Linear and Non-Linear Dimensional Reduction via Class Representatives for Text Classification

Dimitrios Zeimpekis
Computer Engineering & Informatics Dept.
University of Patras, 26500, Greece
dsz@hpclab.ceid.upatras.gr

Efstratios Gallopoulos
Computer Engineering & Informatics Dept.
University of Patras, 26500, Greece
stratis@ceid.upatras.gr

Abstract

We address the problem of building fast and effective text classification tools. We describe a "representatives methodology" related to feature extraction and illustrate its performance using as vehicles a centroid based method and a method based on clustered LSI that were recently proposed as useful tools for low rank matrix approximation and cost effective alternatives to LSI. The methodology is very flexible, providing the means for accelerating existing algorithms. It is also combined with kernel techniques to enable the analysis of data for which linear techniques are insufficient. Numerous classification examples indicate that the proposed technique is effective and efficient with an overall performance superior than existing linear and nonlinear LSI-based approaches.

1 Introduction and motivation

An important component of Information Retrieval (IR) concerns effective "text classification (TC)" or "automated text categorization" [19][1]. In TC, an initial dataset of preclassified documents is partitioned into a training dataset and a test dataset that are subsequently used to construct and evaluate classifiers. For high dimensional TC problems, it becomes critical to apply effective dimensionality reduction techniques (DR), including feature selection and extraction. LSI has been emerging as a powerful DR tool for TC [10, 11, 18, 14, 21, 29], specifically either as global or local preprocessing technique for feature extraction and in combination with any VSM-based classifier. In the "global LSI" approach, compression is applied on the entire training dataset, whereas, in "local LSI" compression is applied separately on subsets of documents belonging to each class so as to take into account classification information. To this end, state-of-the-art local LSI methods augment the local region of each class by considering negative examples and weighting appropriately (cf. [14]). The computational kernel[2] in LSI is singular value decomposition (SVD) whose straightforward application (even for sparse datasets) becomes very costly as the dimensionality of the problem increases, thus motivating the quest for more effective approximations.

To this end, inspired by initial ideas in [6, 16] and our work in [27] we investigate and show the effectiveness, for TC, of a general DR framework, we call "representatives methodology". The framework operates as follows: First the dataset is partitioned (e.g. by means of clustering), then representatives are selected from each partition and collectively used to capture the entire collection. Early work showed that centroid representatives (hereafter denoted by CM) are quite effective [6, 16], and often superior to the SVD, producing better reduced dimensional representations of text for retrieval. It was also shown recently that a few leading left singular vectors from each partition offer an attractive and more flexible, alternative set of representatives [27]; see also [3, 8]. Both approaches were compared for low rank matrix approximation and text retrieval applications in [27]. Both approaches can be cast in a convenient "rank reduction" framework; cf. [27] for more details. Worthy of note is the affinity of the methodology with sampling techniques [1, 7] (the sample being the representatives) and the potential for handling distributed data or for using distributed computing resources in the context of LSI or for using massive datasets in conveniently sized blocks [13]. In fact, the latter property was the justification used by some authors to present such techniques in the first place [12, 17].

In this paper we show that the representatives methodology, especially based on singular vectors, naturally lends

[1] Ensuing discussion assumes the Vector Space Model (VSM): Documents are characterized by a set of features/terms and are encoded as m-dimensional vectors so that a collection of n documents is encoded as an $m \times n$ term-document matrix, "tdm" for short.

[2] Not to be confused with "kernel" in "kernel based methods" discussed below.

itself to TC and outperforms other LSI-based methods. The algorithmic vehicle for our discussion will be CLSI, standing for "clustered LSI" and introduced by the authors in [27] as a method for low rank matrix approximation and text mining. The representatives methodology for TC does not require the application of LSI on the entire dataset, as global LSI does, in fact it takes advantage of the classification of the training set. Moreover, it uses the same projector for all test documents and so outperforms the local LSI approaches. We also show that the representatives methodology can be deployed to improve the performance of LLSF, an interesting TC technique originally proposed in [24]; cf. [20, 22, 26].

Our second contribution is the combination of CLSI with kernel-based techniques ([5, 4]) so as to allow it to be used to resolve nonlinearities in spaces of higher dimension; this complements work in [15] and can be the basis for a unified treatment of the kernelized representative methodology, that can lower the high computational and memory demands of Latent Semantic Kernels [5]. We show through experiments[3] the effectiveness and efficiency of the representatives approach and compare it with other classification techniques. Note that even though we only present results with k-nearest neighbor (abbreviated as kNN) and Rocchio classifiers, our techniques can be combined with any other VSM-based classifier, e.g. support vector machines [4].

2 Representatives methodology for TC

Table 1 shows the typical steps of the proposed methodology as applied in the context of matrix approximation. The target rank is denoted by k; the above steps produce an orthogonal basis, Q, for the subspace spanned by the representatives, as well as the compressed representation, Y, for the documents in the input tdm. The k-dimensional subspace spanned by Q will be used as an approximation of $\texttt{range}(A)$.

In step I, the algorithm performs a partitioning of the training tdm into l groups A_i. This partitioning phase could be the result of clustering (e.g. using Skmeans as in [6]) or some other method. In TC this step is void, since the columns of the training tdm are already labeled, and partitions correspond to classes. Step II sets the number of representatives, at least one, for each partition. We can enforce the use of a single representative for each partition ($k = l$), or utilize more than one representatives for one or more of the partitions ($k > l$). We fall under the temptation to draw an analogy with the U.S. Congress, thus referring to the former case as "Senate" and the latter as "House" representation. In the latter case the algorithm also needs a strategy for selecting the exact number, k_i, of representatives to select

[3]These represent only a small sample of extensive work to be reported in the full version of this paper.

Table 1. The representatives methodology.

Input: $m \times n$ training tdm A, number of classes l, integer k such that $l \geq l$ (rank of approximation)
Output: Orthogonal basis Q for the subspace of representatives approximating $\texttt{range}(A)$, $Y = Q^\top A \in \mathbb{R}^{k \times n}$

I. Partition the tdm $A = [A_1,\ A_2,\ \ldots,\ A_l]$ so that A_i corresponds to class i;
II. Set the number of representatives (cf. Section 2):
  if $(k = l)$ then set all $k_i = 1$
    else set $\{k_i\}_{i=1}^{l}$ so that $\sum_i k_i = k$
III. Select k_i representatives $[z_1^{(i)}, \ldots, \ \ldots, z_{k_i}^{(i)}]$ from each class and form:
  $\mathcal{U} = [z_1^{(1)}, ..., z_{k_1}^{(1)}, z_1^{(2)}, ..., z_{k_2}^{(2)}, ..., z_1^{(l)}, ..., z_{k_l}^{(l)}]$
IV. Compute orthogonal basis $Q_\mathcal{U}$ for $\texttt{range}(\mathcal{U})$, e.g. using QR: $\mathcal{U} = Q_\mathcal{U} R_\mathcal{U}$
V. Set $Q = Q_\mathcal{U}$and $Y = Q_\mathcal{U}^\top A$;

for each partition under the constraint that $\sum_{i=1}^{l} k_i = k$. To this end, denoting by q_i the objective function of class i used by Skmeans [6], we use the formula

$$f(A_i) = \frac{\alpha \|A_i\|_F^2 / \|A\|_F^2}{(1-\alpha) q_i / \sum_{j=1}^{l} q_j},$$

where the numerator measures the relative contribution of A_i into $\texttt{range}(A)$, since $\|A\|_F^2 = \sigma_1^2 + \ldots + \sigma_{\texttt{rank}(A)}^2$, and the denumerator measures the tightness of class i. Parameter $\alpha \in [0, 1]$ determines the relative importance assigned to each one of the aforementioned measures. In our experiments, $\alpha = 0.5$ consistently led to better results, so we use this in Section 3. Therefore, when $k > l$, the selection strategy is summarized as follows: First, l columns of $\mathcal{U}$ are filled with the maximum left singular vector of each A_i; then $f_i = f(A_i)/\sum_i f(A_i), i = 1{:}l$ are computed. Observe that $f_i \in [0, 1]$ and $\sum_{i=1}^{l} f_i = 1$. Next, the remaining columns of $\mathcal{U}$ are filled using $\texttt{round}((l-k)f_i), i = 1{:}l$ leading left singular vectors, where $\texttt{round}$ is the ceiling or floor rounding function. Step III selects the representatives, and Step IV computes the corresponding orthogonal basis.

Representatives in the kernel learning framework: Latent Semantic Kernels (LSK) were proposed as a generalization of LSI and as a global nonlinear method for TC [5]. LSK applies the same steps as LSI for $\bar{A}$, whose columns result from the application of a general (non)linear (kernel) function $\phi : \mathbb{R}^m \to \mathcal{F}$, to A's columns, resulting in a more informative representation in a new (possibly infinite dimensional) feature space. The SVD of $\bar{A}$ can be directly obtained from the eigendecomposition of the Gramian, $G := \bar{A}^\top \bar{A}$. Using appropriate symmetric

kernel functions, that satisfy Mercer's theorem [4], inner products in the new feature space can be computed without forming the new feature vectors, via the "kernel trick". For specific types of data, LSK provides better accuracy than LSI. In addition to the computational constraints of LSI, however, LSK involves further memory overhead, because the Gramian G is full. Our methodology alleviates these problems because both CLSI and CM can be formulated in terms of inner products, computed via the kernel trick. We focus on the kernel variant of CLSI and refer to [15] for CM. Let $\bar{A}_i$ denote the matrix with columns obtained after applying the kernel function, ϕ, on the columns of A_i. We make use of the fact that the local partial SVD analysis for each $\bar{A}_i$ is equivalent to partial eigenanalysis of the corresponding $G_i := \bar{A}_i^\top \bar{A}_i$. From this analysis follows an implicit representation for the leading left singular vectors of $\bar{A}_i$ from $\bar{A}_i \bar{V}_{k_i}^{(i)}$, where $\bar{V}_{k_i}^{(i)}$ contains the leading k_i eigenvectors of G_i, equivalently the right singular vectors[4] of $\bar{A}_i$. These are collected into $\bar{\mathcal{U}}$. To implement the kernel version we make use of $\bar{G} := \bar{\mathcal{U}}^\top \bar{\mathcal{U}}$. Assuming that $\bar{G}$ is positive definite, its Cholesky decomposition returns the R-factor of its QR decomposition, while the Q-factor of the QR decomposition of $\bar{\mathcal{U}}$ is implicitly represented as $\bar{Q} = \bar{\mathcal{U}}\bar{R}^{-1}$. The lower dimensional representation of the columns is $Y = (\bar{R}^{-1})^\top \bar{A}^\top \bar{A}$, while a query vector can be represented by $X(\bar{A})^\top \bar{q}$, where $X = (\bar{R}^{-1})^\top \bar{\mathcal{U}}^\top$ and $\bar{q}$ the mapping of q into $\mathcal{F}$. Just as CLSI brought several advantages in LSI, its kernel version does similarly for LSK. Moreover, there is an additional gain, a side benefit of the special form of the factor $\bar{\mathcal{V}} := \mathrm{diag}[\bar{V}_{k_1}^{(1)}, \ldots, \bar{V}_{k_l}^{(l)}]$, and thus the computation of $\bar{G} = \bar{\mathcal{U}}^\top \bar{\mathcal{U}}$ can be done blockwise, requiring only $O(n_{\max}^2)$ space, where $n_{\max}$ is the number of columns in the "widest" A_i, that is $n_{\max} = \max_i\{n_i\}$.

Numerical performance: We next comment on some issues regarding numerical robustness and computational costs, even though a detailed discussion falls outside the scope of this paper[5]. Specifically, we have implicitly assumed that the matrix of representatives, $\mathcal{U}$, has full rank. This allows us to compute the basis Q using standard QR decomposition, as was done in Table 1 for CLSI as well as in [27, 6, 16] for CM. If $\mathcal{U}$ is not full rank, we can no longer expect the straightforward QR algorithm to safely provide a basis for $\texttt{range}(\mathcal{U})$, and we need to use an alternative, such as pivoted or rank revealing QR [9]. In that case are forced to either reduce the target value of k or augment the representatives with additional elements that are not linearly dependent. The training matrix, A, might not be full rank, in which case it might also be the case that one or more A_i's is not full rank either. Indeed, it might also occur that the dimension of the column space of some A_i is smaller than the corresponding k_i, in which case some of the trailing left singular vector representatives of A_i used in $\mathcal{U}$ will correspond to zero singular values and will be parts of the basis for the null space of $A_i^\top$. This can be easily avoided by testing for zeros amongst the leading singular values of A_i and not using the corresponding left singular vectors to $\mathcal{U}$ but opting to use more singular vectors from some other group, or even reduce the value k.

It is also worth noting that tdm's are usually very sparse, therefore the centroid representation used by CM is also expected to be sparse. This is not the case for the matrix of representatives obtained via CLSI, since singular vectors of sparse matrices rarely possess any particular zero structure. We observed, however, that a very large percentage of the elements of these singular vectors is very small (around 90% in our experiments). Therefore, it is likely that we could employ sparsification, e.g. via thresholding, to "thin-out" small elements and use some approximate QR factorization. For large collections, it would also be appropriate to utilized sparse QR algorithms, Gram-Schmidt orthogonalization or even variants as in [2].

A brief description of the costs involved in CLSI is provided in [27]. CLSI alleviates the cost of computing the truncated SVD decomposition since it exchanges the calculation of leading singular vectors of A with computations of leading singular vectors of much thinner matrices.

Accelerating Linear Least Squares Fit: Linear Least Squares Fit (LLSF) is a statistical classification method based on the SVD of the training tdm [24] that has been found to lead to performance competitive to kNN and support vector machines [25]. It uses a matrix encoding of the training documents in order to produce a term-class matrix, used subsequently to map new documents to classes. Specifically, denoting by Z the class-document matrix, i.e. $\zeta_{ij} = 1$ or 0 based on whether document j belongs to class i, LLSF solves $\min_X \|XA - Z\|_F^2$. In the original version of LLSF, the least squares solution is $X = BA^\dagger$, where $A^\dagger = V_r \Sigma_r^{-1} U_r^\top$ is the pseudoinverse of A and U_r, Σ_r, V_r contain the $r := \mathrm{rank}(A)$ leading singular triplets of A. X models the association of terms to classes. In the test phase, each test document y is multiplied by X and is assigned to class(es) based on the maximum values of $\hat{y} := Xy$. For the multi-label case some thresholding strategy is used.The computational and memory costs involved in computing the pseudoinverse make the direct application of LLSF prohibitive for large problems. One way to address this obstacle is to apply truncated SVD, in approximating X [22]. An alternative that emerges naturally in our context is to deploy the representatives methodology to approximate the trun-

[4]To economize in notation, we omit the premultiplication of $\bar{A}_i \bar{V}_{k_i}^{(i)}$ with a diagonal scaling factor consisting of the inverse of the k_i leading singular values of $\bar{A}_i$.

[5]It is worth taking this opportunity to call for more studies of numerical robustness of algorithms proposed in the IR literature.

cated SVD. Let Q be the basis obtained from our methodology (Table 1) and $Y = Q^\top A$ the coefficients of the projections of A onto $\texttt{range}(Q)$. Let also $Y = \hat{U}\hat{\Sigma}\hat{V}$ be the SVD of Y. Since the columns of Q and $\hat{U}$ are orthogonal and have unit length, $(Q\hat{U})\hat{\Sigma}\hat{V}^\top$ is the SVD of $QQ^\top A$. Using $(Q\hat{U})\hat{\Sigma}\hat{V}^\top$ in place of the truncated SVD of A, we reduce the cost involved in computing the truncated SVD involved in LLSF. In Section 3 we present results that demonstrate that our proposed approach achieves high quality results compared to using the SVD.

3 Experiments

We have conducted extensive numerical experiments with the representatives methodology. We next present some of these results and evaluate the performance of the methodology, mostly focussing on CLSI, comparing with other global and local LSI-based approaches for TC. Codes were developed in MATLAB 7.0 running Windows XP on a 2.8GHz Pentium-IV having 512MB RAM. QR and partial SVD were computed using MATLAB's native `qr` and `svds` functions, the latter being based on a well-known iterative method for large sparse eigenvalue problems.

Datasets, classification algorithms and quality measures: We use two publicly available datasets: i) Reuters-21578[6], and ii) Ohsumed, a subset of the medical abstract dataset used in the TREC-9 filtering track[7]. For our purposes we used two subsets of each dataset; these we call SMODAPTE, SOHSUMED and MMODAPTE, MOHSUMED where S and M stand for single and multi-label case respectively. All collections contained at least one training and one test document for each category. That resulted in 6,495 training and 2,557 test documents classified in 52 categories for SMODAPTE, 634 training and 3,038 test documents from 63 categories for SOHSUMED, 7,672 training and 2,998 test documents classifed in 85 categories for MMODAPTE and 7,953 training and 31,027 test documents from 25 categories for MOHSUMED. To generate the tdm's we used TMG ([28]), and applied stemming, stopword elimination, and removal of terms that appeared in only one document. Based on results from [28], we used logarithmic and inverse document frequency (IDF) local and global term weighting respectively with column normalization for training documents, and binary and probabilistic weightings for test documents. For our experiments we used the well known kNN and Rocchio classifiers. For kNN, k varied depending on the collection and the nature of the dataset (single or multi-label). For single-label collections each document was assigned to the class with maximum score, while for multi-label collections we use the ScutFBR.1 method [23]. For Rocchio, we used the centroid as representative which we found to give good results and did not notice any improvements when we took into account negative examples from each class. Finally, to evaluate the methods we used both macro and micro-averaged F_1 measure [23].

[6] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[7] http://trec.nist.gov/data/t9_filtering.html

Results: In the sequel, we present comparisons of the representatives methodology to basic VSM, LSI and two local LSI methods from [14], namely L-LSI and its weighted counterpart LWR-LSI. For LSI we used $k = l{:}2l$ factors, where l stands for the number of classes for each dataset, while for local LSI methods we used $k = 10{:}10{:}50$. We compared those with CLSI for which we tried all values $k = l : 2l$. In LWR-LSI, we used the parameter setting reported in [14] as the best one. Finally, we tried various values for the k parameter of kNN classifier, namely $k = 1, 3, 5$ for the single-label datasets and $k = 10, 30, 45$ and $k = 5, 10, 15$ for MMODAPTE and MOHSUMED respectively, driven by the maximum number of classes for a single test document for each dataset (14 and 4). Optimal fbr values were first set by fixing all other parameters for each algorithm, and used for the remaining experiments. Runtimes for the multi-label datasets do not include the time consumed by the thresholding phase. Note that local methods gave better results, when combined with an initial global preprocessing step (i.e. a hybrid LSI approach).

Table 2 summarizes the highest macro and micro-averaged F_1 values for each method. We note that the kNN classifier did not seem to be sensitive to the choice of k. The representatives methodology appears to return the best accuracies for both classifiers and all datasets. In particular, CLSI and CM always improve the basic VSM, while outperforming both global and local LSI methods. The improvement is higher for the kNN classifier for the single-label datasets, while results in the multi-label case appear to be significantly improved for both kNN and Rocchio classifiers. For single-label datasets, CM gives the best results, while CLSI returned better results for the multi-label datasets. Both CLSI and CM clearly outperformed all other LSI-based TC methods. Overall, therefore, CLSI and CM gave better accuracy and F_1 values for every combination of dataset and classifier.

Fig. 1 (diagrams at the left and center columns) depicts classification quality and timings for the preprocessing steps required by CLSI, CM, LSI and L-LSI vs. approximation rank for kNN and LLSF. The corresponding plots demonstrate clearly that CLSI and CM are significantly faster. Both require much less memory. The differences are especially acute when comparing with local LSI methods. The memory problem is not seen in the diagrams but follows from the need to store l separate bases for the l subspaces associated with each class. The diagrams, instead, show that CLSI gives roughly similar quality results

Table 2. Macro and micro averaged F_1 for VSM, CLSI, CM, LSI, L-LSI and LWR-LSI. Boldface and underlined denote best macro and micro-averaged F_1 values for each method and dataset (kNN and Rocchio).

Collection		SMODAPTE		SOHSUMED		MMODAPTE		MOHSUMED	
Algorithm	Measure	kNN	Rocchio	kNN	Rocchio	kNN	Rocchio	kNN	Rocchio
VSM	Macro-F_1	0.65	0.66	0.76	0.83	0.47	0.47	0.58	0.56
	Micro-F_1	0.84	0.87	0.82	0.88	0.81	0.75	0.59	0.56
CLSI	Macro-F_1	0.74	0.66	0.82	0.82	**0.62**	**0.59**	**0.70**	0.66
	Micro-F_1	0.94	0.87	0.87	0.88	0.86	0.80	0.71	0.66
CM	Macro-F_1	**0.74**	**0.66**	**0.83**	**0.83**	0.61	0.58	0.70	**0.67**
	Micro-F_1	0.92	0.87	0.88	0.88	0.85	0.81	0.71	0.67
LSI	Macro-F_1	0.55	0.47	0.77	0.81	0.42	0.40	0.65	0.59
	Micro-F_1	0.90	0.78	0.84	0.87	0.81	0.71	0.65	0.56
L-LSI	Macro-F_1	0.52	0.44	0.77	0.81	0.44	0.41	0.57	0.57
	Micro-F_1	0.89	0.75	0.81	0.87	0.82	0.65	0.57	0.57
LWR-LSI	Macro-F_1	0.53	0.44	0.76	0.81	0.45	0.42	0.58	0.57
	Micro-F_1	0.89	0.75	0.83	0.87	0.82	0.66	0.58	0.57

for all values of the approximation rank, therefore memory does not become an issue, neither of course for CM, where the memory requirement is constant. These facts, combined with the improved accuracy of CLSI and CM make them both more suitable for TC than global, local and hybrid LSI. Finally, regarding LLSF, for a given approximation rank, both CLSI and CM returned results of superior quality than truncated SVD. Furthermore, the preprocessing times needed for the CM and CLSI based LLSF clearly outperformed LLSF based on truncated SVD. In all cases we used aggressive $n = 7,672$ training documents, implying a pruning of 98% of the factors. It is natural to expect that a larger number of factors would improve the accuracy of the truncated SVD approach but at significantly higher computational cost. Classifications using LLSF with CLSI and CM were better, with improvements being more pronounced for the macro-averaged F_1. Because testing time for LLSF is relatively small, LLSF combined with CLSI or CM appears to be competitive to kNN and Rocchio. Fig. 1 (upper and lower right diagrams) depict the attained macro and micro-averaged F_1 and runtimes for the kernel variants of CM and CLSI (denoted as KCM and KCLSI) and LSK for SMODPATE and SOHSUMED. For both datasets, CM and CLSI outperform LSK in terms of classification quality, while KCLSI has better performance for SMODAPTE and KCM for SOHSUMED, just as in the linear case. CLSI and CM are more efficient than LSK, especially for large values of approximation rank. Due to memory limitations, LSK was not able to run for the larger matrices corresponding to multi-label datasets. Overall, results for these two datasets appear to be slightly inferior than the results obtained from the linear algorithms. They show, however, that the kernel variants improve the performance of LSK, and so become good candidates for TC when the data requires the application of non-linear methods.

We conclude that the representatives methodology is a powerful tool for TC; in that context, CLSI provides a flexible and effective algorithmic substrate for its implementation.

Acknowledgments: We thank Professor Y. Yang for her kind help regarding thresholding and I. Antonellis for helpful discussions. Research supported in part by a University of Patras "Karatheodori" grant. Zeimpekis also thanks IPAM for his participation at the summer school "Intelligent Extraction of Information from Graphs and High Dimensional Data" and the Bodossaki Foundation for partial support.

References

[1] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations. In *In Proc. 33rd Annual ACM STOC*, pp. 611–618, 2001.

[2] M. W. Berry, S. A. Pulatova, and G. W. Stewart. Algorithm 844: Computing sparse reduced-rank approximations to sparse matrices. *ACM Trans. Math. Softw.*, 31(2):252–269, 2005.

[3] V. Castelli, A. Thomasian, and C.-S. Li. CSVD: Clustering and singular value decomposition for approximate similarity search in high-dimensional spaces. *IEEE Trans. Knowledge and Data Engin.*, 15(3):671–685, 2003.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, 2000.

[5] N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent semantic kernels. In C. Brodley and A. Danyluk, editors, *Proc. 18th ICMLA*, pp. 66–73. Morgan Kaufmann, San Francisco, 2001.

[6] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, Jan 2001.

[7] P. Drineas, R. Kannan, and M. Mahoney. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. Technical Report, TR-1270, Computer Science Dept., Yale Univ., February 2004.

[8] J. Gao and J. Zhang. Clustered SVD strategies in latent semantic indexing. *Information Processing and Management*, 431:10511063, 2005.

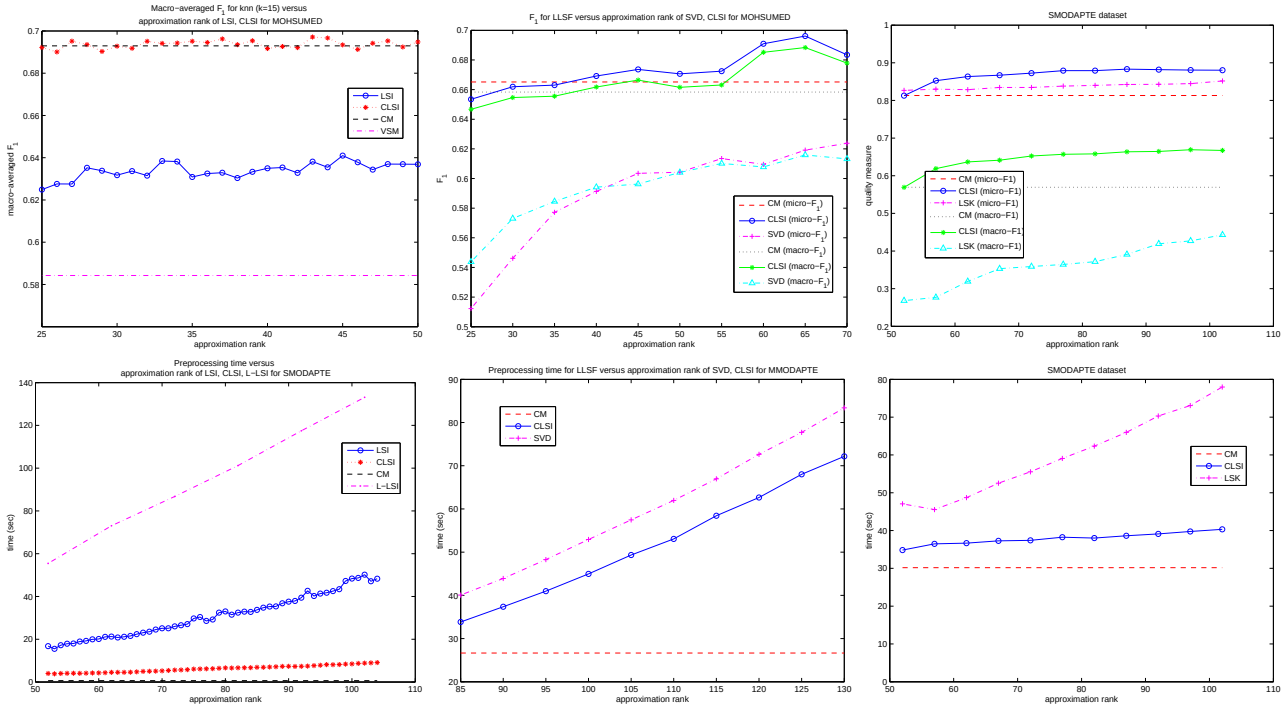

Figure 1. Classification quality and runtimes vs. approximation rank: $i)$ Left and center: LSI, L-LSI, CLSI for kNN (left) and LLSF (center). CM and VSM included for comparison purposes. $ii)$ Right: Classification quality (upper) and runtimes (lower) vs. approximation rank for (kernel) CLSI-CM, LSK.

[9] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 3d edition, 1996.

[10] D. Hull. *Information retrieval using statistical classification*. PhD thesis, Dept. Statistics, Stanford University, Palo Alto, Nov. 1994.

[11] D. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proc. 17th ACM SIGIR*, pages 282–291, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[12] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.

[13] D. Littau and D. Boley. Clustering very large data sets with principal direction divisive partitioning. In J. Kogan, C. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pp. 99–126. Springer, Berlin, 2006.

[14] T. Liu, Z. Chen, B. Zhang, W. Ma, and G. Wu. Improving text classification using local latent semantic indexing. In *Proc. ICDM'04*, pp. 162–169, Washington, 2004. IEEE Computer Society.

[15] C. Park and H. Park. Nonlinear feature extraction based on centroids and kernel functions. *Pattern Recognition*, 37(4):801–810, 2004.

[16] H. Park, M. Jeon, and J. Rosen. Lower dimensional representation of text data based on centroids and least squares. *BIT*, 43(2):427–448, 2003.

[17] Y. Qu, G. Ostrouchov, N. F. Samatova, and A. Geist. Principal component analysis for dimension reduction in massive distributed data sets. In *Workshop on High Performance Data Mining held with 2nd SIAM Int'l. Conf. Data Mining*, pp. 4–9, 2002.

[18] H. Schütze, D. Hull, and J. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proc. 18th ACM SIGIR*, pp. 229–237, New York, 1995.

[19] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surveys*, 34(1):1–47, 2002.

[20] M.-W. Wang and J.-Y. Nie. A latent semantic structure model for text classification. In *Workshop on Mathematical/Formal Methods in IR (MFIR), held during ACM-SIGIR'03*, 2003.

[21] E. Wiener, J. Pedersen, and A. Weigend. A neural network approach to topic spotting. In *Proc. SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pp. 317–332, Las Vegas, US, 1995.

[22] Y. Yang. Noise reduction in a statistical approach to text categorization. In *Proc. 18th ACM SIGIR*, pp. 256–263, New York, 1995.

[23] Y. Yang. A study of thresholding strategies for text categorization. In *Proc. 24th ACM SIGIR*, pp. 137–145, New York, 2001.

[24] Y. Yang and C. Chute. A linear least squares fit mapping method for information retrieval from natural language texts. In *Proc. 14th Conf. Computational Linguistics*, pp. 447–453, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[25] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. 22nd ACM SIGIR*, pp. 42–49, New York, 1999. ACM Press.

[26] Y. Yang, J. Zhang, and B. Kisiel. A scalability analysis of classifiers in text categorization. In *Proc. SIGIR '03*, pp. 96–103, New York, 2003.

[27] D. Zeimpekis and E. Gallopoulos. CLSI: A flexible approximation scheme from clustered term-document matrices. In *Proc. 5th SIAM Int'l. Conf. Data Mining*, pp. 631–635, Newport Beach, California, 2005.

[28] D. Zeimpekis and E. Gallopoulos. TMG: A MATLAB toolbox for generating term-document matrices from text collections. In J. Kogan, C. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data: Recent Advances in Clustering*, pp. 187–210. Springer, Berlin, 2006.

[29] S. Zelikovitz and H. Hirsh. Using LSI for text classification in the presence of background text. In *Proc. CIKM '01*, pages 113–118, New York, 2001. ACM Press.