Noise Reduction in a Statistical Approach to Text Categorization

Yiming Yang
Section of Medical Information Resources
Mayo Clinic/Foundation
Rochester, Minnesota 55905 USA

Abstract

This paper studies noise reduction for computational efficiency improvements in a statistical learning method for text categorization, the Linear Least Squares Fit (LLSF) mapping. Multiple noise reduction strategies are proposed and evaluated, including: an aggressive removal of "non-informative words" from texts before training; the use of a truncated singular value decomposition to cut off noisy "latent semantic structures" during training; the elimination of non-influential components in the LLSF solution (a word-concept association matrix) after training. Text collections in different domains were used for evaluation. Significant improvements in computational efficiency without losing categorization accuracy were evident in the testing results.

1 Introduction

The task of text categorization is to assign predefined categories to texts. It has wide application since a controlled vocabulary (subject categories) is often used to index real-world databases for retrieval purposes. While text categorization is highly related to text retrieval, and many techniques developed for the latter are applicable to the former, there is a particular problem which is more serious in categorization than in retrieval. That is, the vocabulary gap between free texts and the controlled indexing language of a particular database is usually large. Consequently, search methods based on shared words between free text and category names typically exhibit poor performance [1] [2] [3]. The importance of using human knowledge to solve the vocabulary gap problem has been recognized, and statistical learning of text-to-categories mapping based on human assignments has been a major focus in recent research in text categorization [3] [4] [5] [6] [7]. The LLSF mapping is a successful learner relying on past human relevance judgments, and can be used for both text retrieval [2] and text categorization. Significant improvements of LLSF mapping have been observed in previous evaluations, compared to alternatives such as word-based matching methods which do not use any human knowledge, and thesaurus-based methods which are heavily dependent on manually coded human knowledge [3].

LLSF uses a corpus of manually categorized texts for training. The training data are represented using two matrices, A and B, where A is a text-word matrix and B is a text-category matrix. An element of matrix A is the weight of a word in a corresponding text, and an element of matrix B is the weight of a category in a corresponding text. The LLSF problem is defined as to find a matrix X which minimizes the squared error of $AX - B$. More precisely, the problem is to find a matrix X which minimizes the Frobenius norm of the residual matrix $E = AX - B$ where E is $n \times l$, and the Frobenius matrix norm is

$$\|E\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{l} e_{ij}^2}$$

(see [3] for a more detailed presentation of LLSF). The intuitive interpretation is to find X which minimizes the squared error in the mapping from training texts to their categories. The solution X is a word-category association matrix whose elements are the weights of these associations. It is used to transform an arbitrary text represented using a word vector, $\vec{a}$, to a category vector, $\vec{b}$, by computing $\vec{a} \times X = \vec{b}$. The elements of vector $\vec{b}$ are interpreted as the relevance scores of categories with respect to the text. By sorting these scores, a ranked list of categories is obtained, which is the output of the LLSF mapping.

A practical and crucial question about LLSF is the computational complexity when applying LLSF to very large text collections. A conventional method for solving a LLSF problem employs a singular value decomposition (SVD) [8] of the input matrix A (the text-word matrix) as a part of the computation. The standard SVD algorithm, as implemented in LINPACK [9], has a time complexity $O(m^2 n)$, where m is the number of texts in the training corpus and n is the number of distinct words in these texts (assuming $m \geq n$). If m and n are large, this cubic complexity can become a computational bottleneck. Alternate algorithms were developed for very large and sparse matrices. The Lanczos methods [8] [10] [11], for example, are particularly efficient in situations where relatively few singular values are desired and relatively few orthogonalizations are needed. To take full advantage of Lanczos, however, the validity of using a truncated SVD instead of a complete SVD in LLSF needs to be established.

This study seeks methods to improve the computational efficiency of LLSF without sacrificing effectiveness, so larger training corpora can be used. The claim is that this is achievable because of a "noise" or "redundancy" property of natural language texts: not every word or word combination is equally informative. This means that many words and word combinations can be ignored in the training of LLSF, if they can be automatically identified. In other words, a mathematically complete LLSF model based on very noisy training data may not be the most desirable, and a "relaxation" of this model may significantly improve the computational efficiency and possibly improve the categorization accuracy. In the following sections, three methods of noise reduction are proposed and evaluated, including an aggressive removal of "non-informative" words from texts before training, the truncation of "noisy latent semantic structures" during training, and the elimination of non-influential elements in the LLSF solution matrix after training.

2 Measurements and Data

Before describing each of the noise reduction methods, let us overview the methodology, measurements and data collections which are shared in studies of these methods. For each method, different truncation thresholds are used to test the trade-off between improvements in categorization accuracy and computational efficiency. The categorization accuracy of LLSF is judged using a collection of testing documents whose categories were assigned by humans. The conventional recall and precision measures are used:

$$\text{recall} = \frac{\text{categories found and correct}}{\text{total categories correct}};$$

$$\text{precision} = \frac{\text{categories found and correct}}{\text{total categories found}}.$$

Given a document, for recall thresholds of 10%, 20%, ... 100%, the system assigns in decreasing score order as many categories as needed until a recall threshold is achieved, and computes the precision value at that point; the resulting 10 point precision values then are averaged for a global measure of the system performance with respect to the document. For a set of documents, the 10-point average precision values of individual documents are further averaged to obtain the global measure of the system over the entire set. We refer to the 10-point average precision as categorization accuracy. Average precision is the most commonly used measure in evaluations of categorization systems, because many systems provide a ranked list of categories for a given text instead of binary decisions over categories [3] [4] [5] [6] [7]. A ranked list, of course, can be used to obtain binary decisions by setting a threshold. There are discussions about using alternative measures [12] [3]; these issues, however, are open research questions, and are not the focus of this paper.

To measure efficiency improvement, the time savings in SVD is used because it is the major part of the training phase, and the time savings in the text-to-categories mapping is also used because it is the major part of the testing phase. The other computations are rather insignificant in LLSF, so they are omitted in the performance evaluation. The SVD times in the LINPACK and Lanczos algorithms are used for comparison. A SPARCstation 10 was used for the experiments.

Three different collections are used in the evaluation, including patient record texts from the Mayo Clinic, documents from the MEDLINE bibliographical database, and the CACM information retrieval test collection in the computer science field. These collections are named SURCL, MEDCL and CACMCL, and have already been used for evaluations of text categorization and text retrieval. For convenience, I will use document as a generic word for either a patient-record text, or a text derived from a record in MEDLINE or CACM, by concatenating the title, the abstract and keywords (if available) of an article.

(1) **SURCL** is a collection of patient-record texts from the Mayo Clinic archive. The patient records at Mayo include diagnoses and operative reports in natural language texts written by physicians. These texts are manually categorized by experts for the purpose of billing and research. About 1.5 million diagnoses and operative reports are manually coded each year. For this experiment, we arbitrarily chose the cardiovascular subdomain of the canonical classification system ICD-9-CM (International Classification of Diseases, 9th Revision, Clinical Modifications), and used the 6134 surgical procedure reports in this subdomain from the 1990 patient records. We sorted the procedure/category pairs by category and arbitrarily split these pairs into odd and even halves. The odd-half was used as the training set which contained 3067 texts with duplicates, or 1461 unique texts; the even-half was used as the testing set which contained 3067 texts with duplicates, or 1492 unique texts. About 58% of the testing texts had an identical text in the training set; about 99% of the words and 97% of the categories in the testing documents had occurrences in the training set. About 99.8% of the training and testing texts had a uniquely matched category; the rest had two or three categories. The average length of texts was about 9 words. There were totally 281 categories in the cardiovascular subdomain of ICD-9-CM; these 281 categories were used as the candidate space of the categorization tests. The chance of a correct categorization of a procedure text by a random assignment was 1 in 281, or 0.36%. The categories which are not included in the training set will be automatically assigned to a relevance score of zero during the LLSF mapping, and their ranks among zero-valued categories are determined arbitrarily.

(2) **MEDCL** is a collection of MEDLINE documents. This data set was originally designed for an evaluation of the Boolean search of MEDLINE retrieval (Haynes et al. 1990), and has been used for evaluations of other retrieval and categorization systems (Hersh et al. 1992) (Yang & Chute, 1993 July, 1993 November, 1994). We take the words in the title and abstract of each record without distinction, and call these words together a document; we did not use the records where the abstracts were missing. The resulting collection contains 2344 documents, and each document has categories assigned by MEDLINE indexers. We arbitrarily chose a quarter (586 documents) of these documents for training, and the remaining ones (1758 documents) for testing. There were no duplicate documents in the entire collection, and consequently, none of the testing documents was identical to a training document. There were about 168 words and 17 categories per document on average. The training set contained 7813 unique words and 1832 unique categories. The testing set contained 14,339 unique words and 3430 unique categories; about 42% of the words and 46% of the categories in the testing documents had occurrences in the training set. There were 4020 unique categories in total, including the training set and the testing set; these 4020 categories were used as the candidate space of the categorization tests. The chance of a random assignment being correct was 17 in 4020, or 0.42%.

(3) **CACMCL** is a subset of documents derived from the CACM collection which is one of the standard information retrieval test collections [13]. The CACM collection consists of 3703 records each of which contains fields of "title", "abstract", "keys" (keywords), "categories", "author", etc. We take the words in the fields of "title", "abstract" and "keys" without distinction, and call these words together a document. The categories are defined in the Classification System for Computing Reviews (CSCR) and were assigned by humans to documents [14]. We used a subset of the 3703 documents in our experiments. We eliminated documents with an empty category field as obviously unsuitable for the study. We also eliminated documents with empty abstract fields because our primary interest is in documents with the potential for significant word removal. Another consideration in document selection was the type of category codes. Two versions of CSCR category codes had been used in CACM: the original version and an updated replacement version. To avoid potential inconsistencies in the data, we only chose the documents from the larger set that were classified by the original codes. The resulting collection consists of 1121 documents; we refer to it as CACMCL. We further arbitrarily split CACMCL into two halves, and used the first half (561 documents) for training, and the second half (560 documents) for testing. None of the testing documents was identical to a training document; about 60% of the words and

87% of the categories in the testing documents had occurrences in the training set. There were about 120 words and 3 categories per document on average in the CACMCL collection. There was a total of 204 categories in CSCR, and we used the entire set as the candidate space of the categorization tests. The chance of a random assignment being correct was 3 in 204, or 1.5%.

3 Word Removal

Removal of non-informative words is a commonly used technique in document indexing and retrieval to improve the accuracy of the results and to reduce the redundancy of the computation. Non-informative words are often defined by a "stop-word list" which typically consists of about 300 or 400 words, including articles, prepositions, conjunctions and some high-frequency words. Most systems apply the same generic stop-word list to all document collections without change. While the generic stop words are relatively "safe" to remove in the sense that their removal rarely causes a significant accuracy loss, the chance of significant accuracy improvement is also small [3] [15]. Since a generic stop-word list is often much smaller than the vocabularies of real-world document collections, only a limited number of words can be removed from texts, and the improvement in computational efficiency is therefore very limited. Experiments in adding collection-specific high-frequency words to a generic stop-word list have shown no reliable improvements [15].

In contrast to using generic stop words, Wilbur and Sirotkin developed a novel stop-word identification method which allows a far more aggressive removal of words from documents without losing retrieval accuracy [16]. This method uses a collection of training documents to estimate word importance using a score, namely "word strength". Clearly, word strengths are domain specific or application specific. An important difference between word strength and other corpus-dependent word weighting schemes such as the Inverse Document Frequency [17] is that word strength is not computed based on word occurrencies in documents, but based on word co-occurrencies in pairs of related documents. Word strength measures how informative a word is in identifying two related documents. The strength of a word, t, is defined as the probability of finding t in a document which is related to any document in which t occurs,

$$s(t) \stackrel{\text{def}}{=} P_r(t \text{ is in document } y |\ t \text{ is in document } x)$$

where x and y denote an arbitrary pair of distinct but related documents. For computing word strength, one needs a training corpus where relevance judgments between documents are available. It would be ideal to use human judgments as the gold standard. Such relevance judgments, however, are often not available in real-world applications. Wilbur and Sirotkin have shown that one can relax the relevance criterion by assuming two documents are related to each other if they have many words in common. That is, one can use the conventional cosine-coefficiency of two vectorized documents to measure their similarity, and identify a pair of documents as related if their cosine-similarity value is above a threshold. Using the pairs of related documents, one can approximately compute word strength as

$$s(t) \approx \frac{\text{\# of pairs in which } t \text{ co-occurs in both documents}}{\text{\# of pairs in which } t \text{ occurs in the first document}}$$

where the "first document" can be any training document. That is, there are no constraints on the first document of a pair; if (x, y) is a pair of related documents, then (y, x) is also a pair of related documents.

The procedure of word strength computation consists of the following steps:

(1) use a standard stop-word list to eliminate non-informative words from training documents;

(2) compute the similarity values of all pairs of training documents;

(3) select the document pairs whose similarity values are above a threshold (chosen experimentally) as related document pairs;

(4) compute the strength for each word using the related document pairs.

To identify stop words in the Wilbur-Sirotkin method, a word strength is compared with the expected strength of a word which is distributed randomly in training documents with the same frequency. If the word strength is not at least two standard deviations above the strength of such a randomly distributed word, it is designated a stop word (refer to [16] for details). This method was tested using a training set of 71,311 documents from the MEDLINE database in the area of molecular biology/genetics. According to the published results, applying the aggressive stop-word list reduced the 203,040 unique words in the 71,311 documents to 50,508 words (a 75% word removal), and the retrieval results on these 71,311 documents were more favorable (by human judgments) than applying a generic stop-word list of 310 words. In this particular test, queries were randomly selected documents from the database and the task was to find closely related documents in the database. As documents, the queries were longer than most queries a searcher might produce and this may explain why such a high proportion of words could be removed with improved retrieval. In general the proportion of removable words may be less but still vary significant.

Yang and Wilbur have applied the aggressive word removal method to document categorization to remove non-informative words from documents before applying a categorization method to these documents [18]. The effects on several categorization methods on different document collections have been studied and the effectiveness has been evident in the experiments. For all the methods tested, including two statistical learning methods based on manual category assignments and a baseline text matching method based on shared words in documents and category names, no accuracy loss or only insignificant accuracy loss was observed with an aggressive word removal, compared to using a standard stop-word list. In the test of a statistical learning method, Expert Network, for example, a 85% word removal on the CACMCL collection led to a better accuracy (a 2% improvement) than using a standard stop-word list which yielded only 7% word removal. The 85% word removal had a time savings significantly larger than the 7% word removal because the computation in ExpNet was proportional to the number of word occurrences in documents. As another example, the baseline text matching (the SMART system without relevance feedback was used for this experiment) on the CACMCL collection with 71% word removal had the same accuracy as using the standard stop-word list, while the time savings were significant because the match computation is proportional to the number of term occurrences in documents. The focus here is on how LLSF responds to aggressive word removal, particularly, how much time savings can be obtained in the SVD which is the major bottleneck in the LLSF computation.

Figure 1 shows the precision curve of LLSF in response to word cut ratios. The data points at the lowest word cut ratio correspond to applying a standard stop-word list. The data points at the higher word cut ratios correspond to the removal of words with a strength value less than or equal to 0, 0.1, 0.2, ..., 0.9. The word strengths are collection-dependent, i.e. they were computed for the SURCL, MEDCL and CACMCL respectively, using the corresponding document collection as the training corpus. Figure 1 shows a relatively flat precision curve with up to 77% word cuts on SURCL and MEDCL, and up to 54% word cuts on CACMCL, indicating that

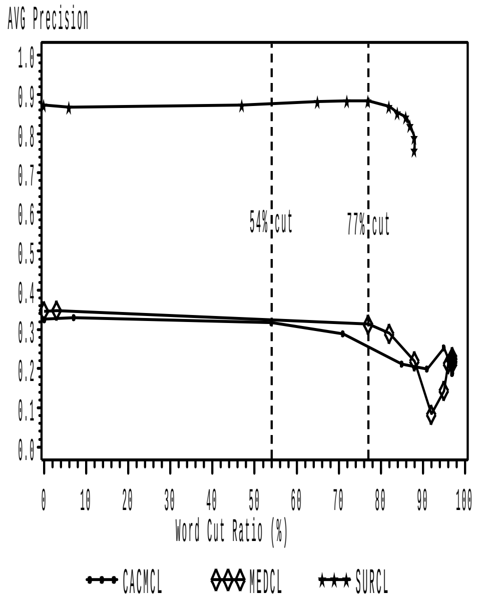

Figure 1: Word cuts and precision curves.

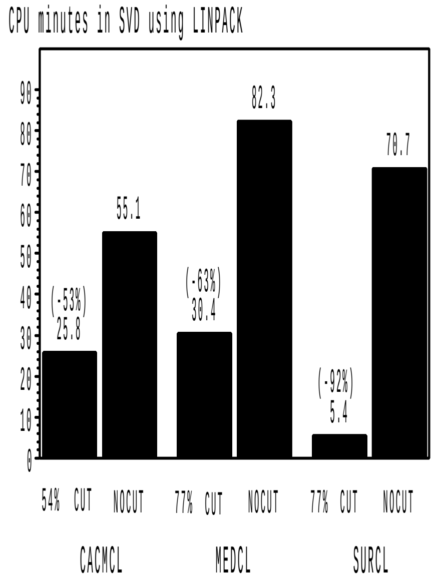

Figure 2: Word cuts and time savings in LINPACK.

the LLSF mapping is only sensitive to a relatively small portion of words in the documents, and that word strengths are useful for identifying these words. While these aggressive word cuts did not lead to impressive changes in precision, their impact on computation time was significant. Figure 2 shows the SVD time with a 77% word cut on SURCL and MEDCL and with a 54% word cut on CACMCL, in comparison with not using word removal. With 77% word cuts on SURCL, for example, the SVD time is reduced from 71 CPU minutes to 5 minutes, that is, a 91% time savings. The time savings on MEDCL and CACMCL were 63% and 53%, respectively. The SVD time was measured using the CPU minutes of the LINPACK algorithm.

4 Singular Values Truncation

The idea of using truncated SVD to reduce the noise level in training documents is inspired by the Latent Structure Analysis theory, a well-known statistical method for analyzing causal factors or hidden reasons behind observed events [19]. A latent structure is defined as a linear combination of the independent variables, and can be computed using the SVD of a matrix which represents correspondences between independent and dependent variables. The SVD produces a set of singular values (SV) which are non-negative real numbers, and two sets of singular vectors each of which is an orthogonal basis of the vector space of the given matrix. One set of the singular vectors has independent variables as the dimensions, and the other set of singular vectors has dependent variables as the dimensions. These singular vectors are interpreted as the "orthogonal factors", "artificial concepts" or "latent semantic structures" (LSS) [20]. The word "latent" is used because it is often difficult to give an intuitive interpretation about the meanings of these structures. It would be helpful, however, to have some understanding of the LSS. In general, the orthogonal basis of a given matrix consists of linear combinations of the row or column vectors of the matrix. In our problem, the rows are words, the columns are documents, and an LSS is a linear combination of documents [20]. Since each document is a vector of word weights, a linear combination of documents is again a vector of word weights. Therefore, we can say that an LSS is "a word combination" which does not mean a phrase, a sentence, or any continuous piece of text. It is a combination of weighted words from different documents. and the weights of these words are determined by the SVD algorithm. An important property of the LSS's is that they are orthogonal dimensions for representing or distinguishing documents. There is a one-to-one correspondence between a singular value and a latent structure, and the magnitude of a singular value reflects how influential the corresponding latent structure is in representing the information in the given matrix. One can select the most influential factors or the principle structures by truncating the singular values and their corresponding singular vectors using a threshold.

While LLSF uses SVD as a part of its computation, truncated SVD is not a formal part of a complete LLSF model. Mathematically, LLSF guarantees a least squared error for a given training set, and any modification of the computation, including the truncation of singular values, can only increase the error. From a real-world application point of view, a complete LLSF computation may not be the best choice given the natural language vagaries in training documents. Assuming that not all the word combinations are meaningful or equally informative from a categorization point of view, it is interesting to see whether a truncated SVD can identify meaningless word combinations and reduce their effects in categorization. Note that when using truncated SVD instead of complete SVD, the solution is no longer a LLSF but a only a "pseudo-LLSF". For convenience, I do not distinguish between a complete LLSF and a pseudo-LLSF, unless specified.

I want to point out the similarities of the pseudo-LLSF and the Latent Semantic Indexing (LSI) in document retrieval [20], and the fundamental difference between my hypothesis about truncated SVD and the claim in LSI. These two approaches are similar in the sense that both compute the SVD of a word-document matrix, and both intend to use an aggressive truncation of singular values ("SV cut") to obtain the most important dimensions or "the Latent Semantic Structures" in the vector space of documents. The difference is that LSI uses the selected structures to reconstruct document vectors, and hypothesizes that the truncation of non-influential structures would result in improved representations of synonyms in

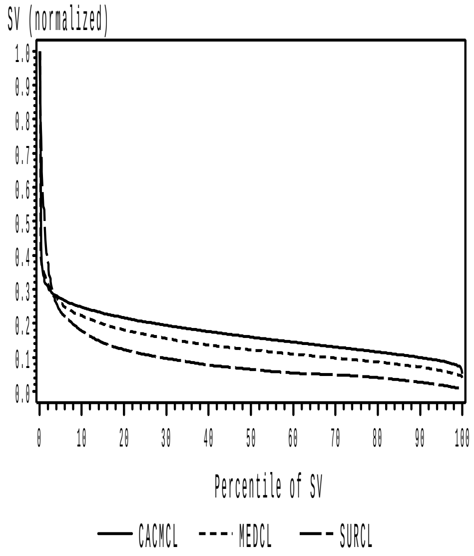

Figure 3: SV curves of different training sets.

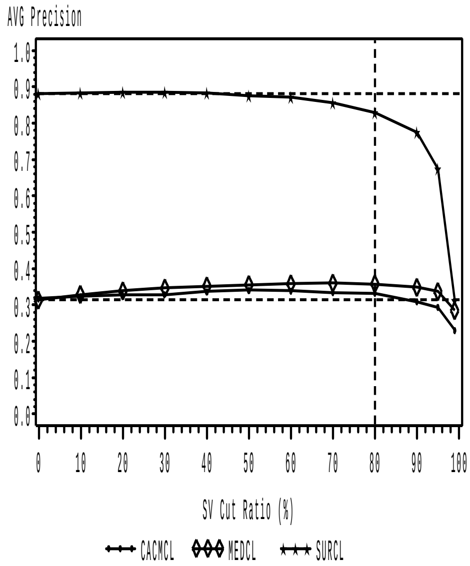

Figure 4: SV cuts and precision curves.

the reconstructed document vectors. Consequently, the retrieval of documents which contain the synonyms of the words in a given query should be improved. This is the basic claim in LSI about how to use truncated SVD and why it would work. It is important for LSI to have significant improvement in retrieval effectiveness over baseline text matching, since there would be no reason to use LSI otherwise, given that the additional cost of the SVD computation is far more expensive than baseline text matching. Evaluation results, however, have shown no reliable improvement of LSI over baseline text matching [20] [21] [22].

My hypothesis about truncated SVD is different from the hypothesis in LSI. I do not claim an improvement of synonym representation in a document matrix by using such an approach. Whether such a hypothesis is true or false is not crucial for LLSF because the document-to-categories mapping is based on human relevance judgments, not based on how synonyms are represented in documents. My hypothesis about truncated SVD is the effective removal of "noisy" and redundant semantic structures or word combinations in documents without lossing categorization accuracy. By "noisy", I mean some word combinations are not meaningful from a categorization point of view. This kind of noise could cause overfitting and wasteful computation, and its removal could improve the categorization accuracy, and significantly reduce the computation if a large number of word combinations can be truncated. This hypothesis is verifiable by observing the effects of SV cutting through experiments.

Figure 3 shows the SV curves of the training documents in the SURCL, MEDCL and CACMCL collections, where the SVs are normalized using the maximum SV of the corresponding collection; only the non-zero SVs are plotted. All these curves have a sharp decline among the top-ranking SVs and a relatively flat curve at the region of lower ranking SVs, meaning that only a small number of word combinations are important or informative for distinguishing the differences between training documents, and that the SVs at the flat tails of these curves are relatively "safe" to be truncated. Figure 4 confirms such an assertion, which shows the accuracy curve of each collection in response to SV cuts at thresholds of 0%, 10%, 20%, ..., 90%, 95% and 99% of the number of non-zero SV values in that collection. On the MEDCL and CACMCL collections, there is an improvement in accuracy with up to 80% SV cuts, compared to not using SV cutting. On the SURCL collection, the accuracy curve remains flat with up to 40% SV cuts, and starts to decline with more aggressive SV cuts. The relatively different response in accuracy of LLSF on SURCL is probably due to the small vocabulary of the SURCL training documents. That is, we applied a 77% word cut on SURCL and MEDCL, and a 54% word cut on CACMCL before the SVD tests. The number of unique words in training documents after word removal was only 265 in SURCL, while there were 3072 unique words in MEDCL and 2657 unique words in CACMCL. The larger vocabularies of MEDCL and CACMCL possibly offer a higher level of noise which can be reduced by SV cutting. Nevertheless, all these curves show a wide range of SV cutting without penalty or with an improvement in accuracy, supporting the validity of using truncated SVD in LLSF for a modification of the original model.

As for the effects of SV cutting on computational efficiency, the degree of improvement is dependent on which SVD algorithm to use, and how many SVs to cut off. For the LINPACK algorithm, which is the most commonly used algorithm to compute the SVD of dense matrices, an efficiency improvement is not possible because this algorithm computes the full set of SVs simultaneously. On the other hand, for the Lanczos algorithms, any subset of SV's can be computed separately, and substantial efficiency improvement is possible if the truncation is sufficiently aggressive. Figure 5 shows the observed time savings in the SVD computation on the three document collections using a Lanczos algorithm, namely the Sparse Singular Value Decomposition or SSVD algorithm [11]. While no time savings were observed with up to 50% SV cuts, around 70% time savings were obtained at the 80% SV truncation threshold.

Table 1 summarizes the effects of combined use of word cutting and SV cutting in solving the LLSF on SURCL, MEDCL and CACMCL. For each of these document collections, precisions and SVD times are compared under the conditions of

(1) no word cutting and no SV cutting;

(2) with a selected word cut but no SV cutting;

Table 1. Results summary of word cutting and SV cutting in LLSF

DATA SET	Word Cut /SV Cut	Average Precision	LINPACK CPU min	SSVD CPU sec
	0%/0%	0.8719	71	Infinite
SURCL	77%/0%	0.8813 (+1.1%)	5.4 (-92%)	38.6
	77%/20%	0.8853 (+1.5%)	5.4 (-92%)	38.3
	0%/0%	0.3466	82	369
MEDCL	77%/0%	0.3139 (-9.4%)	30 (-63%)	292 (-20%)
	77%/80%	0.3575 (+3.1%)	30 (-63%)	93 (-74%)
	0%/0%	0.3265	55	300
CACMCL	54%/0%	0.3180 (-2.6%)	26 (-53%)	258 (-14%)
	54%/80%	0.3321 (+1.7%)	26 (-53%)	74 (-75%)

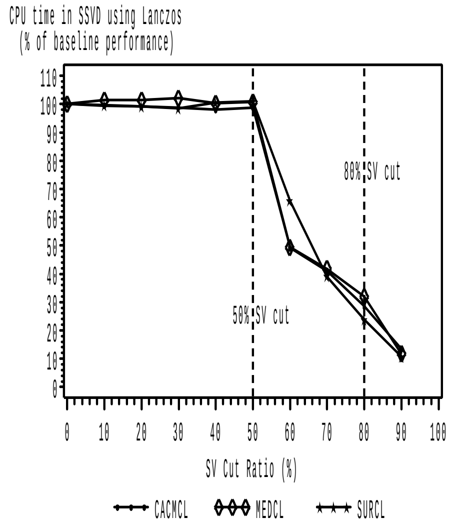

Figure 5: SV cuts and SVD time in Lanczos.

(3) with a selected word cut and a selected SV cut.

The results when not using word cutting and SV cutting were used as the baseline of the comparison on each collection. The word cutting and SV cutting amounts were chosen empirically, with an attempt to "optimize" the trade-off between improvements in accuracy and training time. Since particular applications of LLSF are not the focus of this paper, we do not have particular weights on the two sides of the trade-off. We selected suitable cutting ratios for a generic observation. On the MEDCL collection, for example, with a 77% word cut and a 80% SV cut, the precision improved by 3.1%; the time savings in SVD was 74% when using the SSVD algorithm, and 63% when using the LINPACK algorithm. On the CACMCL collection, as another example, with a 54% word cut and a 80% SV cut, the precision improved by 1.7%; the time savings in SVD was 75% when using the SSVD algorithm, and 53% when using the LINPACK algorithm. On the SURCL collection, there was a convergence problem with the SSVD algorithm when not using word cutting (indicated by "Infinite" in Table 1); nevertheless, the very fast computation with SSVD and the significant time savings with LINPACK were evident, when using an 77% word cut and a 20% SV cut; the accuracy also improved by 1.5%.

To summarize this section, singular value analysis can be used to distinguish meaningful word combinations in training documents from relatively meaningless ones, to reduce the noise at a semantic structure level, and to avoid unnecessary computation in the learning phase of LLSF. The observed accuracy improvements supports the validity of using truncated SVD instead of complete SVD, or using the pseudo-LLSF instead of the complete LLSF model. The significant improvement in computational efficiency shows the practical advantage of this approach. The cross-collection tests give evidence about the generality of this approach, and demonstrate a way to experimentally determine the suitable range of SV cuts for a specific domain, application or data collection. Finally, SV cutting can be used in combination with aggressive word cutting, to achieve a global improvement in both categorization accuracy and computational efficiency.

5 Element Cut in Solution Matrix

An additional research question is whether further noise reduction is possible during the run-time (testing) phase of LLSF. Although training efficiency is most crucial for scaling up LLSF, it is also possible to perform noise reduction and efficiency enhancement beyond the two training-time improvements just discussed. Recall that the solution of an LLSF problem is a word-category association matrix whose role is mapping a document to a vector of weighted categories. The mapping requires the computation $\vec{b} = \vec{a} \times X$, where $\vec{a}$ is the vectorized document, X is the LLSF solution, and $\vec{b}$ is the vector of weighted categories. Matrix X is generally dense. There are 91% non-zero elements in the solution matrix of SURCL, 93% in the solution matrix of MEDCL, and 96% in the solution matrix of CACMCL. This means that a word has a non-zero association to more than 90% of the categories in the training set, which is intuitively not sensible from a semantic point of view. It is possible that the meaning of a word matches to a few categories; however, it is unlikely that a word would have meanings wildly spread over a few hundreds or thousands of categories. This means that the majority of the low-valued non-zero elements in a LLSF solution may not be meaningful, and may not be necessary in the computation of the document-to-categories mapping. If this assertion is true, than one can use a much less dense matrix instead of the full X to achieve the same goal, and significantly reduce the mapping time because the computation in $\vec{a} \times X$ is proportional to the number of non-zero elements in matrix X and vector $\vec{a}$

Figure 6 shows the testing results of element cutting from the X-matrices of the SURCL, MEDCL and CACMCL collections. The element cutting thresholds were set to 0.1% 0.5%, 1%, ..., 50% of the largest absolute value in X; for each threshold, the non-zero elements were replaced by zero if their absolute values are smaller than that threshold. The 10-point average precisions were computed at each threshold and then interpolated. On all of these three data sets, there is almost no accuracy loss with up to 90-95% elements zeroed out. Clearly, the accuracy of categorization in LLSF is only sensitive to a small portion of the elements in matrix

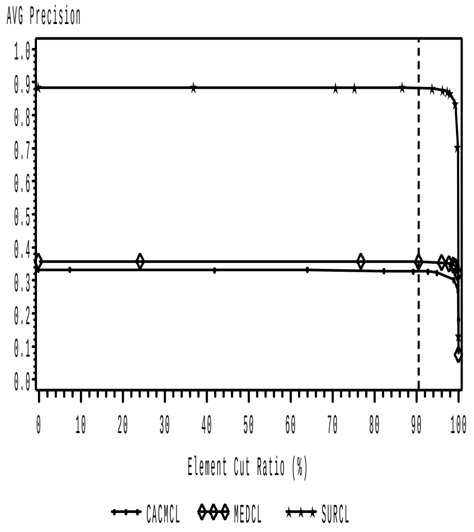

Figure 6: Effects of element cutting in LLSF solutions.

X, and the majority of elements can be eliminated for an efficiency improvement without accuracy loss. In the experiment on the 1758 MEDCL testing documents, for example, a 90.5% element cut from matrix X had the same precision as using the full matrix, and a 81% time savings in the document-to-categories mapping. The actual mapping time was reduced from 2.2 CPU seconds per document to 0.40 seconds. The time savings would be crucial in real-world applications when the category space is much larger and a real-time response of category ranking is required. The full set of categories (17419) used in the MEDLINE database are about 10 times in size of the set of 1832 categories in the MEDCL training documents. The time of document-to-categories mapping will be proportionally increased to roughly 21 seconds per document if not using element cutting in X. This is rather unacceptable as a real-time response (e.g. if we use the LLSF mapping to assist human indexing in an interactive environment). Using a 90.5% element cut in X, on the other hand, will reduce the response time to 4 seconds.

6 Conclusions

To conclude the study in this paper, noise and redundancy reduction is proposed and evaluated in the LLSF approach to document-to-categories mapping, at the levels of words, word combinations, and word-category associations. It is evident that natural language texts are highly noisy and redundant as training data for statistical classification, and that applying a complete mathematical model to such noisy and redundant data often results in over-fitting and wasteful computation in LLSF. It is also evident that the noise and redundancy can be automatically detected and reduced by aggressive word cutting based on the Wilbur word strengths, by truncating noisy semantic structures based on the singular value analysis of training documents, and by eliminating non-influential word-category associations from the LLSF solution matrix. The effective reduction techniques lead to a far more efficient computation in the LLSF mapping without accuracy penalty, a significant impact to the tractability of this method on large databases.

7 Acknowledgement

I would like to thank Drs. Terry Therneau and Chris Chute for the fruitful discussions about Latent Semantic Analysis, Dr. John Wilbur for the collaboration in computing word strengths, Geoffrey Atkin for programming assistance, and Dr. Andrew Anda for making the SSVD codes available for the experiments. This work is supported in part by NIH Research Grant LM-05416 to Mayo Clinic, and National Library of Medicine Training Grant LM-07041 in Medical Informatics to the University of Minnesota.

References

[1] Yang Y, Chute CG. (1992) A Linear Least Squares Fit mapping method for information retrieval from natural language texts. *Proc 14th International Conference on Computational Linguistics (COLING 92)*, 447-453.

[2] Yang Y, Chute CG. (1993, July) An application of Least Squares Fit Mapping to text information retrieval. *Proc 16th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 93)*, 281-290.

[3] Yang Y, Chute CG. (1994) An example-based mapping method for text categorization and retrieval. *ACM Transaction on Information Systems (TOIS 94)*: 253-277.

[4] Fuhr N, Hartmann S, Lustig G, et al. (1991) AIR/X - a rule-based multistage indexing systems for large subject fields. *Proceedings of the RIAO'91*, 606-623.

[5] Tzeras K, Hartmann S. (1993) Automatic indexing based on Bayesian inference networks. *Proc 16th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 93)*, 22-34.

[6] Masand B., Linoff G., Waltz D. (1992) Classifying News Stories using Memory Based Reasoning. *15th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 92)*: 59-64.

[7] Yang Y. (1994) Expert Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval. *17th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 94)*: 11-21.

[8] Golub GH, Van Loan CE. (1989) *Matrix Computations, 2nd Edition*. Baltimore, MD: The Johns Hopkins University Press.

[9] Dongarra JJ, Moler CB, Bunch JR, Stewart GW. (1979) *LINPACK Users' Guide.* , Philadelphia, PA: SIAM.

[10] Cullum JK and Willoughby RA. (1985) *Lanczos Algorithm for Large Symmetric Eigenvalue Computations. Vol.1: Theory.* Boston: Birkhauser.

[11] Berry MW. (1992) Large-Scale Sparse Singular Value Computations. *The International Journal of Super-computer Applications* Vol. 6 No.1: 13-49.

[12] Lewis DD. (1991) Evaluating Text Categorization *Proceedings of the Speech and Natural Language Workshop* Asilomar, Morgan Kaufmann:312-318.

[13] Fox EA. (Ed.). (1990) *Virginia Disc One.* Virginia Polytechnic Institute and State University, Nimbus Records.

[14] *ACM Guide to Computing Literature* Baltimore, MD: Association for Computing Machinery, 1984: 657-658.

[15] Buckley C, Salton G, Allan J. (1993) Automatic Retrieval With Locality Information Using SMART. In: DK Harman, Ed. *The First Text REtrieval Conference (TREC-1)*:59-65.

[16] Wilbur JW, Sirotkin K. (1992) The automatic identification of stop words. *J. Inf. Sci.* 18:45–55.

[17] Salton G. (1989) *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Reading, Pennsylvania: Addison-Wesley.

[18] Yang Y, W.J. Wilbur. (1995) Using Corpus Statistics to Remove Redundant Words in Text Categorization, *J Amer Soc Inf Sci* (accepted).

[19] Jackson JE. (1978) Review of "Methods for statistical data analysis of multivariate observations" *Technometrics* Vol 20: 210-211.

[20] Deerwester S., Dumais ST, Furnas GW, Landauer TK, Harshman R. (1990) Indexing by Latent Semantic analysis. *J Amer Soc Inf Sci* 41, 6, 391-407.

[21] Chute CG, Yang Y. (1991) Latent semantic indexing of medical diagnoses using UMLS Semantic Structures. *Proceedings of the Fifteenth Annual Symposium on Computer Applications in Medical Care (SCAMC 91)*:185-189.

[22] Dumais S. (1994) Latent Semantic Indexing (LSI) and TREC-2. In: DK Harman, Ed. *The Second Text REtrieval Conference (TREC-2)*:105–116.