A concept lattice-based kernel for SVM text classification

Claudio Carpineto, Carla Michini, and Raffaele Nicolussi

Fondazione Ugo Bordoni, Rome, Italy
{carpinet, cmichini, rnicolussi}@fub.it

Abstract. Standard Support Vector Machines (SVM) text classification relies on bag-of-words kernel to express the similarity between documents. We show that a document lattice can be used to define a valid kernel function that takes into account the relations between different terms. Such a kernel is based on the notion of conceptual proximity between pairs of terms, as encoded in the document lattice. We describe a method to perform SVM text classification with concept lattice-based kernel, which consists of text pre-processing, feature selection, lattice construction, computation of pairwise term similarity and kernel matrix, and SVM classification in the transformed feature space. We tested the accuracy of the proposed method on the 20NewsGroup database: the results show an improvement over the standard SVM when very little training data are available.

1 Introduction

Kernel-based learning methods are being actively investigated because they permit to decouple the problem of choosing a suitable feature space from the design of an effective learning algorithm. The idea is to use a linear algorithm to solve a non-linear problem by mapping the original features into a higher-dimensional space where the linear algorithm is subsequently used. A key enabling factor is that kernel methods exploit inner products between all pairs of data items and that such products can be often computed without explicitly representing the transformed, high-dimensional feature space.

Support Vector Machines (SVM) is probably the best known learning algorithm based on kernel, with text classification being one of its most natural applications because retrieval techniques are based just on the inner-products between vectors. While SVMs with bag-of-words kernel have shown to perform well ([10], [12]), they are limited by their inability to consider relations between different terms. If, due to the vocabulary problem, two documents refer to the same issue using different terms, such documents would be mapped to distant regions of the transformed feature space.

The question is whether it is possible to define a semantically-enriched document similarity measure and to embed it in a kernel-defined feature space. This issue has been addressed in a few earlier works, mainly using latent semantic indexing [6] and WordNet [16] as knowledge sources. Our work is in the same

research line, with the difference that we take an approach based on formal concept analysis.

We use a document lattice (i.e., the concept lattice associated with the training documents) to discover relations between the terms in the documents. The relation between two terms is determined using the shortest path (topological distance) between the corresponding attribute concepts in the document lattice; the closer the two attributes are to each other, the greater their semantic relation. Such relations between terms can then be easily incorporated into a *valid* document similarity kernel function, i.e., such that it can be rewritten as an inner product $K(x, y) = \langle \phi(x), \phi(y) \rangle$ for some $\phi : X \rightarrow F$ defining a feature space.

Note that although document lattices have been used in several information retrieval tasks, this is the first attempt, to the best of our knowledge, to investigate the potentials of a concept lattice-based kernel for text. Likewise, the exploitation of structural interdocument similarities to expand the document representation is a novel approach to defining semantic kernels for text.

Following this idea, we have implemented a full SVM text classification system with concept lattice-based kernel. It consists of several steps, namely text pre-processing, feature selection, lattice construction, computation of pairwise term proximity and kernel matrix, and finally SVM classification in the transformed feature space. Each step is described in detail in the paper.

The proposed method has been evaluated by comparing its accuracy to that of SVM with standard kernel. We used the 20 NewsGroup dataset and simulated critical learning conditions with little training data. We found that in this situation the concept lattice-based kernel was more effective.

The remaining of the paper has the following organization. We first introduce SVM and kernels for text, showing the general formulation of a document similarity function used as a kernel. Then we present our specific kernel function based on pairwise term proximities extracted from a document lattice, and we describe the full SVM text classification system in which it has been been embodied. The next sections are about experimental results and related work. We finally offer some conclusions and future research directions.

2 SVM and kernels for text

Let us define $\{(d_i, c_i), i = 1...l\}$ the training set for a binary classification problem, where each d_i is a document described by a set of terms and c_i is either 1 or -1, indicating the class to which the document d_i belongs. Support Vector Machines construct the separating hyperplane that maximizes the *margin* between the two sets of document vectors, i.e., such that it hat has the largest distance to the neighboring documents of both classes. From the definition of Support Vector Machines and the kernel theory ([17], [15]), the decision function is defined as:

$$f(d) = sgn(\sum_{i=1}^{l} \alpha_i \; c_i \; K(d, d_i) + b) \tag{1}$$

where d is the document to be classified, the vector α and the scalar b are the parameters of the maximum margin hyperplane, and K is a document similarity function that satisfies the Mercer's condition. The Mercer's condition states that the Gram matrix $G = K(d_i, d_j), \forall i, j = 1...l$, must be symmetric and positive definite. Under this condition, the function K is a valid kernel.

One of the most popular (Mercer) kernels is the linear kernel, which corresponds to the mapping $\phi(d) = d$. Using the bag-of-words model, the feature space is defined by the terms used to index the documents, and the linear kernel is given by the inner product between the document feature vectors, i.e.

$$K(d_1, d_2) = \langle d_1, d_2 \rangle = d_1^T d_2 \tag{2}$$

where $TD = [d_1..d_l]$ is the classic term by document matrix, whose columns are the documents and whose rows are the terms. Each element $TD(i, j)$ is equal to the number of occurrences of term i in document j, or to a term weight which best reflects the importance of each term in each document. In general, the matrix TD is sparse, as the fraction of elements for which $TD(i, j) = 0$, meaning that term i is not contained in document j, is very high. With this model, the Gram matrix is just the document by document matrix $G = TD^T TD$.

Another classical kernel is the Gaussian kernel, which is given by:

$$K(d_1, d_2) = exp\ (-\ \gamma\ ||d_1 - d_2||^2) \tag{3}$$

It can be shown that a Gaussian kernel performs a mapping into an infinite dimensional space, which can better handle the case when both classes are not linearly separable in the input space and often yields better performance than the linear kernel.

Instead of using the bag-of-words model with the original input features, it is possible to consider a linear mapping of the document vectors $\phi(d) = Pd$, where P is any appropriately shaped matrix. In this case, a valid (linear) kernel is given by:

$$K(d_1, d_2) = d_1^T \; P^T \; P \; d_2 \tag{4}$$

because the corresponding Gram matrix is symmetric and positive definite [6]. The matrix P typically encodes pairwise term similarities, thus implying that the mapping $\phi(d) \; = P \; d$ allows to represent each document not only by its original terms but also by the terms that are related to each of them. For example, if a document is indexed with only the first element of the term index, and the second term of the index is highly related to the first one, then the second component in the *mapped* document vector will be increased from zero to a positive value. In a sense, this amounts to performing a semantic smoothing of the original features via document expansion.

Note also [16] that the the linear transformation expressed by the mapping $\phi(d) = Pd$ can be used to redefine a gaussian kernel according to the semantic smoothing, i.e.:

$$K(d_1, d_2) = exp\ (-\ \gamma\ ||(d_1 - d_2)^T\ P^T\ P\ (d_1 - d_2)||^2) \tag{5}$$

By varying the matrix P one can obtain different transformations of the document feature space, which can be traced back to various document representation models. In the next section we describe a term similarity matrix based on the conceptual proximity between terms in a document lattice.

3 A kernel based on document lattice

Let D be the set of (training) documents, T the set of terms describing the documents, and TD the term by document matrix. Consider the ordered set $(\mathcal{C}(D, T, TD);\ \succ\prec)$ formed by the set of concepts of the context (D, T, TD) along with the nearest neighbour relation $(\succ\prec)$, i.e., for $x, y \in \mathcal{C}(D, T, TD)$, $x \succ\prec y$ if $x \succ y$ or $y \succ x$. Define the *concept distance* between concepts x and y as the least $n \in \mathcal{N}$ for which the following condition holds:

$$\exists z_0,\ z_1,\ \ldots,\ z_n\ \in\ \mathcal{C}(D, T, I);\ \succ\prec) \text{ such that } x = z_0 \succ\prec z_1 \ldots \succ\prec z_n\ =\ y.$$

Consider now two terms $(t_1, t_2), t_i \in T$. The *term distance* between t_1 and t_2 is given by the concept distance (as defined above) between the two corresponding term concepts (t_1', t_1'') and (t_2', t_2''), expressed by the standard prime and double prime operators.

Note that we remove the top and the bottom elements of the document lattice before computing the pairwise term proximities, because such concepts do not have a real meaning (even when the intent of the top concept is not empty it does not bear any information) and they may short-circuit conceptually distant concepts. The top element of the lattice is especially critical because term concepts are typically co-atoms.

The found relations have an intuitive meaning in terms of the properties of near concepts on the document lattice. A zero term distance means that the two terms always occur together in the documents (i.e., their mutual information is maximum). Distance equal to 1 means that there is a term a in the pair that always co-occurs with the other term b (i.e., the conditional probability of a given b is equal to 1). Distance 2 means that *either* term a always co-occurs with term b and there is some other term c that co-occurs with b more frequently than a (i.e., when a is a nephew of b), *or* (when they have a child in common) that a and b co-occur in one or more documents and there is no other term that co-occurs with either a or b in a superset of such documents. And so on.

The proximity between two terms is inversely related to their distance. In order to find the proximity matrix P, we normalize the distance values by dividing by their maximum and then we subtract the normalized values from 1.

As an illustration, consider the simple context for vertebrate animals in Table 1; each row can be seen as a document and each column as an indexing term (possibly formed by multiple words). We first show in Table 2 the bag-of-word kernel matrix, computed with equation 2. The kernel values shown in Table 2 have been normalized to take into account the different length of documents, by using: $\overline{K}(d_1, d_2) = \frac{d_1^T d_2}{||d_1||\ ||d_2||}$. Note that when two animals do not have any property in common their kernel value is always equal to 0.

		breathes in water (a)	can fly (b)	has beak (c)	has hands (d)	has wings (e)	lives in water (f)	vivipar-ous (g)	produces light (h)
1	Bat		x			x		x	
2	Eagle		x	x		x			
3	Monkey				x			x	
4	Parrot fish	x		x			x		
5	Penguin			x		x	x		
6	Shark	x					x		
7	Lantern fish	x					x		x

Table 1. A context for vertebrate animals

	1	2	3	4	5	6	7
1	1	0,666667	0,408248	0	0,333333	0	0
2		1	0	0,333333	0,666667	0	0
3			1	0	0	0	0
4				1	0,666667	0,816497	0,666667
5					1	0,408248	0,333333
6						1	0,816497
7							1

Table 2. Bag-of-words kernel matrix for the context in Table 1 (the matrix is symmetric).

We now turn to the illustration of the concept lattice-based kernel. We show in Figure 1 the concept lattice built from the context in Table 1, and we report in Table 3 the pairwise term distances derived from the concept lattice in Figure 1 after the removal of its top and bottom element. For instance, the distance between terms (g) and (e) is equal to 3, because the shortest connecting path is: (1 3, g) $\succ\prec$ (1, b e g) $\succ\prec$ (1 2, b e) $\succ\prec$ (1 2 5, e).

The proximity matrix is shown in Table 4. The found conceptual proximities are meaningful. For instance, the most related terms are: (e) and (b) (i.e., 'has wings' and 'can fly'), (f) and (a) (i.e., 'lives in water' and 'breathes in water'), (g) and (d) (i.e., 'viviparous' and 'has hands'), (h) and (a) (i.e., 'produces light'

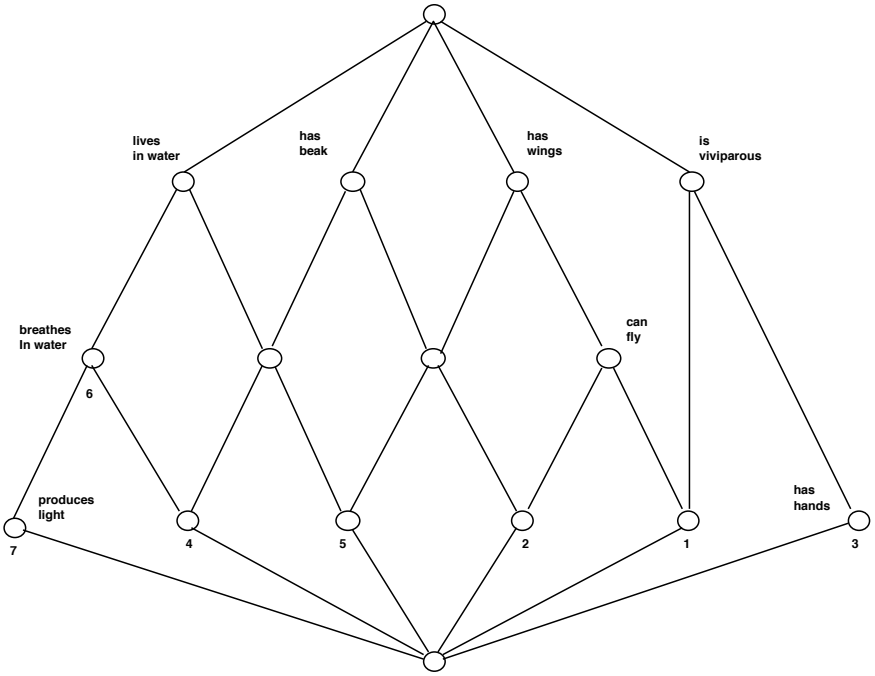

Fig. 1. Concept lattice for the context of Table 1, with minimal labelling.

	a	b	c	d	e	f	g	h
a	0	6	3	9	5	1	8	1
b		0	3	3	1	5	2	7
c			0	6	2	2	5	4
d				0	4	8	1	10
e					0	4	3	6
f						0	7	2
g							0	9
h								0

Table 3. Distances between pairs of the attributes in Table 1, derived from the concept lattice in Figure 1 (the matrix is symmetric).

	a	b	c	d	e	f	g	h
a	1	0.4	0.7	0.1	0.5	0.9	0.2	0.9
b		1	0.7	0.7	0.9	0.5	0.8	0.3
c			1	0.4	0.8	0.8	0.5	0.6
d				1	0.6	0.2	0.9	0
e					1	0.6	0.7	0.4
f						1	0.3	0.8
g							1	0.1
h								1

Table 4. Pairwise term proximities derived from corresponding distances in Table 3 (the matrix is symmetric).

and 'breathes in water'); note that the last relation is due to the fact that the only animal in the given context who produces light is a fish. The most unrelated terms are: (d) and (a) (i.e., 'has hands' and 'breathes in water'), and (h) and (g) (i.e., 'produces light' and 'viviparous').

Turning to pairwise document similarity, the concept lattice-based kernel matrix for the animal context can be computed with equation 4 using the proximity matrix in Table 4. The normalized kernel values are shown in Table 5.

	1	2	3	4	5	6	7
1	1	0,978114	0,95642	0,809072	0,914123	0,740581	0,704969
2		1	0,878241	0,908368	0,976945	0,855411	0,827088
3			1	0,627978	0,769209	0,545051	0,502219
4				1	0,976284	0,993071	0,98505
5					1	0,944986	0,92595
6						1	0,998065
7							1

Table 5. Concept lattice-based kernel matrix for the context in Table 1 (the matrix is symmetric).

Due to the implicit relationships between terms, two documents may have a varying degree of similarity even when they share the same number of terms (e.g., according to Table 5, 'bat' is more similar to 'eagle' than to 'penguin', although 'bat' has two terms in common with both 'eagle' and 'penguin'), or if they do not share any term (e.g., 'eagle' is more similar to 'monkey' than to 'shark').

A less obvious result is that the pairwise similarities in Table 5 may be relatively high even when the two documents are apparently very different (such as with 'eagle' and 'lantern fish') and that most values are distributed in the upper part of the range. This is probably due to the characteristics of this particular context, because each animal does not neatly fit in one class and shares resemblances to animals in other classes. Indeed, this example was originally given to illustrate the fact that several reasonable clusters can be formed out of a set of vertebrates in addition to the standard vertebrates groups, and thus it is probably not the best choice as an example for a classification task. On the other hand, this observation suggests that more analysis is needed to check whether the high similarity values may also be attributed to the specific notion of proximity used in our approach, which may be overly unconstrained. This issue deserves more attention and is left for future work, as alternative criteria are available to measure the distance between concepts in a concept lattice (see Section 6.2).

Before concluding this section we would like to point out that a different approach to finding a concept lattice-based kernel is conceivable. The key idea is that a document lattice can be used to find a conceptual similarity exactly at the document level, thus ignoring the similarities between the single terms describing

the documents. One could merge the document to be classified on the concept lattice of training documents, and then compute the distance between the found concept and each training document concept. This approach is appealing since, unlike the concept lattice-based kernel illustrated above, it allows to compute the similarity between documents directly in the transformed space, without performing the linear transformation $\phi(d) = Pd$. In addition, it would have a nice interpretation from the point of view of document similarity, because the distance in the document lattice can be thought of as the minimal number of admissible intent changes – with respect to the collection at hand – that lead from the description of one document to the description of another document (see [3]).

The main disadvantage of such a similarity measure is that it is not guaranteed that the kernel is valid. On the other hand, its properties (e.g., it is symmetric, has zero diagonal, is nonnegative, obeys the triangle inequality) do not allow us to rule out such a possibility. Note also that good classification is possible despite indefinite kernel matrices, because it has been proved [9] that SVMs still solve a data separation problem (although the solution may be only a local optimum). Experimenting with this approach is an avenue for future research.

4 Full method description

In this section we describe the main steps involved in our implementation of a full SVM text classification system with concept lattice-based kernel.

4.1 Text pre-processing

The aim of text pre-processing is to transform each document into a sequence of *features* that will be used in subsequent steps. The input documents go through text segmentation, punctuation removal, conversion of upper to lower case, and stop-word removal. We use strict single-word indexing and do not not perform any stemming.

4.2 Selection of input features

The features extracted from each document are reduced through an explicit feature selection phase. Working with a restricted set of features improves the overall system efficiency and it may also be useful for increasing classification accuracy, because index terms with low predictive power usually add noise. The selection criterion used in our system is mutual information. Both the text pre-processing and feature selection steps were performed using the Bow (or libbow) toolkit, available at: www.cs.cmu.edu/ mccallum/bow/.

4.3 Construction of document lattice

Once each document has been represented with a set of predictive features, we partition the entire dataset in training and test sets, and construct the concept lattice associated with the training set. To build the document lattice, we use the NextNeighbors algorithm, described in [4] on page 35.

4.4 Pairwise term proximity

To compute the proximity matrix P we need to find the pairwise term distances. This problem is solved by using a free package based on Dijkstra's algorithm to find the shortest path between two nodes in a graph. We apply Dijkstra's algorithm to any pair of term concepts in the document lattice, after removing the top and the bottom elements. If the term concept corresponding to a given term becomes disconnected (which rarely happened in our experiments), then this is signaled by the program and we assign a zero proximity value to all the pairs in which the disconnected term is involved. The computation of the pairwise term distances is the most critical step from a computational point of view, because the running time of Dijkstra algorithm for any pair of term concepts is $O(n\,logn)$, where n is the number of concepts in the lattice, and this operation must be repeated for $\frac{k(k-1)}{2}$ times, where k is the number of attributes.

4.5 Kernel matrix

Using the proximity matrix found in the earlier step, the kernel matrix is computed with equation 4 (or equation 5 for the gaussian kernel). To take into account the relative importance of terms in the documents, we represent the two document vectors d_1 and d_2 as vectors of weighted terms, using the classical *tf-idf* scheme (i.e., term frequency times inverse document frequency) to assign a weight to each term in each document. The same weights were also used, in the experiments described below, to find the bag-of-words kernels using equations 2 and 3 .

4.6 SVM classification

To perform the SVM classification with the various kernels, we used the LIBSVM package, available at www.csie.ntu.edu.tw/ cjlin/libsvm/.

5 Experiments

The aim was to compare the concept lattice-based kernel (CL) to the traditional bag-of-words (BOW) kernel, when used within a SVM learning algorithm. The complexity of our method does not allow us to use high-dimensional data, on which the standard SVM has proved to perform well. So we decided to focus on small datasets, containing little training data.

We hypothesized that the reliance of the standard SVM on the input feature vectors can be questioned when there are not sufficient training data, because there is a higher chance that the terms in the test documents will not be matched, and thus it may be more difficult to identify the regions of the feature space that belong to each class. We ran two experiments. In the first, we considered datasets of varying content and size, but with the same proportion of test and training documents; in the second we kept the test set fixed and let the training set decrease until its size became a small fraction of the test set size. We now describe each experiment in detail.

For the first experiment we used the 20 NewsGroup (20NG) collection, available at http://people.csail.mit.edu/jrennie/20Newsgroups/, which contains 20,000 Usenet articles partitioned in 20 thematic categories. After selecting the eight most different categories (i.e., *Atheism, Computer Graphics, Misc Forsale, Autos, Sport Baseball, Medicine, Talk Religions, and Talk Politics*), we randomly chose four samples of different size (i.e., 5, 10, 15, 20, 25) from each category and merged the documents of each category contained in the corresponding samples. We thus produced five datasets of varying content and size, each containing the same number of documents for each category.

We evaluated the performance of the two document similarity metrics on each dataset. We randomly split each dataset in two halves , each with a proportional number of documents per category. We used one half for training and the other half for testing, measuring the accuracy of each method (i.e., the percentage of test documents that were correctly classified). All runs were performed with a restricted term index containing the 200 words with the highest mutual information value. We experimented with both linear kernels (equations 2 and 4) and gaussian kernels (equations 3 and 5, with $\gamma = 0.001$), so we tested four methods in all, denoted by *linear CL-SVM*, *linear BOW-SVM*, *gaussian CL-SVM*, and *gaussian BOW-SVM*. The results are shown in Table 6.

	40 docs	80 docs	120 docs	160 docs	200 docs
linear BOW-SVM	69.23	63.75	67.27	68.57	69.56
linear CL-SVM	71.79	65.00	69.09	76.42	72.28
gaussian BOW-SVM	69.23	66.25	67.27	70.00	70.65
gaussian CL-SVM	71.79	66.25	70.00	77.14	72.28

Table 6. Accuracy on small subsets of the 20NG collection with two random split of the data.

The main finding is that CL-SVM outperformed BOW-SVM for any comparable pairs of data points, i.e., the superiority of CL-SVM over BOW-SVM held for both the linear and the gaussian versions and across all five datasets. The results in Table 6 also show that the accuracy of the gaussian kernel was only slightly greater or more often just equal to that of the corresponding linear kernel.

To take into account the effect of the feature selection step on performance, we varied the value of the threshold used to select the terms with the highest mutual information scores. We repeated the tests over the same datasets with indexes ranging from few tens to several hundreds of words. We did not notice a significant change in performance, provided that the index size did not become very small. Even the relative performance of the methods was substantially stable across indexes of various size.

For the second experiment, we used the mini 20 NewsGroup collection, which is a reduced version of the full 20NG dataset. It consists of the same set of 20NG categories, but each category contains only 100 articles. To simulate critical learning conditions, we were interested in evaluating what happens when the set of documents used to learn the classifier becomes increasingly smaller. After selecting the same eight categories as in the first experiment, we randomly chose 80 documents from each category and merged the documents of each category, thus forming a dataset with 640 documents. This was the test set, kept constant throughout the experiment. The 20 remaining documents of each category were used to form several training sets. The largest had just all the 160 documents; then we created a smaller training set by randomly removing from it half of the documents in each category, thus producing a set with 80 documents (i.e., 10 per category). We iterated the same procedure two times, thus producing in all three training sets such that they could be fordered according to the set inclusion relation.

We then evaluated the performance of the two document similarity metrics on the three datasets formed by merging each training set with the same test set containing 640 documents. The results are shown in Table 7.

	160 training docs	80 training docs	40 training docs
linear BOW-SVM	58,516	59,8651	58,516
linear CL-SVM	68,1282	65,43	64,5868
gaussian BOW-SVM	61,8887	59,1906	58,516
gaussian CL-SVM	62,3946	62,3946	60,7083

Table 7. Accuracy on a test set of 640 documents of the mini 20NG collection with increasingly smaller training sets.

The main result is that the conceptual method performed better than the standard method across all kernels and datasets. This is consistent with the findings of the first experiment. The benefits of the conceptual method over the standard method are more evident with a linear kernel, although even with a gaussian kernel the conceptual method always achieved higher accuracy. As the size of the training set decreased, the performance of each method decreased very gently, while the gain in performance due to the use of the conceptual method did not grow. One possible explanation for the latter phenomenon is that with very few training documents there is less chance to recover from bad

document expansion, which may more easily result in topic drift. Also, the use of the gaussian kernel together with the conceptual document similarity metric seemed to hurt performance. Here one should consider that a combination of these techniques may not work well, because both ultimately produce the same effect, namely an expansion of the terms describing each document.

6 Related Work

There are three main areas related to this work, which are described in turn below.

6.1 Semantic kernels

One of the first semantic kernels for SVM text classification was proposed in [16]. A term similarity function based on WordNet was used, with the similarity between two terms being inversely proportional to the length of the path linking the two nodes. Such similarities were used to perform a semantic smoothing of the input feature vectors, similar to our approach. More recently [1], WordNet has been used to compute a similarity function between terms based on the sub-hierarchy rooted in their lowest common hypernym. The similarity function between two documents was then defined as the sum of the similarities of all pairs of terms in common to the documents. WordNet is a valuable source of semantic information that can improve text representation, but its use for document expansion is made difficult by several reasons, including its limited coverage, the need for sense disambiguation, and the lack of proper nouns. Another issue is the computational complexity of these algorithms. They require the evaluation of all the term pairs, for each of which it is necessary to navigate the WordNet hierarchy. Our method suffers from similar computational limitations.

Another approach to semantic kernels for text is described in [6]. Inspired by latent semantic indexing [7], which consists of projecting the data into a subspace through a singular value decomposition of the term by document matrix, Cristianini et al. showed that it is possible to apply the same technique to any kernel defined feature space. This approach is, by nature, more similar to ours because it exploits interdocument relationships, but the similarity between documents is based on extracting a subset of invariant features from the input data, rather than expanding the original feature vectors. The main limitations of latent semantic kernels are its high time complexity (i.e., to compute the eigenvalue decomposition of the kernel matrix) and the high number of dimensions required to draw relevant features from the training data. Also, the resulting features might be difficult to interpret.

6.2 Concept similarity

In this paper we have adopted a simple term similarity measure based on edge counting, i.e, the shorter the path between two term concepts, the more related

the terms associated with them. Belohlavek [2] used a similar, more restricted notion in which two attributes are similar if the distance between their corresponding attribute concepts is equal to zero. The edge counting approach takes into account the structural relationships between the subsets of attributes in the collection, but it has the disadvantage that it ignores the specific description of the nodes.

The similarity between formal concepts has been studied in a few recent papers from other perspectives. Formica [8] presents a more comprehensive method where the similarity between the intents of the concepts is explicitly considered. The overall concept similarity is obtained by combining the cardinality of the intersection of the concept extents with the similarity of the *information content* [11] of the two concept intents. The information content of a concept intent *int* is given by $-\ log\ p(int)$, and the information content similarity of two concept intents is computed as the ratio of the information content of the least upper bound of the two concepts to the sum of the information content of each concept intent. Another recent method that combines structural and featural information is given in [18]. It is based on the rough set theory and considers only the join-irreducible and meet-irreducible elements of the lattice to express the similarity between concepts.

6.3 Applications of document lattices

In the last years, a number of information retrieval applications based on document lattices have been proposed. A detailed account of the issues involved in developing concept lattice-based information retrieval applications along with a survey of approaches and systems are given in [4], spanning integration of thesauri, query refinement, text ranking, and search results clustering. Other notable applications include faceted browsing [5], and query elicitation [13]. Uta Priss [14] offers a somewhat complementary view, showing the applications of concept lattices to several subfields of information science including information retrieval.

The most similar earlier approach is [3], in which a user query is mapped on the concept lattice built from the collection being searched, and then a document ranking is automatically computed based on the topological distance between the query concept and the document concepts. This paper takes a step forward by further expanding the technical and practical scope of this research line to text classification.

7 Conclusions and future work

There are several ongoing attempts to extend kernel learning methods with semantic information. In this paper we have defined a concept lattice-based kernel for SVM text classification that makes use of term relations encoded in the document lattice associated with the training documents. The proposed method is formally sound and it has an intuitive meaning. Furthermore, in an experimental

evaluation performed on small datasets, it achieved higher classification accuracy than the standard SVM. Our main conclusion is that the exploitation of hidden, structural document relationships beyond the input description of each document may help build a better classifier, at least when little training data are available.

We plan to extend this research in several direction: (a) using the concept lattice-based document similarity metric within other learning algorithms (e.g., K-nearest neighbors), (b) experimenting with other collections, larger datasets, and more difficult learning conditions (e.g., imbalanced data), (c) using different criteria to measure the distance between concepts in the document lattice, (d) investigating the potentials of the method based on measuring the distance between documents (rather than terms) directly in the document lattice.

References

1. R. Basili, M. Cammisa, and A. Moschitti. A semantic kernel to classify texts with very few training examples. *Informatica*, 30:163–172, 2006.
2. R. Belohlavek. Similarity relations in concept lattices. *Journal of Logic and Computation*, 10(6):823–845, 2000.
3. C. Carpineto and G. Romano. Order-Theoretical Ranking. *Journal of the American Society for Information Science*, 51(7):587–601, 2000.
4. C. Carpineto and G. Romano. *Concept Data Analysis — Theory and Applications*. Wiley, 2004.
5. R. Cole, P. Eklund, and G. Stumme. Document retrieval for e-mail search and discovery using formal concept analysis. *Applied Artificial Intelligence*, 17(3):257–280, 2003.
6. N. Cristianini, J. Shawe-Taylor, and H. Lodhi. Latent semantic kernels. *Journal of Intelligent Information Systems*, 18(2–3):127–152, 2002.
7. S. Deerwester, S. T. Dumais, W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.
8. A. Formica. Concept similarity in formal concept analysis: An information content approach. *Knowledge-Based Systems*, 21(1):80–87, 2008.
9. B. Haasdonk. Feature space interpretation of svms with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–492, 2005.
10. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML-98)*, LNCS 1398, pages 137–142, Chemnitz, Germany, 1998. Springer.
11. D. Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, pages 296–304, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.
12. C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
13. C. Meghini and N. Spyratos. Computing intensions of digital library collections. In *Proceedings of the the 5th International Conference on Formal Concept Analysis (ICFCA 2007)*, LNCS 4390, pages 66–81, Clermont-Ferrand, France, 2007. Springer.

14. U. Priss. Formal concept analysis in information science. *Annual Review of Information Science and Technology (ARIST)*, 40, 2006.
15. H. Schölkopf and Smola A. J. *Learning with Kernels*. MIT Press, 2002.
16. G. Siolas and F. d'Alche Buc. Support vector machines based on a semantic kernel for text categorization. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00) - Volume 5*, pages 205–209, 2000.
17. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
18. L. Wang and X. Liu. A new model of evaluating concept similarity. *Knowledge-Based Systems*, 21(4):842–846, 2008.