A Study of Semi-Discrete Matrix Decomposition for LSI in Automated Text Categorization·

Wang Qiang Wang XiaoLong Guan Yi

School of Computer Science and Technology, Harbin Institute of Technology, Harbin
{qwang, wangxl , guanyi }@insun.hit.edu.cn

Abstract. This paper proposes the use of Latent Semantic Indexing (LSI) techniques, decomposed with semi-discrete matrix decomposition (SDD) method, for text categorization. The SDD algorithm is a recent solution to LSI, which can achieve similar performance at a much lower storage cost. In this paper, LSI is used for text categorization by constructing new features of category as combinations or transformations of the original features. In the experiments on data set of Chinese Library Classification we compare accuracy to a classifier based on k-Nearest Neighbor (k-NN) and the result shows that k-NN based on LSI is sometimes significantly better. Much future work remains, but the results indicate that LSI is a promising technique for text categorization.

1 Introduction

Text categorization, the assignment of free text documents to one or more predefined categories based on their content, is an important component in many information management tasks [1]. Now a variety of text categorization for supervised learning algorithms is based on vector space model (VSM). In this model documents are represented as a set of index terms that are weighted according to their importance for a particular document and for the general collection. But it can be misleading, because a document can be relevant for a test document without having any terms in common with it.

This paper explores the use of Latent Semantic Indexing (LSI) for text categorization as an improvement to VSM. The idea behind the LSI is to map each document and test vector into a lower dimensional space which is associated with concepts and compare the documents in this space [2]. We performed experiments using k-NN LSI, a new combination of the standard k-NN method on top of LSI, and applying a new matrix decomposition algorithm, Semi-Discrete Matrix Decomposition, to decompose the vector matrix. The Experimental results show that text categorization effectiveness in this space will be better and it will also be computationally less costly, because it needs a lower dimensional space.

[1] This investigation was supported by the National Natural Science Foundation (Harbin 60175020) and The high Technology Research and Development Programme (Harbin 2002AA117010-09)

This paper proceeds as follows. Section 2 presents the general framework for a recent LSI method, Semi-Discrete Matrix Decomposition (SDD). Then, the specific application of LSI to text classification is discussed in Section 3. Related work is presented in Section 4. Experimental result is shown in Section 5. Finally, Section 6 makes out plans for future work.

2 Latent Semantic Indexing

Current methods of VSM in indexing and retrieving documents from databases usually depend on a lexical match between query terms and keywords extracted from documents in a database. These methods can produce incomplete or irrelevant results due to the use of synonyms and polysemous words. In fact the association of terms with documents or implicit semantic structure can be derived using large sparse term by document matrices. So both terms and documents can be matched using representations in k-space derived from k of the largest approximate singular vectors of these terms by document matrices. This completely automated approach called Latent Semantic Indexing (LSI), which uses subspaces spanned by the approximate singular vectors to encode important associative relationships between terms and documents in k-space [3]. Using LSI, two or more documents may be close to each other in k-space yet share no common terms.

2.1 Singular Value Decomposition (SVD) for LSI

SVD is the most common method to LSI which decompose a term-by-document rectangular matrix X into the product of three other matrices: $A = U \sum V^T$, Where U (M × R) and V (R × N) have orthonormal columns and Σ (R×R) is the diagonal matrix of singular values. R≤min(M,N) is the rank of A. If the singular values of Σ are ordered by size, the K largest may be kept and the remaining smaller ones set to zero. The product of the resulting matrices is a matrix A_k which is an approximation to A with rank K:

$$A_k = U_k \sum_K V_k^T \quad \textbf{(1)}$$

Where Σ k (K×K) is obtained by deleting the zero rows and columns of Σ, and U_k (M×K) and V_k (N × K) are obtained by deleting the corresponding rows and columns of U and V [4].

A_k in one sense captures most of the underlying structure in A, yet at the same time removes the noise or variability in word usage. Since the number of dimensions K is much smaller than the number of unique words M, minor differences in terminology will be ignored.

2.2 Semi-Discrete Matrix Decomposition (SDD) for LSI

A semi-discrete decomposition (SDD) approximates a matrix as a weighted sum of outer products formed by vectors with entries constrained to be in the set S = {-1, 0, 1}. O'Leary and Peleg introduced the SDD in the context of image compression, and Kolda and O'Leary (1998, 1999) used the SDD for latent semantic indexing (LSI) in information retrieval. The primary advantage of the SDD over other types of matrix approximations such as the truncated singular value decomposition (SVD) is that, it typically provides a more accurate approximation for far less storage.

An SDD of an m×n matrix A is a decomposition of the form:

$$A_k = \underbrace{[u_1 u_2 \cdots u_k]}_{u_k} \cdot \underbrace{\begin{bmatrix} e_1 & 0 & \cdots & 0 \\ 0 & e_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e_k \end{bmatrix}}_{\Sigma_k} \cdot \underbrace{\begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix}}_{v_k^T} \tag{2}$$

Here each u_i is an m-vector with entries from the set S = {-1, 0, 1}, each v_i is an n-vector with entries from the set S, and each e_i is a positive scalar. We call this a k-term SDD.

Although every matrix can be expressed as a mn-term SDD:

$$A = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} e_i e_j^T \tag{3}$$

Where e_k is the k-th unit vector, the usefulness of the SDD is in developing approximations that have far fewer terms.

An SDD approximation can be formed iteratively via a greedy algorithm. Let A_k denote the k-term approximation ($A_0 \equiv 0$). Let R_k be the residual at the kth step, that is $R_k = A - A_{k-1}$. Then the optimal choice of the next triplet (d_k , x_k , y_k) is the solution to the sub problem:

$$\min \; F_k(d, x, y) \equiv \left\| R_k - dxy^T \right\|_F^2 \quad \text{s.t. } x \in S^m, y \in S^n, d > 0 \tag{4}$$

This is a mixed integer programming problem. Assuming that a fixed number of inner iterations are set per step, the complexity of the algorithm is $O(k^2(m+n) + m\log m + n\log n)$. At the meanwhile, since the storage requirement for a k-term SDD is k floating point numbers plus k (m+n) entries from S, it is also inexpensive to store quite a large number of terms [5].

In evaluating queries, a document vector can be treated as

$$\widetilde{A} = \Sigma_k^{1-\alpha} V_k^T \tag{5}$$

This is a k-dimension vector. And the test vector is projected into the same k-dimensional space by:

$$\widetilde{q} = \Sigma_k^{\alpha} U_k^T q \tag{6}$$

The similarity between a document and test vector can be calculated by

$$S = \cos(\tilde{q}^T, \tilde{A}) = \frac{\left|\tilde{q}^T \cdot \tilde{A}\right|}{\sqrt{\sum_{i=1}^{k} (\tilde{q}^T)^2 (\tilde{A})^2}} \quad (7)$$

In this study, the value of the splitting parameter α in equation has left at the default 0.

As in the Vector Model, documents can now be ranked according to their similarity to the test document, and the category of the query is the category of the most similar document.

3 LSI for Text Categorization

The k-NN method is a very simple approach that has previously shown very good performance on text categorization tasks [6][7]. Hence, we decided to use this method for classification. In this paper, we apply LSI model to the k-NN algorithm to verify the improvement on text categorization. To classify an unknown document vector d, the k-nearest neighbor (k-NN) algorithm ranks the document's neighbors among the training document vectors, and uses the class labels of the k most similar neighbors to predict the class of the input document. The classes of these neighbors are weighted by the similarity of each neighbor to d, where similarity may be measured by such as the Euclidean distance or the cosine between the two document vectors.

The k-NN LSI algorithm has four steps:

1. Index the training set

2. Use SDD to map each document vector into a lower dimensional space which is associated with concepts and process the documents in this space.

3. For each document $\vec{x}$ to be classified, retrieve its k most similar documents from the training set (where k is a parameter of the algorithm). Call this set $R_k(\vec{X})$.

4. For each category C, compute its relevance to $\vec{x}$ as:

$$S(c, \vec{x}) = \sum_{\vec{d} \in R_k(\vec{x}, c)} sim(\vec{d}, \vec{x}) \quad (8)$$

Where $R_k(\vec{x}, C)$ is the subset of documents in $R_k(\vec{x})$ that are relevant to C.

There are many ways to transform the scores $S(c, \vec{x})$ for a particular category-document pair into a YES/NO decision on whether to assign that document to that category. In this paper, we use the methods called SCut. SCut assigns to each category a threshold t(C) and assigns a document $\vec{x}$ to category C if $S(c, \vec{x}) \geq$ t(C). The choice method of t(C) is explained in section 5.1.

4 Related Work

Another study has been performed using LSI for text classification. It is made by Ana Cardoso-Cachopo and Arlindo Limede Oliveira(2000). In a comparison between k-NN LSI and vector model, and using the Mean Reciprocal Rank (MRR) as a measure of overall performance, this study proved that k-NN LSI performed almost as well as the best performing methods, such as Support Vector Machine (SVM), for text categorization. But his study is confined to English corpus and SVD technique is adopted.

5 Experiments

This section provides some empirical evidence to show that LSI is a competitive solution to text classification. The results are examined on the data sets of Chinese Library Classification.

5.1 Data Sets and Protocol

The experiment operates Chinese text as processing object and uses the Chinese Library Classification 4 (Simplified Version) as criteria (Table 1), which is a comprehensive one in common use in China's most libraries, information institutes and centers. All the Data Sets are gathered from digital library and Internet. The web pages, all together containing 10,857 pages, are divided into thirty-seven categories.

Table 1. Chinese Library Classification

A Marxism, Leninism, Maoism & Deng Xiao ping's Theory	**K** History and Geography	**TF** Metallurgy Industry	**TS** Light industry and Handicraft
B Philosophy and Religion	**N** Natural Science	**TG** Metal and Metalworking Tech-	**TU** Architecture Science
C Social Science	**O** Mathematics, Physics and Chemistry	**TH** Machine and Meter	**TV** Water Conservancy
D Politics and law	**P** Astronomy and Geosciences	**TJ** Weaponry	**U** Transportation
E Military Science	**Q** Bioscience	**TK** Kinetics Industry	**V** Aviation
F Economics	**R** Medicine and Hygiene	**TL** Atomic Energy	**X** Environmental Science
G Culture, Science, Education and Athletics	**S** Agricultural Science	**TM** Electro technician	**Z** Comprehensive Books
H Linguistics	**TB** Industrial Technology	**TN** Radio electronics Tele-technology	

I Literature	**TD** Mining Engineering	**TP** Automation technology Computer Science	
J Art	**TE** Petroleum and Natural gas Industry	**TQ** Chemistry Industry	

The corpus is broken into words with Word-Lattice algorithm and after removing tokens that occur only once or are on a stoplist, a vocabulary of size 57,040 is left. We set 9,115 pages as the train set and others 1,742 as test set. In the stage of training, the Expected Cross Entropy (ECE) is used on train set for feature selection, which is defined as

$$ECE\ (T) = \sum_i p(c_i \mid T) \log \frac{p(c_i \mid T)}{p(c_i)} \quad (9)$$

Where $p(c_i \mid T)$ is the probability of term T and category c_i co-occurrence and $p(c_i)$ is the probability of category c_i. In the process of turning documents into vectors, the term weights are computed using a variation of the Okapi term-weighting formula [8]:

$$w(t,\vec{d}) = \frac{tf(t,\vec{d})}{0.5+1.5*\frac{len(\vec{d})}{avg\_len}+tf(t,\vec{d})} \times \log\left(\frac{0.5+N-n(t)}{0.5+n(t)}\right) \quad (10)$$

Where $w(t,\vec{d})$ is the weight of term t in document $\vec{d}$; $tf(t,\vec{d})$ is the within-document frequency of term t; N is the number of documents in the training set; n(t) is the number of training documents in which t occurs; $len(\vec{d})$ is the number of tokens in document $\vec{d}$ after stop-word removal; avg_len is the average number of tokens per document in the training set. The values of N, n(t), and avg_len were computed from the entire training set.

Before the k-NN algorithm can be used, the value of k must be set. We used standard m-way cross-validation to set this value; the training data was split into m partitions, with documents assigned randomly to each partition. For each cross-validation run, m_{tr} of these partitions formed the training subset and m_{va} (= m - m_{tr}) partitions the validation subset. Partitions were rotated between the training and validation subsets so that each partition was used m_{tr} times for training and m_{va} times for validation. Performance was averaged over all m runs to produce a final value used for comparison between different values of k.

Setting the values of t(C) for the SCut method is through a Modified Leave-one-out Cross-validation Algorithm. For each document $\vec{d}$ in the training set, use every other document in the training set to assign scores $S(c,\vec{d})$ via the k-NN algorithm. Then set the values of t(C) to be those which produce optimal performance over this

set of scores. This method has the advantage of deriving the values of t(C) from a data set that is as close as possible to the actual training data.

At last, we use the standard cosine-similarity metric to compute similarity between the training and test documents.

$$\text{e.g. } sim(\vec{d},\vec{x}) = \cos(\vec{d},\vec{x}) = \frac{\vec{d}\cdot\vec{x}}{\|\vec{d}\|\cdot\|\vec{x}\|} \tag{11}$$

5.2 Evaluation

In experiment, each document belongs to no more than two categories. But the evaluation only consults the first category with highest score.

For text categorization evaluation, the effectiveness measures of precision, recall and F1 are defined respectively. For $category_j$:

$$precision_j = \frac{l_j}{m_j} \times 100\ \%$$

$$recall_j = \frac{l_j}{n_j} \times 100\ \% \tag{12}$$

$$F1_j = \frac{recall_j \times precision_j \times 2}{recall_j + precision_j}$$

Where l_j is the number of test set category members assigned to $category_j$ and m_j is the total number of test set members assigned to $category_j$ .Where n_j is the number of category members in test set.

Thus for all categories, Macro-recall, Macro-precision and Macro averaged F1 score are respectively defined.

In our experiment, different feature-set sizes were tested, and the size optimized by the global F1 score for classifier. Finally, 5362 features were selected for k-NN. And the k in k-NN was set to 50. Through the transformations of SDD, the 5362 feature terms become a lower dimensional space of rank-k approximation.

Affirmatively applying SDD in train phase upgrade a little training cost, but it is negligible to performance-promoting in test phase. The experiment shows that k-NN (SDD) promotes executive efficiency greatly with storage cost reduced from 96.6M to 5.62M and executive time condensed from 3437s to 435s.

The choice of k-value is an empirical method. The term-document matrix was fed into SDD transformation with a variety of k-values. The results of these levels are displayed in Figure 1. They showed the best average F1 with k=140.

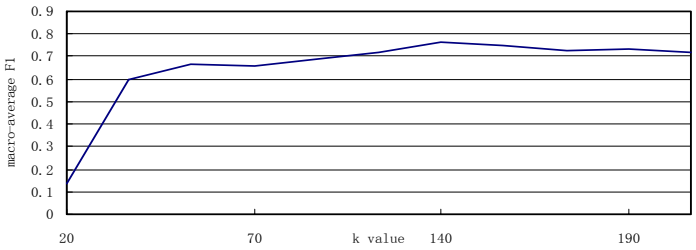

Fig. 1. Result of current LSI (SDD) system with various k-values

Table 2 shows the precision, recall and F1 score for each category on the full training set. Note that in all categories, k-NN LSI method made evaluation measure better in a different degree. Totally using k-NN LSI against k-NN VSM, 143 error documents are eliminated and F_1 score is 9.48 percent higher than before.

Table 2. Comparison of the results on P, R and F_1 between k-NN LSI and k-NN VSM

	n_j	k-NN LSI					K-NN VSM				
		m_j	l_j	$Precsion_j$	$Recall_j$	F_1	m_j	l_j	$Precsion_j$	$Recall_j$	F_1
A	49	49	41	83.67%	83.67%	0.83	50	37	74.00%	75.51%	0.7475
B	41	47	38	80.85%	92.68%	0.86	55	38	69.09%	92.68%	0.7917
C	48	16	14	87.50%	29.17%	0.43	3	2	66.67%	4.17%	0.0784
D	50	72	26	36.11%	52.00%	0.42	59	19	32.20%	38.00%	0.3486
E	50	39	34	87.18%	68.00%	0.76	33	29	87.88%	58.00%	0.6988
F	50	95	48	50.53%	96.00%	0.66	130	48	36.92%	96.00%	0.5333
G	94	101	90	89.11%	95.74%	0.92	100	85	85.00%	90.43%	0.8763
H	50	48	46	95.83%	92.00%	0.93	45	42	93.33%	84.00%	0.8842
I	50	59	49	83.05%	98.00%	0.89	59	48	81.36%	96.00%	0.8807
J	50	49	43	87.76%	86.00%	0.86	49	44	89.80%	88.00%	0.8889
K	50	17	16	94.12%	32.00%	0.47	19	16	84.21%	32.00%	0.4638
N	8	8	6	75.00%	75.00%	0.75	3	2	66.67%	25.00%	0.3636
O	24	19	19	100.00%	79.17%	0.88	14	13	92.86%	54.17%	0.6842
P	50	59	48	81.36%	96.00%	0.88	59	47	79.66%	94.00%	0.8624
Q	50	44	42	95.45%	84.00%	0.89	32	30	93.75%	60.00%	0.7317
R	50	52	48	92.31%	96.00%	0.94	45	38	84.44%	76.00%	0.8000
S	47	46	38	82.61%	80.85%	0.81	59	34	57.63%	72.34%	0.6415
T	50	28	14	50.00%	28.00%	0.35	9	2	22.22%	4.00%	0.0678
T	48	51	40	78.43%	83.33%	0.80	48	38	79.17%	79.17%	0.7917
T	45	46	43	93.48%	95.56%	0.94	47	42	89.36%	93.33%	0.9130
T	49	58	36	62.07%	73.47%	0.67	54	30	55.56%	61.22%	0.5825
T	48	50	24	48.00%	50.00%	0.48	78	28	35.90%	58.33%	0.4444
T	48	38	25	65.79%	52.08%	0.58	37	16	43.24%	33.33%	0.3765

TJ	50	58	44	75.86%	88.00%	0.81	63	45	71.43%	90.00%	0.7965
T	47	48	37	77.08%	78.72%	0.77	45	32	71.11%	68.09%	0.6957
T	47	48	42	87.50%	89.36%	0.88	51	41	80.39%	87.23%	0.8367
T	49	50	37	74.00%	75.51%	0.74	51	33	64.71%	67.35%	0.660
T	46	46	35	76.09%	76.09%	0.76	55	38	69.09%	82.61%	0.7525
T	50	83	47	56.63%	94.00%	0.70	90	48	53.33%	96.00%	0.6857
T	48	55	37	67.27%	77.08%	0.71	59	40	67.80%	83.33%	0.7477
T	48	32	28	87.50%	58.33%	0.70	8	8	100.00%	16.67%	0.2857
T	50	47	37	78.72%	74.00%	0.76	42	33	78.57%	66.00%	0.7174
T	50	62	44	70.97%	88.00%	0.78	65	44	67.69%	88.00%	0.7652
U	49	58	42	72.41%	85.71%	0.78	57	41	71.93%	83.67%	0.7736
V	50	32	25	78.13%	50.00%	0.60	26	16	61.54%	32.00%	0.4211
X	50	23	16	69.57%	32.00%	0.43	40	15	37.50%	30.00%	0.3333
Z	9	9	8	88.89%	88.89	0.88	3	2	66.67%	22.22%	0.33
M				77.32%	75.03%	0.76			69.26%	64.29%	0.6668

Figure 2 compare the performance curves of the improved and the original classifiers on F_1 score with respect to the 37 categories. These curves are obtained by regarding categories as the horizontal axis and plotting the per category F_1 scores for each classifier.

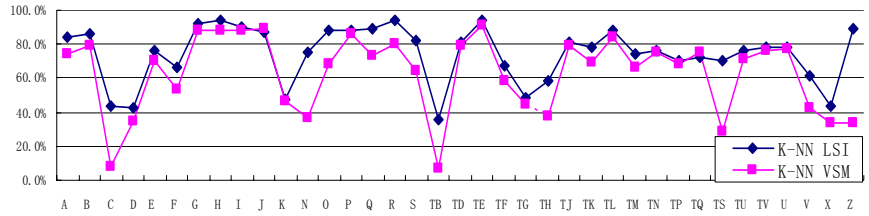

Fig. 2. The macro-averaged F_1 curves for each category using k-NN VSM versus k-NN LSI

Through experiments we find that the efficiency of automated text categorization is dependent on train set quality seriously, therefore the parameter value presented in this paper should be adjusted to corpus in others application.

6 Conclusion

In this paper we presented a study with significance analyses on text categorization methods based on LSI (SDD). Our main conclusions are:

- LSI (SDD) can achieve similar or higher performance at a much lower storage cost and little executive time in automated text categorization. As the volume of information available on the Internet continues to increase, online text categorization is required seriously. So LSI (SDD) is a promising technique for text categorization.
- LSI (SDD) is a technique that warrants further investigation for text classification.

7 Future Work

Much work remains. Firstly, we can improve the method to compute the distance between two document vectors[9].With cosine computing documents similarity, the technique neglects the different among features and all index terms are of equal importance in the process of computing, which lead to the inexact results and drop down the precision of text categorization. Secondly, lexical analysis can be further studied by treating some phrases as terms. Thirdly, the reasons for the poor performance of a number of categorization should be investigated.

Acknowledgements

We would like to thank Sun ChenJie and Mao YongQuan for gathering web pages and system robust tests, which made the significant test carried out smoothly.

References

1. Kjersti Aas and Line Eikvil. June (1999). Text Categorisation: A Survey. Norwegian Computing Center.
2. Ana Cardoso-Cachopo and Arlindo Limede Oliveira, (2000). An Empirical Comparison of Text Categorization Methods, Instituto Superior T_ecnico Departamento de Engenharia Inform_atica Av. Rovisco Pais.
3. Michael W. Berry and Ricardo D.Fierro . (1996). Low_Rank Orthogonal Decompositions for Information Retrieval Applications. Numerical Linear Algebra with Applications, Vol1(1),1-72.
4. Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science.
5. Kolda, T. G. and O'Leary, D. P. (2000). Algorithm 805: Computation and uses of the semidiscrete matrix decomposition, ACM Transactions on Mathematical Software 26(3): 415{435.
6. Yiming Yang and Xin Liu. August (1999). A re-examination of text categorization methods. Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 42-49, Berkeley, CA, USA.
7. S. Dumais, J. Platt, D. Heckerman, and M. Sahami . (1998). Inductive Learning Algorithms and Representations for Text Categorization, Technical Report, Microsoft Research.
8. Tom Ault and Yiming Yang. (1999) kNN at TREC-9. Language Technologies Institute and Computer Science Department Newell Simon Hall 3612C, Carnegie Mellon University Pittsburgh, PA 15213?213, USA
9. Caron, J. (2000). Experiments with lsa scoring: Optimal rank and basis, Technical report,SIAM Computatio-nal IR Workshop.URL: citeseer.nj.nec.com