Local Semantic Kernels for Text Document Clustering

Loulwah AlSumait and Carlotta Domeniconi
Department of Information and Software Engineering
George Mason University
lalsumai@gmu.edu, carlotta@ise.gmu.edu

Abstract

Document clustering is a fundamental task of text mining, by which efficient organization, navigation, summarization and retrieval of documents can be achieved. The clustering of documents presents difficult challenges due to the sparsity and the high dimensionality of text data, and to the complex semantics of the natural language. Subspace clustering is an extension of traditional clustering that is designed to capture local feature relevance, and to group documents with respect to the features (or words) that matter the most.

This paper presents a subspace clustering technique based on a Locally Adaptive Clustering (LAC) algorithm. To improve the subspace clustering of documents and the identification of keywords achieved by LAC, kernel methods and semantic distances are deployed. The basic idea is to define a local kernel for each cluster by which semantic distances between pairs of words are computed to derive the clustering and the local term weightings. The proposed approach, called *Semantic LAC*, is evaluated using benchmark datasets. Our experiments show that Semantic LAC is capable of improving the clustering quality.

1 Introduction

With the astonishing expansion of the internet, intranets, and digital libraries, billions of electronic text documents are made available. Extracting implicit, non-trivial, and useful knowledge from these huge corpora is essential for applications such as knowledge managements in large business enterprises, semantic web, and automatic processing of messages, emails, surveys, and news.

Document clustering is a fundamental task of text mining by which efficient organization, navigation, summarization, and retrieval of documents can be achieved. Document clustering seeks to automatically partition unlabeled documents into groups. Ideally, such groups correspond to genuine themes, topics, or categories of the corpus [19].

The classical document representation is a word-based vector, known as Vector Space Model (VSM) [16]. According to VSM, each dimension is associated with one term from the dictionary of all the words that appear in the corpus. VSM, although simple and commonly used, suffers from a number of deficiencies. Inherent shortages of VSM include breaking multi-word expressions, like *Machine Learning*, into independent features, mapping synonymous words into different components, and treating polysemous as one single component. Although traditional preprocessing of documents, such as eliminating stop words, pruning rare words, stemming, and normalization, can improve the representation, it is still essential to embed semantic information and conceptual patterns in order to enhance the prediction capabilities of clustering algorithms.

Moreover, the VSM representation of text data can easily results in tens or hundreds of thousands of features. As a consequence, any clustering algorithm would suffer from the *curse of dimensionality*. In such sparse and high dimensional space, any distance measure that assumes all features to have equally importance is likely to be not effective. This is because points within the same cluster would have at least few dimensions on which they are far apart from each other. As a result, the farthest point is expected to be almost as close as the nearest one. In order to capture local feature relevance, local operations that embed different distance measures in different regions are required. Subspace clustering is an extension of traditional clustering that is designed to group data points, i.e. documents, with respect to the features (or words) that matter the most.

In this paper, a subspace clustering technique based on the Locally Adaptive Clustering (LAC) algorithm [7] is applied. To improve the subspace clustering of documents and the identification of keywords achieved by LAC, kernel methods and semantic distances are deployed. The idea is to define a local kernel for each cluster, by which semantic distances between pairs of words are computed to derive the clustering and the local term weightings.

The rest of the paper is organized as follows. Some background on kernel methods for text is provided in Section 2. The LAC algorithm is briefly described in Section 3. In Sections 4 and 5, we present our approach, the experimental design and results. A review of recent work on the application of semantic information in kernel-based learning methods is provided in Section 6. Our final conclusions and suggestions for future work are discussed in Section 7.

2 Kernel Methods for Text

Kernel methods are a promising approach to pattern analysis. Formally, the Vector Space Model (VSM) can be defined as the following mapping

$$\phi : d \mapsto \phi(d) = (tf(t_1, d), tf(t_2, d), \ldots, tf(t_D, d)) \in \mathcal{R}^D$$

where $tf(t_i, d)$ is the frequency of term t_i in document d, and D is the size of the dictionary.

To represent the whole corpus of N documents, the Document-Term matrix, $\mathcal{D}$, is introduced. $\mathcal{D}$ is a $N \times D$ matrix whose rows are indexed by the documents and whose columns are indexed by the terms [16].

The basic idea of kernel methods is to embed the data in a suitable feature space, such that solving the problem in the new space is easier (e.g., linear). A kernel represents the similarity between two objects (e.g, documents or terms), defined as dot-product in this new vector space. The kernel trick allows to keep the mapping implicit. In other words, it is only required to know the inner products between the images of the data items in the original space. Therefore, defining a suitable kernel means finding a good representation of the data objects.

In text mining, semantically similar documents should be mapped to nearby positions in feature space. In order to address the omission of semantic content of the words in VSM, a transformation of the document vector of the type $\tilde{\phi}(d) = \phi(d)Sem$ is required, where Sem is a semantic matrix. Different choices of the matrix Sem lead to different variants of VSM. Using this transformation, the corresponding vector space kernel takes the form

$$\begin{aligned} \tilde{k}(d_1, d_2) &= \phi(d_1)SemSem^\top \phi(d_2)^\top \\ &= \tilde{\phi}(d_1)\tilde{\phi}(d_2)^\top \end{aligned} \tag{2.1}$$

Thus, the inner product between two documents d_1 and d_2 in feature space can be computed efficiently directly from the original data items using a kernel function. The semantic matrix Sem can be created as a composition of successive embeddings, which add additional refinements to the semantics of the representation. Therefore, Sem can be defined as:

$$Sem = RP$$

where R is a diagonal matrix containing the term weightings or relevances, while P is a *proximity matrix* defining the semantic similarities between the different terms of the corpus. One simple way of defining the term weighting matrix R is to use the inverse document frequency (idf). In this paper, a new weighting measure, dynamically learned by means of the LAC algorithm, is used to construct R.

P has non-zero off diagonal entries, $P_{ij} > 0$, when the term i is semantically related to the term j. Embedding P in the vector space kernel corresponds to representing a document as a less sparse vector, $\phi(d)P$, which has non-zero entries for all terms that are semantically similar to those present in document d. There are different methods for obtaining P. A semantic network, like *WordNet*, which encodes relationships between words of a dictionary in a hierarchical fashion, is one source of term similarity information. In alternative, the proximity matrix can be computed using *Latent Semantic Indexing* (LSI). The Singular Value Decomposition (SVD) of the matrix $\mathcal{D}^\top$ is calculated to extract the semantic information, and to project the documents into the space spanned by the first k eigenvectors of the $\mathcal{D}^\top\mathcal{D}$ matrix. The corresponding kernel is called Latent Semantic Kernel (LSK). The simplest method to compute P, and used in this paper, is the *Generalized VSM* (GVSM) [21]. This technique aims at capturing correlations of terms by investigating their co-occurrences across the corpus. Two terms are considered semantically related if they frequently co-occur in the same documents. Thus, a document is represented by the embedding

$$\tilde{\phi}(d) = \phi(d)\mathcal{D}^\top$$

and the corresponding kernel is

$$\tilde{k}(d_1, d_2) = \phi(d_1)\mathcal{D}^\top\mathcal{D}\phi(d_2)^\top \tag{2.2}$$

where the $(i, j)^{th}$ entry of the matrix $\mathcal{D}^\top\mathcal{D}$ is given by

$$(\mathcal{D}^\top\mathcal{D})_{ij} = \sum_{i,j} tf(t_i, d)tf(t_j, d)$$

The matrix $\mathcal{D}^T\mathcal{D}$ has a nonzero entry $(\mathcal{D}^T\mathcal{D})_{ij}$ if there is a document d in which the corresponding terms t_i and t_j co-occur, and the strength of the relationship is given by the frequency and the number of their co-occurrences.

3 Locally Adaptive Clustering (LAC)

In LAC [7] (a preliminary version appeared in [8]), a weighted cluster is defined as a subset of data points, together with a weight vector $\mathbf{w}$, such that the points are closely clustered according to the corresponding weighted Euclidean distance. w_j measures the degree of correlation of points along dimension j. The objective of LAC is to find cluster centroids, and weight vectors.

The partition induced by discovering weighted clusters is formally defined as follows. Given a set S of N points $\mathbf{x} \in \mathcal{R}^D$, a set of k centers $\{\mathbf{c}_1, ..., \mathbf{c}_k\}$, $\mathbf{c}_j \in \mathcal{R}^D$, $j = 1, ..., k$, coupled with a set of corresponding weight vectors $\{\mathbf{w}_1, ..., \mathbf{w}_k\}, \mathbf{w}_j \in \mathcal{R}^D, j = 1, ..., k$, partition S into k sets:

$$S_j = \{\mathbf{x} | (\sum_{i=1}^{D} w_{ji}(x_i - c_{ji})^2)^{\frac{1}{2}} \quad < \quad (\sum_{i=1}^{D} w_{qi}(x_i - c_{qi})^2)^{\frac{1}{2}}, \quad \forall \quad q \neq j\}$$

where w_{ji} and c_{ji} represent the i^{th} components of vectors $\mathbf{w}_j$ and $\mathbf{c}_j$, respectively.

The set of centers and weights is optimal with respect to the Euclidean norm, if they minimize the error measure:

$$E_1(C, W) = \sum_{j=1}^{k} \sum_{i=1}^{D} (w_{ji} \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2)$$

subject to the constraints $\forall j \sum_i w_{ji} = 1$. C and W are $(D \times k)$ matrices whose columns are $\mathbf{c}_j$ and $\mathbf{w}_j$ respectively, i.e. $C = [\mathbf{c}_1 ... \mathbf{c}_k]$ and $W = [\mathbf{w}_1 ... \mathbf{w}_k]$, and $|S_j|$ is the cardinality of set S_j. The solution

$$(C^*, W^*) = \arg \min_{C,W} E_1(C, W)$$

will discover one-dimensional clusters: it will put maximal (unit) weight on the feature with smallest dispersion $\frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2$, within each cluster j, and zero weight on all other features. In order to find weighted multidimensional clusters, where the unit weight gets distributed among all features according to the respective dispersion of data within each cluster, LAC adds the regularization term $\sum_{i=1}^{D} w_{ji} \log w_{ji}$. This term represents the negative entropy of the weight distribution for each cluster. It penalizes solutions with maximal weight on the single feature with smallest dispersion within each cluster. The resulting error function is

$$E(C, W) = \sum_{j=1}^{k} \sum_{i=1}^{D} (w_{ji} \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2 + h w_{ji} \log w_{ji})$$

subject to the constraints $\forall j \sum_i w_{ji} = 1$. The coefficient $h \geq 0$ is a parameter of the procedure; it controls the strength of the incentive for clustering on more features. Increasing its value will encourage clusters on more features, and vise versa. By introducing the Lagrange multipliers, the solution of this constrained optimization problem is

$$(3.3) \qquad w^*_{ji} = \frac{\exp\left(\frac{-\frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2}{h}\right)}{\sum_{i=1}^{D} \exp\left(\frac{-\frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2}{h}\right)}$$

$$(3.4) \qquad c^*_{ji} = \frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} x_i$$

To find a partition that identifies the clustering solution, a search strategy that progressively improves the quality of initial centroids and weights is proposed. The search starts with *well-scattered* points in S as the k centroids, and weights equally set to $1/D$. Then, for each centroid $\mathbf{c}_j$, the corresponding sets S_j are computed as previously defined. Next, the average distance of the points in S_j to the centroid $\mathbf{c}_j$, along each dimension, is computed. The smaller the average, the larger the correlations of points along dimension i. To credit weights to features (and to clusters), the average distances are used, as given above. Consequently, the computed weights are used to update the sets S_j , and therefore the centroids' coordinates. The procedure is iterated until convergence is reached.

4 Semantic LAC

The local weights provided by LAC are exploited to identify the keywords specific to each topic (cluster). To improve the subspace clustering of documents and the computation of local term weights, the learning paradigm of kernel methods is used. The idea is to use a semantic distance between pairs of words by defining a local kernel for each cluster, derived from Equation (2.1), as follows:

$$(4.5) \qquad K_j(d_1, d_2) = \phi(d_1) Sem_j Sem_j^\top \phi(d_2)^\top$$

where $Sem_j = R_j P$. R_j is a local term weighting diagonal matrix corresponding to cluster j, and P is the proximity matrix between the terms. The weight vector that LAC learns is used to construct the weight matrix R for each cluster. Formally, R_j is a diagonal matrix where $r_{ii} = w_{ji}$, i.e. the weight of term i for cluster j, for $i = 1, \ldots, D$, and can be expressed as follows:

$$R_j = \begin{pmatrix} w_{j1} & 0 & ... & 0 \\ 0 & w_{j2} & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & w_{jD} \end{pmatrix}$$

To compute P, GVSM is used. Since P holds a similarity figure between words in the form of co-occurrence information, it is necessary to transform it to a distance measure before utilizing it. To this end, the values of P are recomputed as follows:

$$P_{ij}^{dis} = 1 - (P_{ij}/max(P))$$

where $max(P)$ is the maximum entry value in the proximity matrix.

4.1 Semantic LAC Algorithm Similarly to the LAC algorithm described in Section 3, Semantic LAC starts with k initial centroids and equal weights. It partitions the data points, recomputes the weights and data partitions accordingly, and then recomputes the new centroids. The algorithm iterates until convergence or a maximum number of iterations is exceeded. Semantic LAC uses a semantic distance. A point $\mathbf{x}$ is assigned to the cluster j that minimizes the semantic distance of the point from its centroid. The semantic distance is derived from the kernel in (4.5) as follows:

$$L_w(\mathbf{c}_j, \mathbf{x}) = (\mathbf{x} - \mathbf{c}_j) Sem_j Sem_j^\top (\mathbf{x} - \mathbf{c}_j)^\top$$

Thus, every time the algorithm computes S_j's, the semantic matrix must be computed by means of the new weights. The resulting algorithm, called Semantic LAC, is summarized in **Algorithm 1**.

Algorithm 1 Semantic LAC

```
Input: $N$ points $\mathbf{x} \in \mathcal{R}^D$, $k$, and $h$
1. Initialize $k$ centroids $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$
2. Initialize the weights, $w_{ji} = \frac{1}{D}$, for each centroid
   $\mathbf{c}_j$, $j = 1, \dots, k$, and for each feature $i = 1, \dots, D$
3. Compute $P$; then compute $P^{dis}$
4. Compute $Sem$ for each cluster $j$:
      $Sem_j = R_j P^{dis}$
5. For each centroid $\mathbf{c}_j$, and for each point $\mathbf{x}$, set:
      $S_j = \{\mathbf{x} | j = \arg\min_l L_w(\mathbf{c}_l, \mathbf{x})\}$,
      where $L_w(\mathbf{c}_l, \mathbf{x}) = (\mathbf{x} - \mathbf{c}_l) Sem_l Sem_l^\top (\mathbf{x} - \mathbf{c}_l)^\top$
6. Compute the new weights:
      for each centroid $\mathbf{c}_j$, and for each feature $i$:
      - $w_{ji} = \frac{\exp \frac{\frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2}{h}}{\sum_{i=1}^{D} \exp \frac{\frac{1}{|S_j|} \sum_{\mathbf{x} \in S_j} (c_{ji} - x_i)^2}{h}}$;
7. For each centroid $\mathbf{c}_j$, and for each point $\mathbf{x}$:
      - Recompute $Sem$ matrix using new weights;
      - Recompute $S_j = \{\mathbf{x} | j = \arg\min_l L_w(\mathbf{c}_l, \mathbf{x})\}$
8. Compute the new centroids:
      Set $\mathbf{c}_j = \frac{\sum_{\mathbf{x}} \mathbf{x} 1_{S_j}(\mathbf{x})}{\sum_{\mathbf{x}} 1_{S_j}(\mathbf{x})}$ for each $j = 1, \dots, k$,
      where $1_s(.)$ is the indicator function of Set $S$
9. Iterate 5, 6, 7, and 8 until convergence, or
      maximum number of iterations is exceeded
```

The running time of one iteration of LAC is $O(kND)$. For Semantic LAC, local kernels are computed for each cluster, based on semantic distances between pairs of terms. Thus, the running time of one iteration becomes $O(kND^2)$. However, we perform feature selection in our experiments, and reduce D to the hundreds. In addition, as discussed later, Semantic LAC reaches better solutions in fewer iterations than LAC in general. Thus, by stopping the execution of Semantic LAC when a maximum number of iterations is reached, we are able to limit the computational burden without affecting accuracy.

5 Experimental Results

5.1 Datasets The datasets used in our experiments, see Table 1, were preprocessed according to the following steps: removal of stop words, stemming of words to their root source, and removal of rare words that appeared in less than four documents. A global feature selection algorithm, called DocMine, which is based on frequent itemset mining, was also performed [3]. Briefly, DocMine mines the documents to find the frequent itemsets, which are sets of words that co-occur frequently in the corpus, with a given support level (SL). In principle, the support level is driven by the target dimensionality of the data. The union of such frequent items is used to represent each document as a bag of frequent itemsets. The weight of the new entry is the frequency of the corresponding word in the document [11, 3]. In the following, we provide a short description of the datasets.

Email-1431. The original Email-1431 corpus consists of texts from 1431 emails manually classified into three categories: conference (370), job (272) and spam (789). The total dictionary size is 38713 words. In this paper, we consider a 2-class classification problem by combining the conference and job emails into one class (Nonspam). In addition, the set with 285 features, that corresponds to 10% support level, is used.

Ling-Spam. This dataset is a mixture of spam messages (453) and messages (561) sent via the linguist list, a moderated list concerning the profession and science of linguistics. The original size of the dictionary is 24627. In our experiments, the sets with 350, 287, 227, and 185 features were used, corresponding to 7%, 8%, 9%, and 10% support level, respectively.

20NewsGroup. This dataset is a collection of 20,000 messages collected from 20 different netnews newsgroups. In this paper, two-class classification problems are considered using the following two categories: Auto (990 documents) and Space (987 documents); and Electronics (981 documents) and Medical (990 documents). The dimension of the former set is 366 which correspond to 3% support level, while the latter set has 134 features with 5% support level.

Classic3. This dataset is a collection of abstracts from three categories: MEDLINE (1033 abstracts from medical journals), CISI (1460 abstracts from IR papers); CRANFIELD (1399 abstracts from aerodynamics papers). We consider 4 problems constructed from the Classic3 set, which consist of 584 ($SL = 2\%$), 395 ($SL = 3\%$), 277 ($SL = 4\%$), and 219 ($SL = 5\%$) features, respectively.

Table 1: Characteristics of the datasets

Dataset	k	SL	D	N	*points/class*
Email1431	2	10	285	1431	S(789), NS(642)
LingSpam(10%)	2	10	185	1014	S(453), NS(561)
LingSpam(9%)	2	9	227	1014	S(453), NS(561)
LingSpam(8%)	2	8	287	1014	S(453), NS(561)
LingSpam(7%)	2	7	350	1014	S(453), NS(561)
Auto-Space	2	5	166	1977	A(990), S(987)
Medical-Electric	2	5	134	1971	M(990), E(981)
Classic3(5%)	3	5	219	3892	Med(1033), Cran (1399), Cisi(1460)
Classic3(4%)	3	4	277	3892	Med(1033), Cran (1399), Cisi(1460)
Classic3(3%)	3	3	395	3892	Med(1033), Cran (1399), Cisi(1460)
Classic3(2%)	3	2	584	3892	Med(1033), Cran (1399), Cisi(1460)

5.2 Results We ran LAC 6 times on all the datasets for $1/h = 1, \ldots, 6$. Table 2 lists the average error rate, standard deviation, and the minimum error rate obtained by running Semantic LAC on all the datasets, along with the corresponding results of the LAC algorithm, and K-Means as baseline comparison. Figures 1 and 2 illustrate the error rates of Semantic LAC and LAC as a function of the h parameter values for the Classic3 (3%) and NewsGroup/Medical-Electronic, respectively. Error rates are computed according to the confusion matrices using the ground truth labels.

LAC and Semantic LAC provided superior partitionings of the data with respect to K-Means for all the datasets. As a further enhancement, Semantic LAC provided error rates lower than LAC for all the datasets, many of which, as in Ling-Spam and Classic3, are with major improvements. Although, in some cases, LAC found solutions with minimum error rates, e.g. Newsgroup/Auto-Space and Classic3 2% and 3%, Semantic LAC, on average, performed better. In addition, the standard deviations of the error rates for Semantic LAC were significantly smaller than those of LAC, which demonstrates the stability of our subspace clustering approach when semantic information is embedded. This is a relevant result since the setting of the h parameter is an open problem, as no domain knowledge for its tuning is likely to be available in practice. Furthermore, parameter tuning is a difficult problem for clustering in general. Thus, the achievement of robust clustering is a highly desirable result.

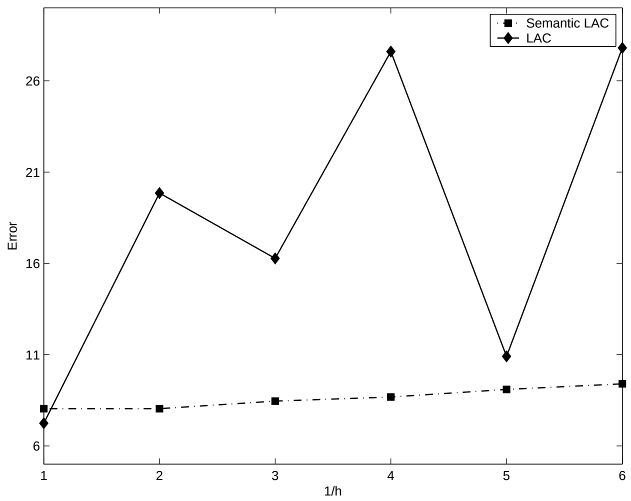

Figure 1: Error rate vs. h values for Semantic LAC and LAC on Classic3 3%.

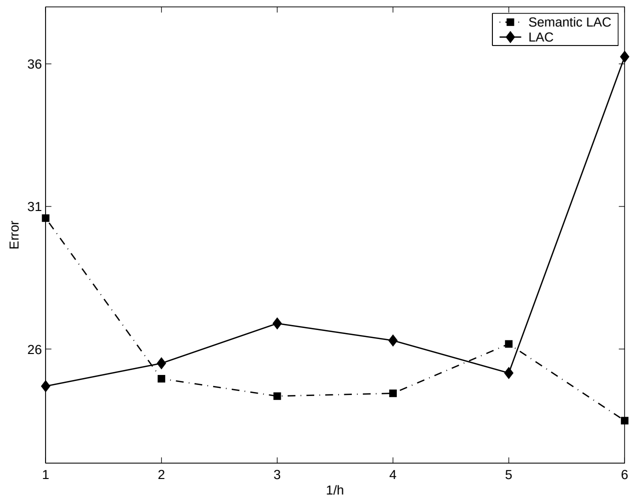

Figure 2: Error rate vs. h values for Semantic LAC and LAC on NewsGroup20/Medical-Electrical.

We further investigated the behavior of Semantic LAC using different support levels. Figures 3 and 4 illustrate the error rate vs. h values for different support levels on the Classic3 data, using Semantic LAC and LAC, respectively. For increasing support values, i.e. decreasing number of selected features, Semantic LAC resulted in a clear increasing trend for the minimum rate, as well as the average error rate. At the same time, no clear trend can be observed for LAC, (see Figure 4). As expected, as more features are used to

Table 2: Experimental results of Semantic LAC compared to LAC and K-Means

	LAC			Sem LAC			K-Means		
DataSet	Ave Error	Std Dev	Min Error	Ave Error	Std Dev	Min Error	Ave Error	Std Dev	Min Error
Email1431	2.12	0.45	1.54	1.7	0.21	1.50	42.85	3.95	40.01
LingSpam 10%	6.33	0.28	5.96	4.67	0.26	4.24	20.07	19.16	6.31
LingSpam 9%	10.8	12.7	5.3	3.63	0.08	3.55	23.47	19.78	8.97
LingSpam 8%	5.5	0.9	4.0	3.18	0.18	3.06	20.15	18.74	7.40
LingSpam 7%	12.2	12.1	5.2	3.1	3.06	5.4	31.69	19.09	6.71
Auto-Space	28.7	6.35	24.7	27.6	2.02	25.0	42.85	3.95	40.01
Medical-Electric	27.47	4.37	24.7	25.67	2.61	24.35	44.83	2.96	42.89
Classic3 5%	24.54	10.97	12.25	10.79	0.32	10.38	27.22	11.44	12.79
Classic3 4%	10.18	0.81	9.0	9.36	0.33	8.99	29.05	9.24	10.63
Classic3 3%	18.28	8.49	7.24	8.46	0.45	8.04	28.44	9.59	9.27
Classic3 2%	11.8	7.3	5.9	7.15	0.5	6.45	23.03	16.13	8.58

represent the documents, better clustering results are obtained with semantic LAC, and embedding semantic information within distance metrics further enhanced the clustering quality.

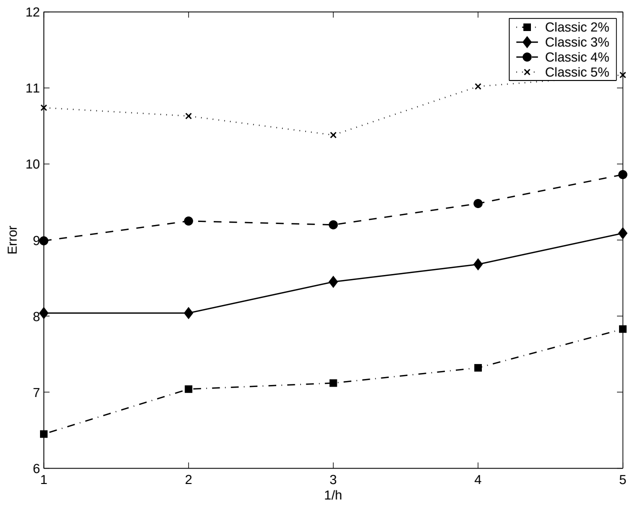

Figure 3: Error rate vs. h values for different support levels on Classic3 data using Semantic LAC.

Moreover, lower error rates were achieved for higher h, i.e. lower $1/h$ (Figure 3), which favors multi-dimensional clusters. Nonetheless, the trend is slightly different for Ling-Spam dataset (Figure 5). The average and minimum error rates have the same increasing trend with respect to the support level but, in general, lower error rates resulted from lower h, i.e. higher $1/h$. In general, it is expected that the optimal dimensionality depends on the nature of the dataset. The Non spam emails in the Ling-Spam data comes from one narrow area of linguistics, so fewer words are required to correctly identify the class. On the other hand, Classic3, although collected from three different areas, is basically

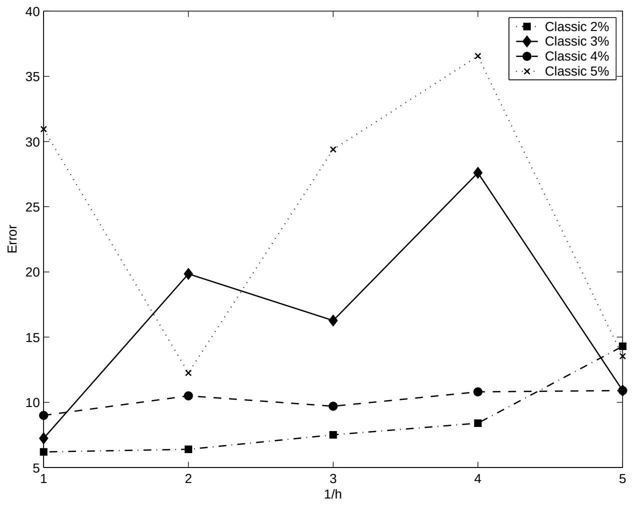

Figure 4: Error rate vs. h values for different support levels on Classic3 data using LAC.

a collection of scientific journal abstracts. Therefore, many words may be shared across classes, and the algorithm requires more features to correctly identify the classes.

Finally, we compared the convergence behavior of Semantic LAC and LAC. Results are shown in Figures 6, 7, and 8. To partially eliminate the randomness of the initial centroids, the figures plot the error rate starting at iteration three for both LAC and Semantic LAC. Figure 6 shows the error rate at every iteration of a single run for the Classic3 5% set. Figures 7 and 8 illustrate the error rate at each iteration averaged over six runs for the LingSpam 10% and the Medical-Electric datasets, respectively. Although one iteration of Semantic LAC requires more computations, Semantic LAC does converge to a stable error value in fewer iterations. In fact, on average, Semantic LAC reaches

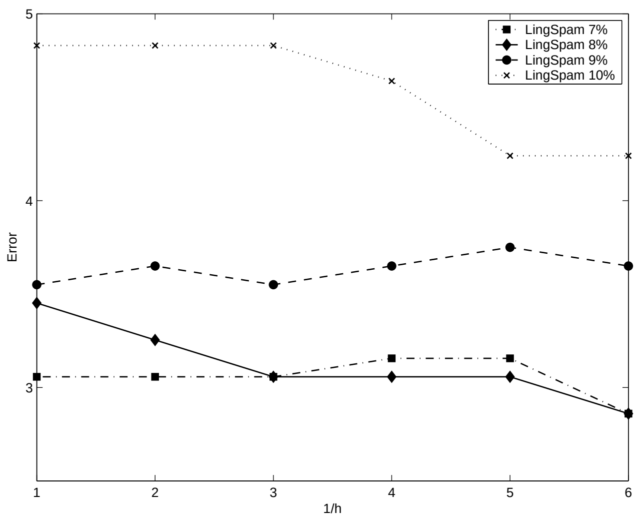

Figure 5: Error rate vs. h values for different support levels on Ling-Spam data using Semantic LAC.

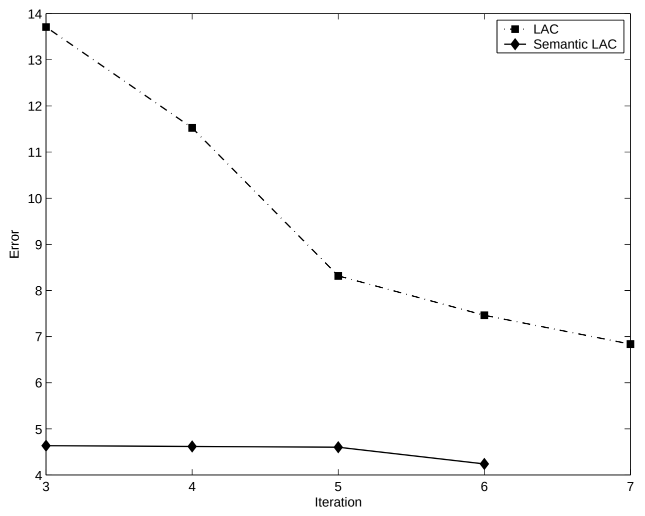

Figure 7: Average error rate vs. number of iterations for LAC and Semantic LAC on LingSpam 10%.

better error rates in less than six iterations. Particularly, the results reported in Table 2 are executed with the maximum threshold for the number of iterations set to five. This demonstrate the potential of using local kernels that embed semantic relations as a similarity measure.

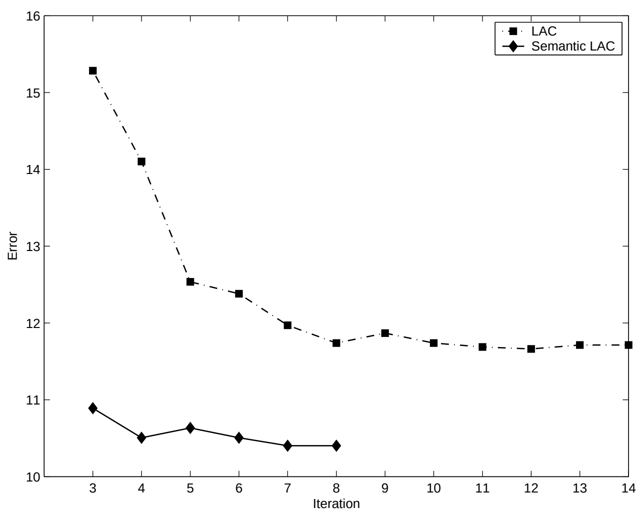

Figure 6: Error rate vs. number of iterations for LAC and Semantic LAC on Classic3 5%.

6 Related Work

There exists a number of subspace clustering algorithms in the literature of data mining. Examples of such algorithms include the CLIQUE algorithm [2], Density-based Optimal projective Clustering (DOC) [15], and Projected Clustering (PROCLUS) [1]. A comparative survey of subspace clustering techniques can be found in [14].

The problem of embedding semantic information within the document representation and/or distance metrics has recently attracted a lot of attention. Most of the work focused on text classification, e.g. [4, 5, 6, 12]. In particular, Cristianini et al. [6] introduced and developed latent semantic kernels (LSK), as described earlier, and applied their novel approach on multi-class text classification problems. LSKs were tested using Reuters21578, and some improvement was reported. The authors in [17] have also combined a semantic kernel with SVMs to define a new similarity measure for text classification. To identify semantic proximities between words, WordNet was utilized: the length of the path linking each possible pair of words in the taxonomy was used to measure the similarity. By incorporating the proximities within the VSM, documents were represented by new less sparse vectors and, hence, a new distance metric was induced and integrated into K-NN and SVMs. While the work in [17] reported improvement in terms of accuracy when the semantic kernel was deployed, the task involves supervised learning.

Hotho et al. [9] integrated conceptual account of terms found in WordNet to investigate its effects when deployed for unsupervised document clustering. They introduced different strategies for disambiguation, and applied Bi-Section-KMeans, a variant of KMeans clustering [18], with the cosine similarity measure on the Reuters21578 corpus. Similarly, [20] deployed WordNet to define a sense disambiguation method based on the semantic relatedness among the senses which was used in basic document clustering algorithms, e.g. K-

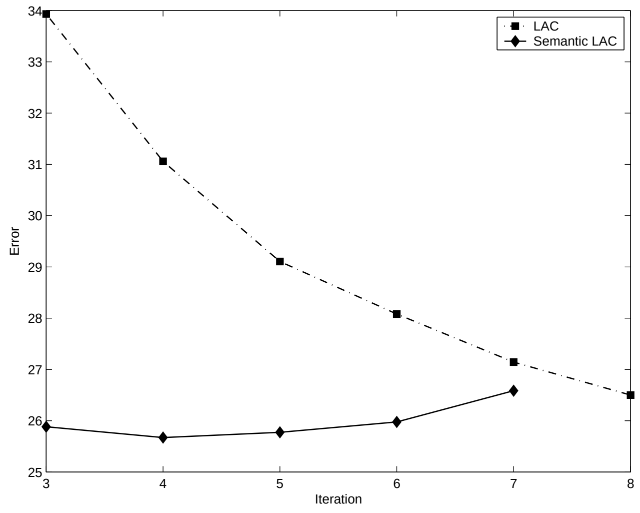

Figure 8: Average error rates vs. number of iterations for LAC and Semantic LAC on Medical-Electric

means, bisecting K-means, and HAC. They found that the use of senses of words can improve the clustering performance, but the improvement was statistically insignificant.

Recently [10], a subspace clustering approach that uses an ontology-based semantic distance has been proposed. In this approach, an ontology-based VSM is constructed from the original term-based VSM by means of a Principal-Component-Analysis-like factorization of a term mutual information matrix. This approach captures the semantic relationship between each pair of terms based on the WordNet ontology. Similarly to LAC, the work in [10] applied the new representation to a subspace clustering algorithm, extended from the standard KMeans. It identifies clusters by assigning large weights to the features that form the subspaces in which the clusters exist. The work in [10] generates fixed weights prior to the clustering process. On the other hand, our semantic distance measure is driven by the local term weighting within each cluster, and the clusters together with the local weights are derived and enhanced by the embedded semantic, iteratively during the clustering process.

7 Conclusion and Future Work

In this paper, the effect of embedding semantic information within subspace clustering of text documents is investigated. In particular, a semantic distance based on a GVSM kernel approach is embedded in a locally adaptive clustering algorithm to enhance the subspace clustering of documents, and the identification of relevant terms. Results have shown improvements over the original LAC algorithm in terms of error rates for all datasets tested. In addition, the semantic distances resulted in more robust and stable subspace clusterings.

The proposed approach requires more analysis and exploration. In particular, in our future work we plan to perform more experiments using different datasets and various feature selection approaches. In addition, other kernel methods, e.g. semantic smoothing of VSM, LSK, and diffusion kernels, may provide more sophisticated semantic representations. Furthermore, an analysis of the distribution of the terms' weights produced by Semantic LAC may identify the keywords that best represent the semantic topics discussed in the documents.

Acknowledgments

This work was in part supported by NSF CAREER Award IIS-0447814.

References

[1] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. *Fast algorithms for projected clustering*, In Proceedings of the 1999 ACM SIGMOD international conference on Management of data, (1999), pp. 61-72.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, *Automatic subspace clustering of high dimensional data for data mining applications*, In Proceedings of the 1998 ACM SIGMOD international conference on Management of data, (1998), pp. 94-105.

[3] D. Barbara, C. Domeniconi, and N. Kang, *Classifying Documents Without Labels*, in Proceedings of the SIAM International Conference on Data Mining , Lake Buena Vista, Florida, (2004).

[4] R. Basili, M. Cammisa, and A. Moschitti, *A Semantic Kernel to classify texts with very few training examples*, W4: Learning in Web Search, at the 22nd International Conference on Machine Learning, Bonn, Germany, (2005).

[5] R. Basili, M. Cammisa and A. Moschitti, *A Semantic Kernel to exploit Linguistic Knowledge*, In proceedings of the 9th Conference of the Italian Association for Artificial Intelligence (AI*IA 2005), Lecture notes in Computer Science - Springer Verlag, (2005).

[6] N. Cristianini, J. Shawe-Taylor, and H. Lodhi, *Latent Semantic Kernels*, Journal of Intelligent Information Systems, Springer Netherlands. 18(2-3), (2002), pp. 127-152.

[7] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan, and D. Papadopoulos, *Locally Adaptive Metrics for Clustering High Dimensional Data*, Data Mining and Knowledge Discovery Journal, Springer. Published on line January 26, 2007

[8] C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma, *Subspace Clustering of High Dimensional Data*, In Proceedings of the SIAM International Conference on Data Mining, 2004.

[9] A. Hotho, S. Staab, and G. Stumme, *WordNet improves Text Document Clustering*, Proc. Of the Semantic Web Workshop at SIGIR-2003, 26th Annual International ACM SIGIR Conference, (2003).

[10] L. Jing, L. Zhou, M. K. Ng, and J. Zhexue Huang, *Ontology-based Distance Measure for Text Clustering*, Proceedings of the Fourth Workshop on Text Mining, the Sixth SIAM International Conference on Data Mining, Bethesda, Maryland, (2006).

[11] N. Kang, C. Domeniconi, and D. Barbara, *Categorization and Keyword Identification of Unlabeled Documents*, in Proceedings of the Fifth IEEE International Conference on Data Mining , Huston, Texas, (2005).

[12] C. H. Lee and H. C. Yang, *A Classifier-based Text Mining Approach for Evaluating Semantic Relatedness Using Support Vector Machines*, IEEE International Conference on Information Technology: Coding and Computing (ITCC'05), I, (2005), pp. 128-133.

[13] D. Mochihashi, G. Kikui, and K. Kita, *Learning Nonstructural Distance Metric by Minimum Cluster Distortions*, Proceedings of the conference of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL04). (2004).

[14] L. Parsons, E. Haque, and H. Liu, *Evaluating Subspace Clustering Algorithms.* Workshop on Clustering High Dimensional Data and its Applications, SIAM International Conference on Data Mining (SDM 2004). (2004), pp. 48-56.

[15] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali, *A Monte Carlo algorithm for fast projective clustering*, In Proceedings of the 2002 ACM SIGMOD international conference on Management of data, (2002), pp. 418-427.

[16] John Shawe-Taylor and Nello Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.

[17] G. Siolas , F. d'Alché-Buc, *Support Vector Machines Based on a Semantic Kernel for Text Categorization*, Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), 5, (2000), pp. 5205.

[18] M. Steinbach, G. Karypis, and V. Kumar, *A Comparison of Document Clustering Techniques.* University of Minnesota. Technical Report #00-034. (2000).

[19] P. Tan, M. Stinbach, and V. Kumar, *Introduction to Data Mining*, Pearson Addison Wesley, 2006.

[20] Y. Wang and J. Hodges, *Document Clustering with Semantic Analysis*, In Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06), 3, (2006), pp.54c.

[21] S. K. M. Wong, W. Ziarko, and P. C. N. Wong, *Generalized vector space model in information retrieval*, In ACM SIGIR Conference on Research and Development in Information Retrieval, (1985), pp. 18-25.