 \quad \mathbf{-0.2} \quad \mathbf{-0.2} \quad \mathbf{-0.2} \quad 0)\\
sim^{LSI(k=3)}(c1,doc) &= (1 \quad 0.6 \quad 1.0 \quad 1.0 \quad 0.2 \quad 0 \quad 0 \quad 0 \quad \mathbf{0.3})\\
sim^{LSI(k=4)}(c1,doc) &= (1 \quad 0.3 \quad 0.4 \quad 0.2 \quad \mathbf{0} \quad 0 \quad 0 \quad 0 \quad 0)\\
sim^{LSI(k=5)}(c1,doc) &= (1 \quad 0.2 \quad 0.4 \quad 0.2 \quad \mathbf{0} \quad 0 \quad 0 \quad 0 \quad 0)\\
sim^{LSI(k=6)}(c1,doc) &= (1 \quad 0.3 \quad 0.3 \quad 0.2 \quad \mathbf{0} \quad 0 \quad 0 \quad 0 \quad 0)\\
sim^{LSI(k=7)}(c1,doc) &= (1 \quad 0.3 \quad 0.3 \quad 0.2 \quad \mathbf{0} \quad 0 \quad 0 \quad 0 \quad 0)\\
sim^{LSI(k=8)}(c1,doc) &= (1 \quad 0.2 \quad 0.3 \quad 0.2 \quad \mathbf{0} \quad 0 \quad 0 \quad 0 \quad 0)\\
sim^{SNOS}(c1,doc) &= (1 \quad 0.5 \quad 0.4 \quad 0.2 \quad 0.1 \quad 0 \quad 0 \quad 0 \quad 0)
\end{aligned}$$

It can be seen that: (1) the Cosine similarity between c1 and c5 is zero although they are in the same predefined class. This is due to the non-orthogonal features of the input space; (2) there exists some negative values among the similarity solved by LSI, the meaning of which could not be intuitively understood; (3) the performance of LSI is crucially dependent on the parameter *k*. However, the selection of *k* is still an open issue in LSI. In contrast, our proposed SNOS can design the similarity effectually, in other words, the similarity values solved by SNOS are all between zero and one, truly reflecting the similarity values of the intra class objects are larger than those of inter class objects.

4.2 The MSN Search Click-thru Log Data

In this section, we compare SNOS with cosine similarity on the real MSN click-through logs data on the task of finding similar queries. It is noticed that LSI failed on this data set because SVD can not deal with the large scale, and sparse matrices. This is a predominance of our method than LSI.

4.2.1 Dataset

In order to study the effectiveness of SNOS for measuring the similarity of web objects, experiments are conducted on a real user query click-through log collected by the MSN Web search engine in December, 2003. It contains about 4.2 million query requests recorded sampled from a period of six hours. The log we obtained has already been processed into a predefined format, i.e. each query request is associated with the URL of one clicked web page. A single query (or web page URL) can occur multiple times in the query click-through log.

Before running the experiments, some preprocessing steps are applied to the queries and web page URLs. All queries are converted to lower-cases, stemmed by the Porter algorithm. The stop-words in the queries are removed. After these steps, the average query length is about 2.7 words. All URLs are converted into canonical form by performing such tasks as replacing unsafe characters with their escape sequences and collapsing sequences like "..\..". Each URL is considered as a feature, while each query is treated as an object. (We can also treat URLs as objects, and treat queries as features). Our proposed algorithm can solve similarities between objects and between features at the same iteration.) The weight for a query on a URL is the frequency of the query leading to the URL.

4.2.2 Evaluation Metrics

Since our proposed algorithm aims to find better similarity between objects, we developed an operational measure of precision to evaluate the performance. Given an object as input, we ask 10 volunteers to identify the correct similar objects from the top *N* returned results by each algorithm. The precision is defined as

$$Precision = \frac{|M|}{|N|} \qquad (6)$$

where $|N|$ is the number of top *N* similar objects to be evaluated, and $|M|$ is the number of correct similar objects tagged by the volunteers. The final relevance judgment for each object is decided by majority vote. In our experiment, $|N|$ is set as 10.

4.2.3 Finding Similar Queries

In this experiment, the volunteers were asked to evaluate the precision of results for the selected 10 queries (which are *air tickets, auto trader, bank of America, cannon cameras, Disney, mapquest, msn content, Presario 2100, united airlines, and weather report*). Figure 2 shows the comparison of the SNOS approach with cosine similarity. We found that SNOS outperforms the cosine similarity in precision by 80.6%.

Through careful study of the query "*Presario 2100*" which was a popular laptop model, we found that our proposed algorithm not only can filter some un-related queries, but also can find some close-related laptop models. In Table 3 the tag "Y" represents that the query is similar to the given query "*Presario 2100*"; "N" indicates "not similar". Although the cosine similarity returns some similar queries (the 1st – 5th results), it suffers from the "topic drift" issue (the 6th – 10th results), e.g. "*Presario 2100*" and "*Linux Compaq 2100*" share some clicked web pages, however, those web pages discuss how to install Linux in Presario 2100, and therefore, those common clicked web pages is actually a kind of "noise feature" for *"Presario 2100"*, which causes "*Linux Compaq 2100*" to be returned as a similar query by cosine similarity. On the other hand, SNOS finds out similar models which are not revealed by the cosine similarity. Although different models have many different clicked web pages, those clicked web pages are computed as "similar" since they are queried by "similar" queries, such as *"Compaq Presario"*, "*Compaq notebook", "Compaq laptop"*, etc. Hence, different models have higher similarity based on SNOS than based on cosine similarity.

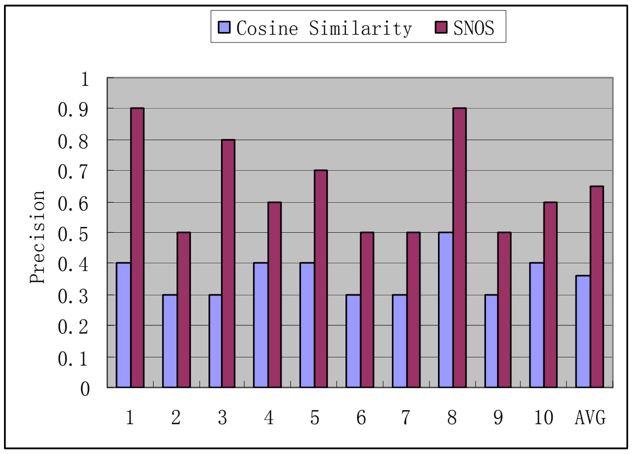

Figure 2 Precision of similarity between queries

Table 3. Similar queries for "Presario 2100"

	Cosine Similarity		SNOS	
1	Compaq Presario	Y	Compaq Presario	Y
2	Compaq notebook	Y	Compaq notebook	Y
3	Compaq laptop	Y	Compaq laptop	Y
4	online Compaq Presario	Y	online Compaq Presario	Y
5	Compaq Presario laptop	Y	Compaq Presario laptop	Y
6	compaque	N	Compaq Presario 9642	Y
7	Notebook price compare	N	Presario 2800 sale	Y
8	Compaq support	N	Compaq 2575us	Y
9	Linux Compaq 2100	N	Presario 9642	Y
10	Used Compaq reseller	N	Compaq batteries	N

4.3 The 20NG Data

To demonstrate the classification performance of the proposed algorithm, the commonly used 20NG data was used here *(http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html)*. We choose the five classes about computer which contains 4881 documents altogether. The Cosine similarity in non-orthogonal space was used as the baseline. We use the simple Nearest Neighbor classifier to assess the performance of SNOS on 20NG data. Figure 3 shows the error rate of Cosine similarity in contrast to our SNOS. The X-axis denotes the proportion of data used for training.

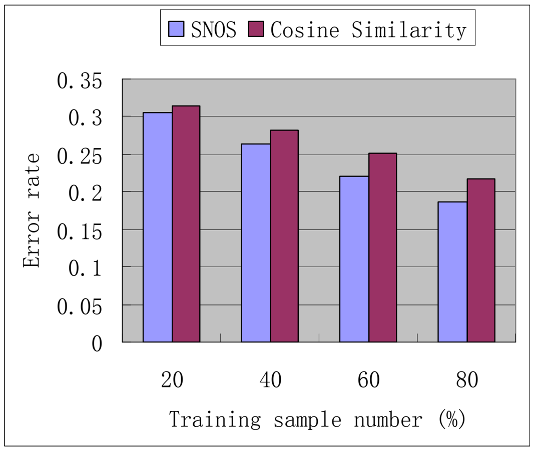

Figure 3 The classification error rate on 20NG dataset by the nearest neighbor classifier

5. CONCLUSIONS AND FUTURE WORKS

In this paper, we propose a novel approach to calculate the similarity metric in the non-orthogonal space (SNOS). In contrast to LSI, which aims at solving the non-orthogonal space similarity problem, our proposed approach has the following advantages: (1) lower cost both in storage computation (2) preserves the sparse structure of VSM by avoiding orthogonalizing the input space. We demonstrated the disadvantage of classical Cosine similarity in the non-orthogonal space and showed the performance of SNOS by experiments on a synthetic dataset as well as a large sampled MSN search click-thru log.

In the next step, we will explore the efficiency, effectiveness, and generality of the SNOS approach. We notice that there is considerable room for improvement in the quality of clustering of both Web pages and queries. We believe that SNOS could be successfully applied to improve the effectiveness of clustering. Finally, we plan to apply our approach to more complex settings, such as in digital libraries, where there are more types of objects, so as to demonstrate further the generality of SNOS. Besides, we will give some variations of SNOS in our future work.

6. ACKNOWLEDGEMENT

Jun Yan thanks the support of the Lab of Mathematics and Applied Mathematics of Peking University. Qiang Yang would like to thank Hong Kong RGC for their support.

7. REFERENCES

[1] Ando, R.K., *Latent Semantic Space: Iterative Scaling Improves Precision of Inter-document Similarity Measurement*. In Proceedings of the SIGIR, (Athens, Greece, 2000), 216--223.

[2] Atnafu, S., Brunie, L. and Kosch, H., *Similarity-Based Operators and Query Optimization for Multimedia Database Systems.* In Proceedings of the International Database Engineering and Application Symposium, (Grenoble, France, 2001), 346-355.

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

[4] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R.A. *Indexing by Latent Semantic Analysis*. Journal of the American Society for Information Science, *41(6)*. 391-407.

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

[6] Dumais, S.T., Platt, J., Heckerman, D. and Sahami, M.,

Inductive Learning Algorithms and Representations for Text Categorization. In Proceedings of the 7th International Conference on Information and Knowledge Management., (Bethesda, Maryland, 1998).

[7] Kandola, J., Taylor, J.S., Cristianini, N. and Davis., *Learning Semantic Similarity*. In Proceedings of the Neural Information Processing Systems (NIPS), (Whistler, B.C., 2003).

[8] Leopold, E. and Kinderman, J. *Text Categorization with Support Vector Machines. How to Represent Texts in Input Space?* Machine Learning, 46. 423-444.

[9] Salton, G. and McGill, M.J. *Introduction to Modern Retrieval*. McGraw-Hill Book Company, 1983.

[10] Sanders, I. and Kenny, L., *Heuristics for placing non-orthogonal axial lines to cross the adjacencies between*

orthogonal rectangles. In Proceedings of the 13th Canadian Conference on Computational Geometry, (2001), 153--156.

[11] Schultz, M. and Joachims, T., *Learning a Distance Metric from Relative Comparison.* In Proceedings of the Neural Information Processing Systems, (Whistler, B.C., 2003).

[12] Tosun, A.S. and Ferhatosmanoglu, H., *Vulnerabilities in Similarity Search Based Systems.* In Proceedings of the 11th ACM International Conference on Information and Knowledge Management (CIKM), (McLean,VA, 2002).

[13] Xing, E., Y., A., Jordan, M. and Russell, S., *Distance Metric Learning, with Application to Clustering with Side-Information.* In Proceedings of the Neural Information Processing Systems (NIPS), (Whistler, B.C., 2003).

[14] Zelikovitz, S. and Hirsh, H., *Using LSI for Text Classification in the Presence of Background Text.* In Proceedings of the 10th ACM International Conference on Information and Knowledge Management (CIKM), (Atlanta, US, 2001), ACM Press, New York, US, 113--118.

[15] Zhai, C. and Lafferty, J., *Model-based Feedback in the Language Modeling Approach to Information Retrieval.* In Proceedings of the 10th ACM International Conference on Information and Knowledge Management (CIKM*)*, (Atlanta, US, 2001), 403-410.

APPENDIX

Definition 3: the 1-norm of matrix $A=(a_{ij})\in R^{m\times n}$, is $\|A\|_1=\max_{1\le j\le n}\{\sum_{i=1}^{m}|a_{ij}|\}$.

Definition 4: the infinite-norm of matrix $A=(a_{ij})\in R^{m\times n}$, is $\|A\|_\infty=\max_{1\le i\le m}\{\sum_{j=1}^{n}|a_{ij}|\}$.

Lemma 1 for matrices $A\in R^{m\times n}$, $B\in R^{n\times n}$, the *Line-First Vectorization* of matrix ABA^T equal to a vector $(A\otimes A)\vec{B}\in R^{m^2}$.

Proof:

$$(A\otimes A)\vec{B}=\begin{pmatrix} a_{11}A & a_{12}A & \cdots & a_{1n}A \\ a_{21}A & a_{22}A & \cdots & a_{2n}A \\ \vdots & \vdots & & \vdots \\ a_{m1}A & a_{m2}A & \cdots & a_{mn}A \end{pmatrix}\begin{pmatrix} b_{11} \\ b_{12} \\ \vdots \\ b_{nn} \end{pmatrix}$$

$$=\begin{pmatrix} \sum_{i=1}^{m}\sum_{j=1}^{n} a_{1,i}a_{1,j}b_{i,j} \\ \sum_{i=1}^{m}\sum_{j=1}^{n} a_{1,i}a_{2,j}b_{i,j} \\ \vdots \\ \sum_{i=1}^{m}\sum_{j=1}^{n} a_{m,i}a_{m,j}b_{i,j} \end{pmatrix}_{mm\times 1} = (\overrightarrow{ABA^T})$$

□.

Lemma 2 the similarity matrices S_o, S_f defined in (1) and (2) are bounded.

Proof: from lemma 1, $\vec{S}_o^{k+1}=\lambda_1(B\otimes B)\vec{S}_f^k+\vec{L}_1^k$ (4).

Entries of $\vec{S}_f^k$ and $\vec{S}_o^{k+1}$ could be denoted as $s_f^k(l),\ s_o^{k+1}(g), l=1,2,\cdots,n^2, g=1,2,\cdots,m^2$. From the initial value, we know that $0\le s_f^0(l)\le 1$. Suppose the element in S_o^{k+1} correspond to $s_o^{k+1}(g)$ is $s_o^{k+1}(i,j)$.

Note that $s_o^{k+1}(g)=1$ when its corresponding element in matrix S_o^{k+1} is a diagonal element, i.e. $i=j$, *due* to $\vec{L}_1^k$. Otherwise, when $i\ne j$, if $0\le s_f^k(l)\le 1$, $l=1,2,\cdots,n^2$,

$$\begin{aligned} 0\le s_o^{k+1}(g)=\lambda_1(b_i\otimes b_j)^T\vec{S}_f^k &\le \frac{1}{\|B\|_\infty}(b_i\otimes b_j)^T\vec{S}_f^k \\ &\le \frac{1}{|b_i||b_j|}(b_i\otimes b_j)^T\vec{S}_f^k\le 1 \end{aligned},$$

where b_i is the i^{th} line of matrix B and $|b_i|$ is the number of non-zero elements in b_i. Then we could draw the conclusion that the entries of $\vec{S}_o^{k+1}$ belong to $[0,1]$ by induction.

In the same way, $0\le S_f^{k+1}\le 1$ for all *k*. □

Lemma 3 the entries of similarity matrices S_o, S_f defined in (1) and (2) are non-decreasing.

Proof: The same as the proof of lemma 2, we transform the algorithm into an equal formulation:

$$\begin{aligned} \vec{S}_o^{k+1} &= \lambda_1(B\otimes B)\vec{S}_f^k+\vec{L}_1^k \\ \vec{S}_f^{k+1} &= \lambda_2(B^T\otimes B^T)\vec{S}_o^k+\vec{L}_2^k \end{aligned}$$

It is obviously that $s_o^k(l)=1$ if it corresponds to a diagonal entry of matrix S_1 due to the effectiveness of $\vec{L}_1^k$. On the other hand, for all the non-diagonal entries, $s_o^1(l)\ge 0=s_o^0(l)$ since the initial value of $s_o^0(l)$ is zero. We could draw the conclusion that $S_o^1\ge S_o^0$.

If $S_o^k\ge S_o^{k-1}$, for all the non-diagonal entries $s_f^k(i,j)$ who correspond to $s_f^k(l)$, we have,

$$\begin{aligned} s_f^{k+1}(l)-s_f^k(l) &= \lambda_2(b_i\otimes b_j)\vec{S}_o^k-\lambda_2(b_i\otimes b_j)\vec{S}_o^{k-1} \\ &= \lambda_2(b_i\otimes b_j)(\vec{S}_o^k-\vec{S}_o^{k-1})\ge 0 \end{aligned}$$

For all the corresponding diagonal entries of S_f, we have $s_f^k(l)=1$.

From the discussion talked above, the entries of S_f are non-decreasing. Moreover, the entries of S_o are non-decreasing due to the same reason [3]. □