Latent Semantic Space: Iterative Scaling Improves Precision of Inter-document Similarity Measurement

Rie Kubota Ando
Department of Computer Science, Cornell University, Ithaca, NY 14853-7501, kubotar@cs.cornell.edu
IBM T. J. Watson Research Center, Saw Mill River Road, Hawthorne, NY 10532*

Abstract

We present a novel algorithm that creates document vectors with reduced dimensionality. This work was motivated by an application characterizing relationships among documents in a collection. Our algorithm yielded inter-document similarities with an *average precision* up to 17.8% higher than that of singular value decomposition (SVD) used for Latent Semantic Indexing. The best performance was achieved with dimensional reduction rates that were 43% higher than SVD on average. Our algorithm creates basis vectors for a reduced space by iteratively "scaling" vectors and computing eigenvectors. Unlike SVD, it breaks the symmetry of documents and terms to capture information more evenly across documents. We also discuss correlation with a probabilistic model and evaluate a method for selecting the dimensionality using log-likelihood estimation.

1 Introduction

This paper presents a novel algorithm that creates vector representations for natural language documents by reducing dimensionality. Our experimental results show that it yields higher similarity precision than singular value decomposition (SVD).

Similarities and differences among documents need to be measured in a variety of applications such as document clustering. The vector space model [18] is often used with an assumption that the "bag of words" approach works well. Among the approaches based on vector space, Latent Semantic Indexing (LSI) [5, 7] using SVD is a well-known successful approach applied to text analysis applications [8, 9, 10, 20], as well as information retrieval.

The algorithm that we propose was originally developed for a multiple document summarization program used for presenting query results. This summarization program [1] creates vectors that represent topics underlying a given document set. Given that source document sets were relatively small (e.g. top-ranked 100 documents), one of the requirements was to capture information from every document without ignoring any single one.

Our algorithm is closely related to SVD used for LSI. As in SVD, starting with a term-document matrix, it creates document vectors with reduced dimensionality by repeatedly computing eigenvectors. In fact, we were motivated by the observation that, using SVD, the topics underlying *outlier documents* (i.e. the documents very different from other documents) tended to be lost as we chose lower numbers of dimensions.

A general explanation of LSI's good performance is that when eigenvectors with smaller eigenvalues are left out, noise is eliminated; and as a result, the similarities among the linguistic units are measured more accurately than in the original space. According to the mathematical formulation of SVD, dimensional reduction comes from two sources: outlier documents, and minor terms. We note that these two kinds of noise are mathematically equivalent and inseparable under SVD. However, we do not want to consider the outlier documents as "noise" when our interest is in characterizing the relationships among the documents while all the documents are assumed to be equal. Thus, our algorithm differs from SVD in that terms and documents are treated in a nonsymmetrical way. By scaling[1] the vectors in each computation of eigenvectors, it tries to eliminate noise from the minor terms but not eliminate the influence of the outlier documents.

We evaluated our algorithm in terms of the precision of a cosine similarity measurement among the documents. Our algorithm yielded an average precision up to 17.8% higher than that of SVD.

There are studies of applying modified or generalized SVD to document representations. The semidiscrete decomposition (SDD) has been proposed to reduce the storage and computational costs of LSI [15]. It has been shown that user feedback can be integrated into LSI models by using the Riemannian SVD (R-SVD) [13]. Compared with these studies, the algorithm presented here differs in that it focuses on improving the precision of simi-

*Portions of this work were done while the author was at IBM T. J. Watson Research Center as a visitor.

[1] Note that this is essentially different from scaling or normalizing of vectors as a preprocess or a post-process of SVD mentioned in [5]. As shown later, it also differs from scaling of some factor analysis methods (see e.g. [16]) performed mainly for "standardizing".

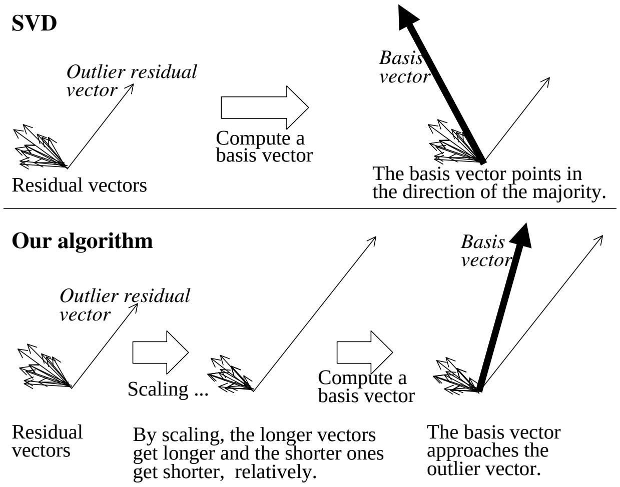

Figure 1: Intuitive difference between SVD and our algorithm. By scaling, the longer residual vector gets longer and the shorter ones get shorter, relatively. As a result, the basis vector computed next points closer to the longer residual vector.

larities among all the documents in a collection.

Several studies have provided theoretical interpretations of LSI in relation to the Bayesian regression model [19], Multidimensional Scaling (MDS) [3], and probabilistic models [6, 17]. A method for selecting the number of dimensions based on a subspace-based model and the minimal description length principle has been proposed [21]. Among these studies, we adopt the similarity-based probability model proposed by Ding [6] to analyze our approach and to find the optimal number of dimensions.

We describe the algorithm in Section 2. Section 3 discusses the algorithm in comparison to SVD and in relation to the Ding's probabilistic model. Section 4 presents the evaluation results; it also evaluates the probabilistic method of choosing the number of dimensions. We will conclude in Section 5.

2 Algorithm

The input to the algorithm is a term-document matrix of the conventional vector space model. From the term-document matrix, we create basis vectors for a reduced space. The document vectors with reduced dimensionality are created from the basis vectors and the term-document matrix.

2.1 Input: term-document matrix

Let n be the number of documents, and let m be the number of terms. A *term-document matrix* is an m-by-n matrix whose $[i, j]$th element is the degree of relevance of the ith term to the jth document. We call each column vector, associated with a document, a *term-document vector*. The term-document vectors should be length normalized so that every document is treated equally.

2.2 Basis vector and document vector creation

Basis vector creation is the critical process where our algorithm differs from SVD. Both our algorithm and SVD work by trying to find a smaller set of basis vectors for a reduced space. A rough overview of this basis vector creation process is as follows. First, an initial basis vector is chosen in such a way as to be "representative" of the document set. Then, the influence of this basis vector is subtracted from the document vectors, resulting in *residual vectors*. The residual vectors for documents that were not well-represented by the initial basis vector will be relatively large, whereas the residual vectors for documents that were well-represented will be somewhat small. Then, a new basis vector is chosen for these residual vectors, and the process is repeated.

Where our algorithm differs from SVD is as follows. Consider the situation pictured in Figure 1 where many small residual vectors point in essentially the same direction, but one (resulting from an "outlier" document) points in a different direction. In SVD, a basis vector will be picked that is representative of the majority of the documents, even though these documents have already been well-represented by some previous basis vectors. Thus, the outlier document will not be well-represented.

We attempt to compensate for this bias towards many small residual vectors by scaling outlier documents, thus amplifying their influence on the choice of the next basis vector, as shown in the bottom half of Figure 1. This is accomplished by the following procedure.

A matrix $\mathbf{R}$ is initialized by the term-document matrix[2]. $\mathbf{R}$'s column vectors keep the information not yet represented by any basis vectors (i.e. the $\mathbf{r}_i$'s are residual vectors). In each iteration, each residual vector is scaled by a power of its own length, and then the first eigenvector[3] of $\mathbf{R}_s \mathbf{R}_s^T$ (where $\mathbf{R}_s$ is a matrix of the scaled residual vectors) is chosen as the next basis vector.

At the end of each iteration, the information captured by the newly computed basis vector is subtracted from each residual vector. The basis vectors are orthogonal because this subtraction makes the residual vectors perpendicular to the previous basis vectors, and because the eigenvector computed next is a linear combination of the residual vectors.

After all the basis vectors are chosen, a term-document vector $\mathbf{d}_i$ is converted to the *reduced document vector* (the document vector in a reduced space) $\hat{\mathbf{d}}_i$ by multiplying the matrix of basis vectors following the standard method of orthogonal transformation.

The pseudo-code is shown in Figure 2. Its running time is proportional to the number of basis vectors, i.e. the dimensionality of the reduced space. Since eigenvector

[2] Throughout this paper, we denote a matrix by a bold-faced upper case letter, and its ith column vector by the bold-faced lower case letter with index i, unless defined otherwise. For instance, $\mathbf{a}_i$ denotes the ith column vector of a matrix $\mathbf{A}$.

[3] Following convention, we denote the eigenvector with the ith largest eigenvalue as the ith eigenvector.

Basic vector creation
Input: term-document matrix $\mathbf{D}$, scaling factor q
Output: basis vectors $\mathbf{b}_1, \mathbf{b}_2, \cdots$

$\mathbf{R} = \mathbf{D}$ /* Initialize a residual matrix by the term-doc matrix */
For ($i = 1$; until reaching some criterion; $i = i + 1$)
  $\mathbf{R}_s = [|\mathbf{r}_1|^q \mathbf{r}_1, \cdots, |\mathbf{r}_n|^q \mathbf{r}_n]$ /* scale each residual vector */
  $\mathbf{b}_i$ = the first unit eigenvector of $\mathbf{R}_s \mathbf{R}_s^T$
  $\mathbf{R} = \mathbf{R} - \mathbf{b}_i \mathbf{b}_i^T \mathbf{R}$ /* subtract out the $\mathbf{b}_i$ coordinate */
End for

Document vector creation
$\hat{\mathbf{d}}_i = [\mathbf{b}_1, \cdots, \mathbf{b}_k]^T \mathbf{d}_i$ /* reduce the dimensionality to k */

Figure 2: Basis vector and document vector creation.

computation is relatively expensive (see [11] for eigenvector computation), a small number of dimensions is advantageous.

There are two important variables in this algorithm: the number of dimensions and the *scaling factor* (denoted by k and q in the pseudo-code, respectively). Note that if the scaling factor is set to zero (i.e. the residual vectors are not scaled at all), the basis vectors become the left singular vectors of SVD. These two variables will be discussed in Section 3.

3 Discussion of the algorithm

Scaling of the residual vectors is the critical part of our algorithm that makes the resultant basis vectors different from the left singular vectors of SVD. In Section 3.1, we investigate the application of SVD to a term-document matrix; on this basis, we discuss how our algorithm differs from SVD. Section 3.2 analyzes the algorithm in relation to a probabilistic model.

3.1 SVD and our algorithm

It is known that a real m-by-n matrix $\mathbf{D}$ can be decomposed into three matrices (singular value decomposition), $\mathbf{D} = \mathbf{U\Sigma V}^T$. $\mathbf{\Sigma}$ is an m-by-n matrix such that $\sigma_i = \mathbf{\Sigma}[i, i]$ is the square root of the ith largest eigenvalue of $\mathbf{DD}^T$ (singular value), and $\mathbf{\Sigma}[i, j]$ is zero for $i \neq j$. $\mathbf{U}$ is an m-by-m orthonormal matrix of eigenvectors of $\mathbf{DD}^T$ (left singular vectors), in the order of eigenvalues, and $\mathbf{V}$ is an n-by-n orthonormal matrix of eigenvectors of $\mathbf{D}^T\mathbf{D}$ (right singular vectors) [12].

In the discussion below, we assume $\mathbf{D}$ is a term-document matrix, i.e. $\mathbf{D}$'s column vectors and row vectors represent the documents and terms respectively. We also assume that $m > n$ and $\mathbf{D}$'s rank is n since a typical term-document matrix (for a relatively small document set) takes this shape.

Regarding the first n left singular vectors as the basis vectors, a term-document vector with dimension m can be transformed into an n-dimensional vector. For document vectors, this transformation is the same as a rotation. Thus, the dimensionality of the document vectors can be reduced from m to n without losing (or gaining) any information, since the rank of n vectors' linear subspace is at most n. However, when we reduce the number of basis vectors to $k < n$, the transformation is no longer a rotation. For a given k, the first k left singular vectors make the optimal approximation of the linear subspace of the term-document vectors in terms of the least square errors or Frobenius matrix norm (F-norm)[4]

Here is an intuition that we need for investigating the left singular vectors. Generally, $\sum_{i=1}^{n} (\mathbf{x}^T \mathbf{a}_i)^2$, subject to $\mathbf{x}^T\mathbf{x} = 1$, is maximized when $\mathbf{x}$ is the first eigenvector of $\mathbf{AA}^T$; this is well known as a way of "fitting" $\mathbf{x}$ to the $\mathbf{a}_i$'s. This maximization can be explained as follows. Since $\mathbf{x}^T\mathbf{a}_i = |\mathbf{x}||\mathbf{a}_i| \cos_{(\mathbf{x},\mathbf{a}_i)}$, it increases as $\mathbf{x}$ approaches $\mathbf{a}_i$ (a larger cosine) and as $\mathbf{a}_i$ is longer. Therefore, *the first eigenvector* of $\mathbf{AA}^T$, which maximizes $\sum_{i=1}^{n} (\mathbf{x}^T \mathbf{a}_i)^2$, points *in the direction of the majority* of $\mathbf{a}_i$'s while also pointing somewhat *closer to the longer* $\mathbf{a}_i$'s.

The first left singular vector $\mathbf{u}_1$ is the first eigenvector of $\mathbf{DD}^T$. Using the intuition described above, we know that $\mathbf{u}_1$ points in the direction of the majority of the term-document vectors (assuming that the vectors are initially normalized to the same lengths; see Section 2.1). By subtracting the direction of $\mathbf{u}_1$ from the term-document vectors, we get residual vectors $\mathbf{r}_i$. The second left singular vector $\mathbf{u}_2$ is the second eigenvector of $\mathbf{DD}^T$, and also the first eigenvector of $\mathbf{RR}^T$ ($\mathbf{R}$ is a residual matrix). Using the intuition again, we know that the second left singular vector points in the direction of the majority of the residual vectors while being somewhat closer to the longer residual vectors. The process is repeated. Thus, as the left singular vectors are computed one by one, the document vectors having more nearby vectors are represented earlier. The outlier document vectors are represented later. Therefore, when low dimensionality is chosen, we lose more information from the outlier documents.

Because of symmetry of rows and columns, we know that the right singular vectors represent the term vectors having more nearby term vectors (resulting from the terms co-occurring with more terms) and the longer term vectors (resulting from the terms occurring more frequently) first. Other terms (*minor terms*) are represented later.

By dimensional reduction, we eliminate information from both the outlier documents and the minor terms at the same time. These are inseparable because SVD is essentially symmetric: $\mathbf{u}_i = (1/\sigma_i)\mathbf{D}\mathbf{v}_i$ and $\mathbf{v}_i = (1/\sigma_i)\mathbf{D}^T\mathbf{u}_i$. However, our interest is not necessarily symmetric (see Section 1). While minor terms can be regarded as "noise", we want to treat all the documents as equally important when our interest is in characterizing the relationships among the documents.

We wish to choose a small dimensionality k because it saves space and runtime cost. Using SVD, however, we

[4] $||\mathbf{A}||_F = \sqrt{\sum_{i,j} \mathbf{A}[i,j]^2}$

lose more information from outlier documents than the others when k is small. As a result, those outlier documents could be incorrectly regarded as similar to non-outlier documents. For instance, if we run a clustering algorithm on the resultant reduced document vectors, we would observe that small clusters just disappear.

Figure 1 illustrates the intuitive difference between SVD and our algorithm. We scale the length of each residual vector by multiplying it by a power of its own length. By doing this, the longer residual vectors get even longer, and the shorter ones get even shorter, relatively. (Suppose that x is the length ratio of a longer vector to a shorter one. We have $x < x^{q+1}$ for $q > 0$ since $x > 1$.) This scaling makes the basis vector closer to the longer residual vectors. A longer residual vector $\mathbf{r}_i$ (resulting in a shorter document vector $\hat{\mathbf{d}}_i$) means that the ith document is not well-represented by the basis vectors. By scaling appropriately, the longer residual vectors receive more attention. As a result, information is captured more evenly from all the documents, and we get a more even distribution of the reduced document vector lengths. This is how we break the symmetry of documents and terms in order to treat all the documents more equally.

3.2 Probabilistic model

A similarity-based probability model has been proposed as a theoretical explanation of LSI. In this section, we discuss correlation between our algorithm and this model.

First, we describe the outline of the model with the notation adjusted for our algorithm. The details may be found in [6]. The assumption is that, given the basis vectors $\mathbf{B}_k = \{\mathbf{b}_1, \cdots, \mathbf{b}_k\}$, the documents are distributed with the probability $p(\mathbf{d}|\mathbf{B}_k) = \exp(\sum_{j=1}^{k}(\mathbf{d}^T\mathbf{b}_j)^2)/Z(\mathbf{B}_k)$ where $\mathbf{d}$ denotes a document vector and $Z(\mathbf{B}_k) = \int \exp(\sum_{j=1}^{k}(\mathbf{x}^T\mathbf{b}_j)^2)d\mathbf{x}$ for normalization. This follows a Gaussian distribution when we regard $\mathbf{B}_k$ as the mean and an inner product as a measure of similarity. Assuming independence, the log-likelihood for the document vectors reduced to dimension k is computed as

$$\begin{aligned} l_k &= \log(\prod_{i=1}^{n} p(\mathbf{d}_i|\mathbf{B}_k)) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k}({\mathbf{d}_i}^T\mathbf{b}_j)^2 + (-n)\log(Z(\mathbf{B}_k)) \end{aligned}$$

Ding [6] has argued that larger log-likelihood indicates a better statistical model, (i.e. a better semantic space), and that SVD maximizes the first term of l_k while the second term is negligible because it changes very slowly compared to the first term.

3.2.1 Our interpretation

We note that l_k's second term ($(-n)\log(Z(\mathbf{B}_k))$), ignored by SVD, plays an important role in our algorithm.

The second term of l_k represents the probability of the occurrence of the term-document subspace when $\{\mathbf{b}_1, \cdots, \mathbf{b}_k\}$ are treated as the mean. Ding [6] estimated the second term as $(-n)\log(\sum_{i=1}^{n}\exp(\sum_{j=1}^{k}({\mathbf{d}_i}^T\mathbf{b}_j)^2))$ by taking the documents as unbiased data drawn from the population, and by ignoring $d\mathbf{x}$ treating it as being independent of k. Rewriting the estimated second term as $(-n)\log(\sum_{i=1}^{n}\exp(|\hat{\mathbf{d}}_i|^2))$, we note that, for $\sum_{i=1}^{n}|\hat{\mathbf{d}}_i|^2$ fixed, the estimation of the second term is maximized when $|\hat{\mathbf{d}}_1|^2 = \cdots = |\hat{\mathbf{d}}_n|^2$. In other words, the second term is maximized when the lengths of the resultant reduced document vectors are all the same. Thus, *the second term increases as information is preserved more evenly from all the documents.* Recall that this is exactly why our algorithm scales the residual vectors $\mathbf{r}_i$'s. Our algorithm tries to increase both the first and second terms of the log-likelihood estimation while the left singular vectors of SVD maximize only the first term, ignoring the second term.

We do not know an easy way to compute the basis vectors that maximizes the likelihood without ignoring the second term. To approximate the maximum likelihood, we could search for the scaling factor that will yield the larger log-likelihood estimation (hereafter, $\hat{l_k}$) in each computation. However, its computation is so expensive that we chose to fix the scaling factor in all the iterations. In the experiments we observed that, in general, our algorithm yielded larger values of $\hat{l_k}$ than SVD.

3.2.2 Choosing the optimal number of dimensions

So far, we have analyzed the log-likelihood estimation for a fixed dimensionality. To use it for selecting the optimal number of dimensions, we need to consider the estimated log-likelihood $\hat{l_k}$ for k ranging from 1 to n. We note that $\hat{l_k}$ is maximized when $k = n$, assuming that the term-document vectors are initially normalized to the same length. (When $k = n$, the lengths of the reduced document vectors are all the same, and therefore the estimation of the second term is maximized. The first term is also maximized since it increases monotonically in k.) However, the best performance is often produced when $k << n$, both by SVD and by our algorithm (we show this later in Section 4.2).

We conjecture that the estimation of $Z(\mathbf{B}_k)$ causes $\hat{l_k}$ to be incomparable for very different values of k, since the estimation of $Z(\mathbf{B}_k)$ is based on the simplifying assumption that $d\mathbf{x}$ is independent of k. If $\hat{l_k}$ and $\hat{l_h}$ are in fact comparable for k and h that are close enough, we expect that, instead of the global maximum, the local maximum of the log-likelihood will give a clue to the optimal number of dimensions. This expectation will be experimentally verified in Section 4.3.

4 Evaluation

This algorithm was originally developed for a multi-document summarization program [1] used for presenting query results. For this reason, it was designed with the following considerations: (1) it should model a relatively small document set (e.g. 100 top-ranked documents) treating every document equally and (2) it should run fast enough to be used in an interactive application. Based on these requirements, we evaluated our algorithm in terms of the precision of inter-document similarities and the rates of dimensional reduction.

We chose cosine as a similarity measurement because it was required for our application, and because another potential application is document clustering that generally assumes symmetry in measures [14]. However, it will be interesting to evaluate other similarity measurements such as inner products and Euclidian distances.

4.1 Experimental framework

4.1.1 Test data

For the evaluation, we used the Text REtrieval Conference (TREC) collections that were provided for formal training for SUMMAC. Twenty topics including extraterrestrial life, animal husbandry, and nuclear non-proliferation treaties were used[5]. We made two disjoint document pools, called 'pool1' and 'pool2', from the TREC documents relevant to exactly one of the twenty topics. The total number of the documents was 684.

We made 15 document sets from each of the two document pools (i.e. 30 document sets in total) by selecting the documents containing the same keyword[6]. This was a simplified simulation of the result that might be obtained by submitting a query to a search program. The number of documents for each set ranged from 31 to 126 with an average of 63, and the number of topics ranged from 6 to 20 with an average of 13. Relatively small document sets were used because of the original requirements for our algorithm. For each of the document sets, the algorithm created the document vectors in a reduced space.

4.1.2 Baseline algorithms

We evaluated two algorithms besides ours for the purpose of comparison. One was SVD taking the left singular vectors as the basis vectors. The other was the term-document vectors without any basis conversion. The term-document matrix was created for all the tested algorithms in the way described in Section 4.1.4.

4.1.3 Evaluation metrics

Our assumption is that similarity should be higher for any document pair relevant to the same topic (intra-topic pair) than for any pair relevant to different topics (cross-topic pair). This assumption is based on the consideration of how the document vectors would be used by the application. For instance, our multi-document summarization program tries to find clustered documents by detecting the document pairs having the largest similarities.

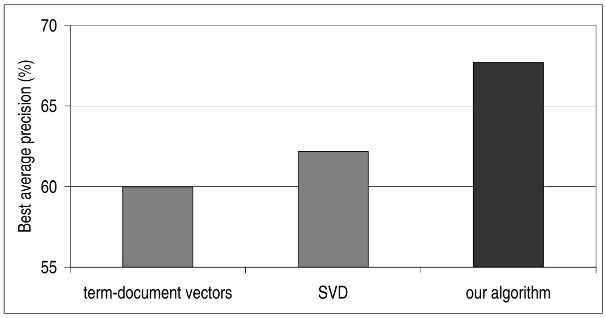

Figure 3: 'Best average precision' average over 30 document sets

We evaluated the *average precision* used in TREC, regarding an intra-topic pair as a relevant document and the similarity value as the ranking score. More formally, let p_i denote the document pair that has the ith largest similarity value among all pairs of documents in the document set. We evaluated precision for an intra-topic pair p_k by

$$precision(p_k) = \frac{\text{\# of intra-topic pairs } p_j \text{ where } j \leq k}{k}$$

The average of the precision values over all intra-topic pairs was computed as the *average precision*.

4.1.4 Implementation

We extracted single- and multi-word terms from the document sets using TALENT 4.1 (a document processing environment; see [4]). We removed the functional words and very common words (e.g. "say") as stop words.

We observed that, by removing proper nouns, the average precision was improved for all the tested algorithms while performance trends were maintained. Since we are interested in the optimal performance we could get, we will report only the performance with the proper nouns removed.

Dumais [7] reported that the performance of LSI was affected (both positively and negatively) by the selection of term weighting methods used to create the term-document matrix. To avoid the influence of term weighting methods, we simply used the term frequency.

4.1.5 Scaling factor

To determine the scaling factor, parameter training was performed. For the test of the document sets from 'pool1', the document sets from 'pool2' were used as training data, and vice versa. As described in Section 4.1.1, 'pool1' and 'pool2' were disjoint. The parameter space of the scaling factor from 1 to 10 was explored, in increments of 1.

4.2 Results

To treat the selection of dimensions as a separate issue, in this section we will report the best average precision over all the possible numbers of dimensions, hereafter 'best average precision'.

As a measurement of how much information is transferred from the term-document vectors to the reduced document vectors, we define *preservation rate* to be f^2/n

[5]Specifically, topic number 202, 205, 217, 220, 225, 236, 238, 240, 246, 250, 252, 253, 259, 260, 263, 264, 277, 293, 294, and 299 were used.

[6]'law', 'attack', 'citizen', 'crime', 'evidence', 'food', 'foreign', 'illegal', 'intelligence', 'life', 'mine', 'newspaper', 'nuclear', 'power', and 'terrorist' were arbitrarily chosen from the topic descriptions.

term-			red.rate$>50\%$	
doc	SVD	Ours	SVD	Ours
70.9	0.0	**+17.8**	-7.6	**+17.8**
55.1	+0.6	**+14.0**	-7.7	**+14.0**
59.4	0.0	**+13.5**	-8.1	**+13.5**
62.3	+3.6	**+13.1**	+3.6	**+13.1**
66.3	+0.2	**+12.3**	-2.1	**+12.3**
54.7	0.0	**+11.2**	-2.7	**+11.2**
57.9	+0.1	**+11.1**	-2.7	**+11.1**
59.1	+5.0	**+10.9**	+5.0	**+10.9**
71.7	0.0	**+9.6**	-0.2	**+9.6**
52.2	+6.8	**+9.6**	+6.8	**+9.6**
71.7	+5.5	**+9.1**	+5.5	**+9.1**
80.1	0.0	**+8.0**	-6.6	**+8.0**
67.9	+6.5	**+8.0**	+6.5	**+8.0**
45.4	+0.1	**+7.8**	-0.8	**+7.8**
53.2	+5.4	**+7.3**	+5.4	**+7.3**
48.0	+3.3	**+7.3**	+3.3	**+7.3**
34.3	+2.9	**+7.1**	+2.9	**+7.1**
62.8	+1.7	**+6.9**	+1.7	**+6.9**
67.5	+6.6	**+6.7**	+6.6	**+6.7**
54.0	0.0	**+6.5**	-9.4	**+6.5**
55.3	0.0	**+6.2**	-6.8	**+6.2**
67.3	+0.2	**+5.4**	-4.3	**+5.4**
68.0	+0.9	**+5.2**	-7.6	**+1.2**
73.6	+0.2	**+3.8**	-15.4	**+3.8**
61.3	0.0	**+3.6**	-3.7	**+3.6**
56.6	0.0	**+3.3**	-5.0	**+3.3**
67.6	**+4.1**	+2.6	**+4.1**	+2.6
59.0	+0.2	**+2.0**	-6.2	**+0.8**
65.6	**+11.7**	+1.5	**+11.7**	+1.4
30.4	**+1.0**	+0.3	**+0.3**	-4.1

Figure 4: Best average precision for each document set (sorted by the third column). For SVD and our algorithm, relative performance with respect to term-document vectors is shown. Boldface indicates the best performing algorithm. The right half is the best average precision where the reduction rate $>50\%$, and the degraded performances are underlined.

where f is the F-norm of the matrix of the reduced document vectors. (Recall that the left singular vectors yield the optimal approximation of the document subspace in terms of F-norm.) The *reduction rate* is its complement, $1-$ preservation rate. We also define the *dimensional reduction rate* as $1-$ (# of dimensions / max # of dimensions).

Figure 3 shows the 'best average precision' on average over 30 document sets. Our algorithm's performance was 5.5% higher than SVD and 7.7% higher than the term-document vectors.

We are aware that the absolute performance largely depends on the document set. Just as information retrieval research observes that there are easy topics and hard ones [2], we observe that there exist easy document sets and hard ones. The left half of Figure 4 shows the best average precision for each document set. For our algorithm and SVD, relative performance with respect to the term-document vectors is shown. For 27 document sets out of 30, the best average precision of our algorithm was better than both baseline algorithms by up to 17.8%. For 3 document sets, it fell between SVD and the term-document vectors.

Note that, theoretically, the best average precision for SVD and our algorithm should never be worse than the one for the term-document vectors. This is because, when the preservation rate is one, the dimensional reduction does not change the cosine values between the document vectors (as mentioned in Section 3.1). However, our interest is in getting good performance with as high a reduction rate as possible. For this purpose, we evaluated the best average precision in the range of higher reduction rates. The right half of Figure 4 shows the best average precision where the reduction rate is higher than 50%. The performance was degraded for only 4 document sets by our algorithm, while it was degraded for 18 document sets by SVD.

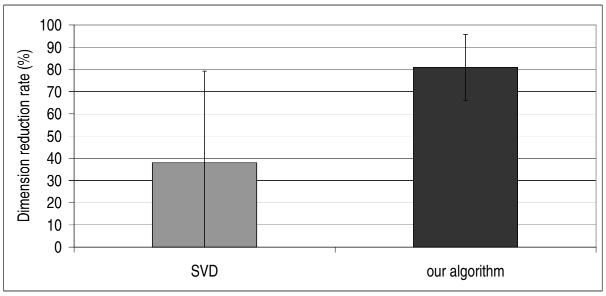

Figure 5: Reduction rates that yielded the 'best average precision'. The average over 30 document sets is shown for each algorithm. The "error bars" represent one standard deviation.

Figure 5 shows the reduction rates that yielded the best average precision. Our algorithm shows a 35.8% higher reduction rate than SVD on average. The relatively small standard deviation for our algorithm indicates that training could work well to find the optimal dimensions.

The algorithm's computational efficiency is dependent on the number of dimensions computed. Our algorithm yielded the best average precision at a dimensional reduction rate 43.0% higher than that of SVD on average. Thus, our algorithm has the advantage of being computationally inexpensive, assuming that we can find the optimal number of dimensions.

Figure 6 shows the average precision and precision-recall curve for the document sets for which our algorithm yielded the best and worst performance with respect to SVD. The precision-recall curve is the one for the dimensionality that yielded the best average precision. 'Best1' and 'Best2' show that the performance of our algorithm is better than baseline algorithms for almost all the preservation rates and recall levels. We conjecture that the scal-

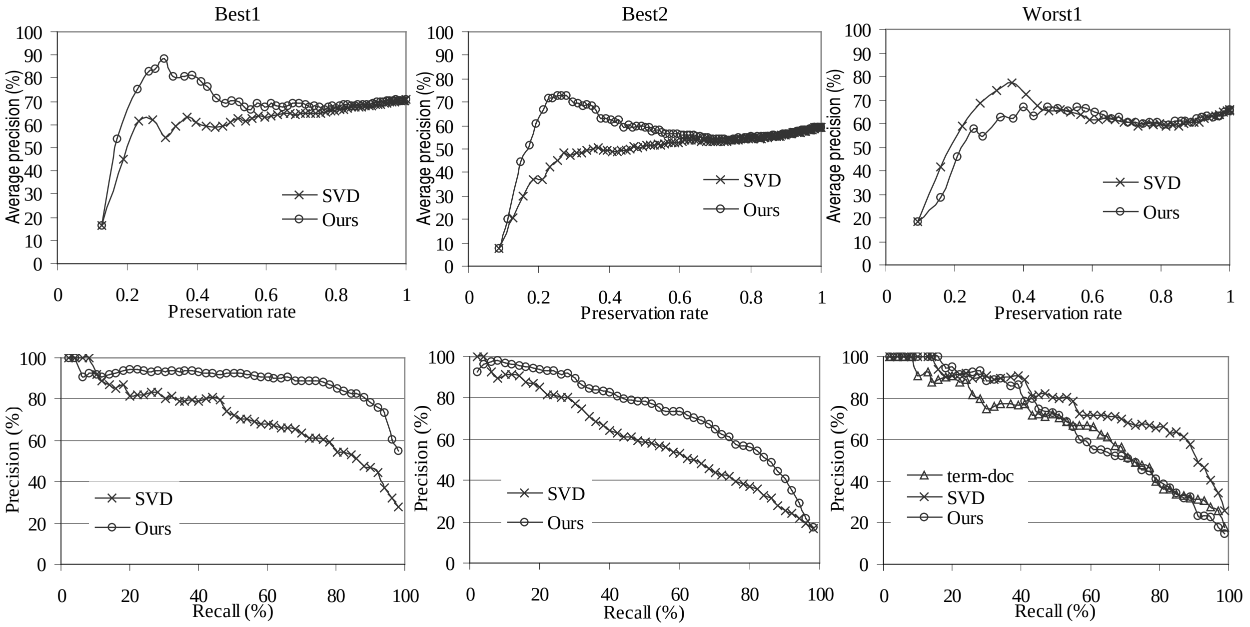

Figure 6: Examples of average precision and precision-recall curve.
The performances for the document sets for which our algorithm did best and worst w.r.t. SVD are shown. The average precision of the term-document vectors are the same as that of SVD (and our algorithm) when preservation rate $= 1$. The precision-recall curves for the term-document vectors are not shown for 'Best1' and 'Best2' because they almost overlapped with the ones for SVD.

ing factor was not appropriate for 'Worst1'. Choosing the scaling factor without training will be our future work.

4.3 Selection of dimensions by log-likelihood

In this section, we describe and evaluate a method that uses the log-likelihood for choosing the number of dimensions.

For the purposes of comparison, we evaluated two more methods: a training-based method and a random guess-based method. We chose to evaluate the training-based method because, according to [21], in practice the number of dimensions is often determined experimentally for LSI. The random guess-based method was a baseline.

Log-likelihood method: Based on the discussion in Section 3.2, we take the dimensionality that yields a larger log-likelihood estimation than its neighbors. To do so, we compare the estimated log-likelihood for k and an average estimated log-likelihood of k's neighbors by computing

$$f(k) = \hat{l_k} - (\sum_{i=k-c*n/2}^{k-1} \hat{l_i} + \sum_{i=k+1}^{i=k+c*n/2} \hat{l_i})/(c*n)$$

Here n is the number of documents, and c is a parameter to adjust the number of neighbors. The dimension numbers that give larger $f(k)$ are preferred. For the evaluation, $c = 0.25$ was used.

Training-based method: Let p be the average of the preservation rates that yielded the best average precision on the training data. The numbers of dimensions that yield preservation rates closer to p are preferred. For the evaluation, the training was performed using the document sets from 'pool1' as training data for the test of the document sets from 'pool2', and vice versa.

Random guess-based method: We determine the number of dimensions randomly.

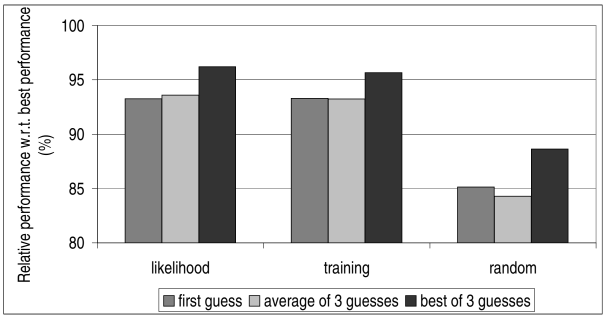

Figure 7: Ratios of average precision to the best average precision. The average over 30 document sets is shown for each method.

4.3.1 Evaluation

Each method made three guesses. The relative performance (average precision / best average precision) was evaluated for the numbers of dimensions that each method guessed. Figure 7 shows the performance of the first guess, the average performance over three guesses, and the best performance among three guesses.

The log-likelihood method clearly does better than the random guess-based method and almost same or slightly better than the training-based method. We believe that this result supports the correlation between our algorithm and the probabilistic model based on Gaussian distribution.

5 Conclusion

We presented an algorithm that creates document vectors via dimensional reduction for a relatively small document set. The experimental results showed that our algorithm achieved higher precision of similarity measurement with higher reduction rate than the baseline algorithms. The best average precision was up to 17.8% higher than SVD and the original term-document vectors. The dimensional reduction rate yielding the 'best average precision' was 43% higher than SVD on average. The experimental results indicated that the log-likelihood based on the Gaussian distribution would support our approach and suggest how to select the optimal number of dimensions.

We expect that dynamic determination of scaling factors will further improve the performance. We also plan to apply our algorithm to other smaller linguistic units.

Acknowledgements

We thank Branimir Boguraev, Roy Byrd, Herb Chong, James Cooper, Lillian Lee, Alan Marwick, Mary Neff, John Prager, Dragomir Radev, and Edward So for helpful discussions, and the anonymous reviewers for their comments and suggestions. The author was partly supported by a McMullen fellowship from Cornell University.

References

[1] R. K. Ando, B. K. Boguraev, R. J. Byrd, and M. S. Neff. Multi-document summarization by visualizing topical content. In *Proceedings of ANLP/NAACL 2000 Workshop on Automatic Summarization*, 2000.

[2] D. Banks, P. Over, and N.-F. Zhang. Blind men and elephants: Six approaches to TREC data. *Information Retrieval*, 1:7–34, 1999.

[3] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Latent Semantic Indexing is an optimal special case of Multidimensional Scaling. In *Proceedings of SIGIR'92*, pages 161–167, 1992.

[4] B. Boguraev and M. Neff. Discourse segmentation in aid of document summarization. In *Proceedings of Hawaii International Conference on System Sciences (*HICSS-33*), Minitrack on Digital Documents Understanding*, Maui, Hawaii, 2000. IEEE.

[5] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the Society for Information Science*, 41:391–407, 1990.

[6] C. H. Ding. A similarity-based probability model for Latent Semantic Indexing. In *Proceedings of SIGIR'99*, pages 1:58–65, 1999.

[7] S. T. Dumais. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments, & Computers*, 23(2):229–236, 1991.

[8] S. T. Dumais and J. Nielsen. Automating the assignment of submitted manuscripts to reviewers. In *Proceedings of SIGIR'92*, pages 233–244, 1992.

[9] P. W. Foltz and S. T. Dumais. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12):51–60, 1992.

[10] P. W. Foltz, W. Kintsch, and T. K. Landauer. The measurement of textual coherence with Latent Semantic Analysis. *Discourse Processes*, 25(2&3):285–307, 1998.

[11] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Baltimore and London: The Johns Hopkins University Press, third edition, 1996.

[12] M. T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill, 1996.

[13] E. P. Jiang and M. W. Berry. Information filtering using the Riemannian SVD (R-SVD). In *Proceedings of IRREGULAR'98*, pages 386–395, 1998.

[14] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley series in probability and mathemetical statistics. John Wiley and Sons, New York, 1990.

[15] T. G. Kolda and D. P. O'Learly. A semidiscrete matrix decomposition for Latent Semantic Indexing in information retrieval. *ACM Transactions on Information Systems*, 16(4):322–346, 1998.

[16] J. B. Kruskal. Factor Analysis and Principal Components. In *International Encyclopedia of Statistics*. New York: Free Press, 1978.

[17] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent Semantic Indexing: A probablistic analysis. In *Proceedings of Symposium on Principles of Database Systems (PODS)*. ACM Press, 1998.

[18] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[19] R. E. Story. An explanation of the effectiveness of Latent Semantic Indexing by means of a Bayesian regression model. *Information Processing & Management*, 32(3):329–344, 1996.

[20] M. B. W. Wolfe, M. E. Schreiner, B. Rehder, and D. Laham. Learning from text: Matching readers and text by Latent Semantic Analysis. *Discourse Processes*, 25:309–336, 1998.

[21] H. Zha, O. Marques, and H. D. Simon. Large-scale SVD and subspace-based methods for information retrieval. In *Proceedings of IRREGULAR'98*, pages 29–42, 1998.