Alexandros Karatzoglou and Ingo Feinerer

# Text Clustering with String Kernels in R

Working Paper

# Text Clustering with String Kernels in R

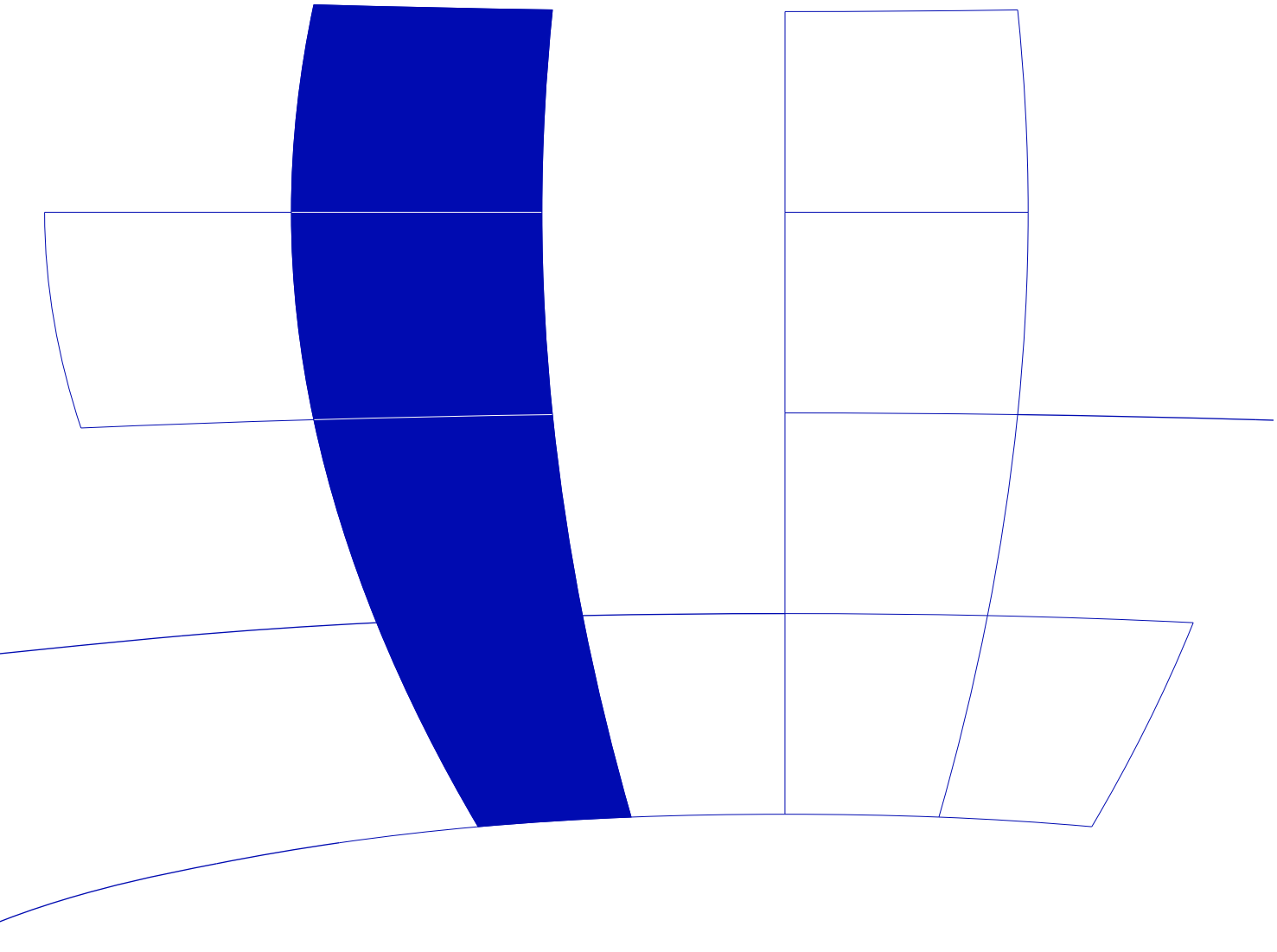
**Alexandros Karatzoglou, Ingo Feinerer**

Department of Statistics and Mathematics
Wirtschaftsuniversität Wien

**Research Report Series**

Report 34
May 2006

http://statmath.wu-wien.ac.at/



# Text clustering with string kernels in R

Alexandros Karatzoglou
Department of Statistics and Probability Theory,
Technische Universität Wien, A-1040 Wien, Austria
alexis@ci.tuwien.ac.at

Ingo Feinerer
Department of Statistics and Mathematics,
Wirtschaftsuniversität Wien, A-1090 Wien, Austria
h0125130@wu-wien.ac.at

May 6, 2006

**Abstract**

We present a package which provides a general framework, including tools and algorithms, for text mining in R using the S4 class system. Using this package and the `kernlab` R package we explore the use of kernel methods for clustering (e.g., kernel $k$-means and spectral clustering) on a set of text documents, using string kernels. We compare these methods to a more traditional clustering technique like $k$-means on a bag of word representation of the text and evaluate the viability of kernel-based methods as a text clustering technique.

## 1 Introduction

The application of machine learning techniques to large collections of text documents is a major research area with many application such as document filtering and ranking. Kernel-based methods have been shown to perform rather well in this area, particularly in text classification with SVM using either a simple "bag of words" representation (i.e. term frequencies with various normalizations) [6], or more sophisticated approaches like string kernels [11], or word-sequence kernels [1]. Despite the good performance of kernel methods in classification of text documents, little has been done on the field of clustering text documents with kernel-based methods.

## 2 Software

R [13] is a natural choice for a text mining environment. Besides the basic string and character processing functions it includes an abundance of statistical analysis functions and packages and provides a Machine Learning task view with a wide range of software.

## 2.1 `textmin` R Package

The `textmin` package provides a framework for text mining applications within R. It fully supports the new S4 class system and integrates seamlessly into the R architecture.

The basic framework classes for handling text documents are:

**TextDocument:** Encapsulates a text document, irrelevant from its origin, in one class. Several slots are available for additional meta data, like an unique identification number or a description.

**TextDocumentCollection:** Represents a collection of text documents. The constructor provides import facilities for common data formats in textmining applications, like the Reuters21578 news format or the Reuters Corpus Volume 1 format.

**TermDocumentMatrix:** Stands for a term-document matrix with documents (in fact their id numbers) as rows and terms as columns. Such a term-document matrix can be easily built from a text document collection. A bunch of weighting schemes are available, like binary, term frequency or term frequency inverse document frequency. This class can be used as a fast representation for all kinds of bag-of-words textmining algorithms.

## 2.2 `kernlab` R Package

Kernel-based learning methods, like kernel $k$-means, use an implicit mapping of the input data into a high dimensional feature space defined by a kernel function, i.e., a function returning the inner product $\langle \Phi(x), \Phi(y) \rangle$ between the images of two data points $x, y$ in the feature space. The learning then takes place in the feature space, provided the learning algorithm can be entirely rewritten so that the data points only appear inside dot products with other points. This is often referred to as the "kernel trick" [14]. More precisely, if a projection $\Phi : X \to H$ is used, the inner product $\langle \Phi(x), \Phi(y) \rangle$ can be represented by a kernel function $k$

$$k(x, y) = \langle \Phi(x), \Phi(y) \rangle, \tag{1}$$

which is computationally simpler than explicitly projecting $x$ and $y$ into the feature space $H$. Once a valid kernel function has been selected, one can practically work in spaces of any dimension without paying any computational cost, since feature mapping is never effectively performed. One can also design and use a kernel for a particular problem that could be applied directly to the data without the need for a feature extraction process. This is particularly important in problems where a lot of structure of the data is lost by the feature extraction process as is the case in text processing. The inherent modularity of kernel-based learning methods allows one to use any valid kernel on a kernel-based algorithm. `kernlab` is an extensible package for kernel-based machine learning methods in R. It takes advantage of R's new S4 object model and provides a framework for creating and using kernel-based algorithms. The package contains implementations of most popular kernels, and also inludes a range of kernel methods for classification, regression (support vector machine, relevance vector machine) clustering (kernel $k$-means, spectral clustering), ranking, and principal component analysis. Moreover it provides a general purpose quadratic

programming solver `ipop`, and methods for computing a incomplete Cholesky decomposition. `kernlab` also includes methods for computing commonly used kernel expressions (e.g., the kernel matrix) and allows the user to define and easily use a kernel with any of the existing methods in the package.

# 3 Methods

The $k$-means clustering algorithm is one of the most commonly used clustering methods providing solid results but also having some drawbacks. Denoting clusters by $\pi_j$ and a partitioning of points as ${\pi_j}_{j=1}^k$ the $k$-means objective function using Euclidean distances becomes:

$$D({\pi_j}_{j=1}^k) = \sum_{j=1}^{k} \sum_{a \in \pi_j} \|a - m_j\|^2 \tag{2}$$

$$\text{where} \quad m_j = \frac{1}{\|\pi_j\|} \sum_{a \in \pi_j} a \tag{3}$$

A major drawback of $k$-means is that it cannot separate clusters that are not linearly separable in input space.

## 3.1 Kernel $k$-means

One technique for dealing with this problem is mapping the data into a high-dimensional non-linear feature space with the use of a kernel. Kernel $k$-means uses a kernel function to compute the inner product of the data in the feature space. All computations are then expressed in terms of inner products thus allowing the implicit mapping of the data into this feature space. If $\Phi$ is the mapping function then the $k$-means objective function using Euclidean distances becomes:

$$\mathcal{D}({\pi_j}_{j=1}^k) = \sum_{j=1}^{k} \sum_{a \in \pi_j} \|\Phi(a) - m_j\|^2 \tag{4}$$

$$\text{where} \quad m_j = \frac{1}{\|\pi_j\|} \sum_{a \in \pi_j} \Phi(a) \tag{5}$$

in the expansion of the square norm only inner products of the form $\langle \Phi(a), \Phi(b) \rangle$ appear which are computed by the kernel function $k(a, b)$.

The implementation of kernel $k$-means included in `kernlab` makes use of the triangle inequality [3] in order to avoid unnecessary and computational expensive distance calculations. This leads to significant speedup particularly on large data sets with a high number of clusters.

## 3.2 Spectral Clustering

Spectral clustering [12], [15] works by embedding the data points of the partitioning problem into the subspace of the $k$ largest eigenvectors of a normalized affinity matrix. The use of an affinity matrix also brings one the advantages of kernel methods to spectral clustering, since one can define a suitable affinity

for a given application. For example if the feature vectors represent color histograms simple $k$-means clustering is inappropriate since an $L_2$ distance between histograms is not meaningful. In such a case one can employ a suitable affinity function such as the $\chi^2$-distance. In our case we use a string kernel to define the affinities between two documents and construct the kernel matrix. The data is then embedded into the subspace of the largest eigenvectors of the normalized kernel matrix. This embedding usually leads to more straightforward clustering problems since points tend to form tight clusters in the eigenvector subspace. Using a simple clustering method like $k$-means on the embedded points usually leads to good performance. It can be shown that most spectral clustering methods boil down to a a graph partitioning problem [2] that can be solved by a weighted kernel $k$-means algorithm.

### 3.3 String kernels

String kernels [17], [5] are defined as a similarity measure between two sequences of characters $x$ and $x'$. The generic form of string kernels is given by the equation:

$$k(x, x') = \sum_{s \sqsubseteq x, s' \sqsubseteq x'} \lambda_s \delta_{s,s'} = \sum_{s \in A^*} \mathrm{num}_s(x) \mathrm{num}_s(x') \lambda_s \tag{6}$$

where $A^*$ represents the set of all non empty strings and $\lambda_s$ is a weight or decay factor which can be chosen to be fixed for all substrings or can be set to a different value for each substring. This generic representation includes a large number of special cases, e.g. setting $\lambda_s \neq 0$ only for substrings that start and end with a white space character gives the "bag of words" kernel [7]. In this paper we will focus on the case where $\lambda_s = 0$ for all $|s| > n$ that is comparing all substrings of length less that $n$, this kernel will be referred to in the rest of the paper as full string kernel. We also consider the case where $\lambda_s = 0$ for all $|s| \neq n$ which we referred to as the string kernel. The computational complexity of the string kernels we consider is $O(\mathrm{n}, |x|, |x'|)$.

## 4 Experiments

We will now compare the performance of the various clustering techniques on text data by running a series of experiments on the well known Reuters text data set.

### 4.1 Data

The Reuters-21578 dataset [10] contains stories for the Reuters news agency. It was compiled by David Lewis in 1987, is publicly available and is currently one of the most widely used datasets for text categorization research. A Reuters category can contain as few as 1 or as many as 2877 documents. In our experiments we used a subset of the Reuters dataset so that the computation of a full kernel matrix in memory was not a concern. We used the "crude" which contains about 580 documents, the "corn" category which includes 280 documents and a sample of 1100 documents from the "acq" category. Our dataset thus consists of 1720 documents after preprocessing:

We removed the stop words that occur in a stop list and any empty documents and convert all characters to lower case. We also removed punctuation and white space and performed stemming on the documents using the `Rstem` [8] `omegahat` R package.

## 4.2 Experimental Setup

We perform clustering on the dataset using the kernel $k$-means and spectral clustering methods in the `kernlab` package and the $k$-means method in R. For the kernel $k$-means and spectral methods we also use the string kernels implementations provided in `kernlab`. In order to learn more about the effect of the string kernels hyper-parameters on the clustering results we run the clustering algorithms over a range of the length parameter $n$ which controls the length of the strings compared in the two character sets and the decay factor $\lambda$. We study the effects of the parameters by keeping the value of the decay parameter $\lambda$ fixed and varying the length parameter. Note that for each parameter set a new kernel matrix containing different information has to be computed.

We use values from $n = 3$ to $n = 14$ for the length parameter and $\lambda = 0.2$, $\lambda = 0.5$ and $\lambda = 0.8$ for the decay factor. We also use both the string (or spectral) and the full string kernel and normalize in order to remove any bias introduced by document length. We thus use a new embedding $\hat{\phi} = \frac{\phi(s)}{\|\phi(s)\|}$ which gives rise to the kernel:

$$\hat{K}(s, s') = \langle \hat{\phi}(s), \hat{\phi}(s') \rangle = \left\langle \frac{\phi(s)}{\|\phi(s)\|} \frac{\phi(s')}{\|\phi(s')\|} \right\rangle = \tag{7}$$

$$\frac{\langle \phi(s), \phi(s') \rangle}{\|\phi(s)\| \|\phi(s')\|} = \frac{K(s, s')}{\sqrt{K(s, s) K(s', s')}} \tag{8}$$

For the classical $k$-means method we create a term document matrix of the term frequencies and also an inverse term frequencies matrix.

## 4.3 Performance measure

We evaluate the performance of the various clustering techniques using the recall rate which is a typical measure for evaluating the performance of a clustering algorithm when the actual labels of the clustered data are known. Given a discovered cluster $\gamma$ and the associated reference cluster $\Gamma$, recall $R$ is defined as in:

$$R = \frac{\sum_{\Gamma=1}^{k} n_{\gamma\Gamma}}{\sum_{\Gamma=1}^{k} N_{\Gamma}} \tag{9}$$

where $n_{\gamma\Gamma}$ is the number of documents from reference cluster $\Gamma$ assigned to cluster $\gamma$, $N_{\Gamma}$ is the total number of documents in cluster $\gamma$ and $N_{\Gamma}$ is the total number of documents in reference cluster $\Gamma$.

## 4.4 Results

The main goal of these experiments is to establish if kernel methods along with string kernels are a valiable solution for grouping a set of text documents.

From the experiments we run it became obvious that the $\lambda$ parameter influences the performance only minimally and thus we chose to look at the results

| kernel matrix calculations | $\approx$ 2 h. |
|---|---|
| spectral clustering | $\approx$ 20 sec. |
| kernel k-means | $\approx$ 30 sec. |
| term matrix k-means | $\approx$ 40 sec. |

Table 1: Timings for the clustering methods and the computation of the kernel matrix.

in relation to the string length kernel parameter which seems to have a more profound influence on the performance of the kernel-based clustering methods. The performance of the $k$-means clustering method is also very similar with both the simple document matrix or the inverse frequency document matrix.

Figure 1 shows the average recall rate over 10 runs for the spectral clustering methods, and the kernel $k$-means method with the full string kernel compared to the reference recall rate of the inverse term document matrix clustered with a simple $k$-means algorithm. The plot shows that both the spectral method and kernel $k$-means fail to improve over the performance of the standard $k$-means clustering technique. We also note that the spectral clustering technique provides very stable results thous almost zero variance. This can be attributed to the fact that the projection of the data into the eigenspace groups the data into tight clusters which are easy to separate with a standard clustering technique.

Figure 2 displays the average recall rate of the kernel $k$-means with a string kernel along with the standard $k$-means clustering results. It is clear that for a range of values of the string length parameter the kernel $k$-kmeans functions outperforms $k$-means clustering and the full string kernel methods with a full string kernel. The method does not provide stable performance and the variance of the recall rate over the 10 runs seems quite high compared to the other methods.

Figure 3 shows the recall rate of the spectral clustering method with a string kernel averaged over 10 runs compared to the standard $k$-means clustering results. This is clearly the best performing clustering method for this set of text documents and also exhibits some interesting behavior. For rather small lengths of substrings considered $(3, 4, 5)$ the performance seems to increase monotonically and at the the value of 6 hits a threshold. For the range of values between 6 and 10 the performance increase is much smaller and for the value of 10 the highest recall rate of 0.927 is reached. For higher values of the length parameter the performance drops sharply only to increase again for a string length value of 14. Again this method is very stable and exhibits minimal variance.

## 4.5 Timing

We have also evaluated the methods in terms of running time. The experiments where run on a Linux machine with a 2.6 GHz Pentium 4 CPU. Table 1 provides the running time for the calculation of a full kernel matrix and the running time for the clustering methods. Note that the running time for the kernel-based clustering methods is the time needed to cluster data with a precomputed kernel matrix. From the results it is clear that most of the computing time is spend on the calculation of the kernel matrix.

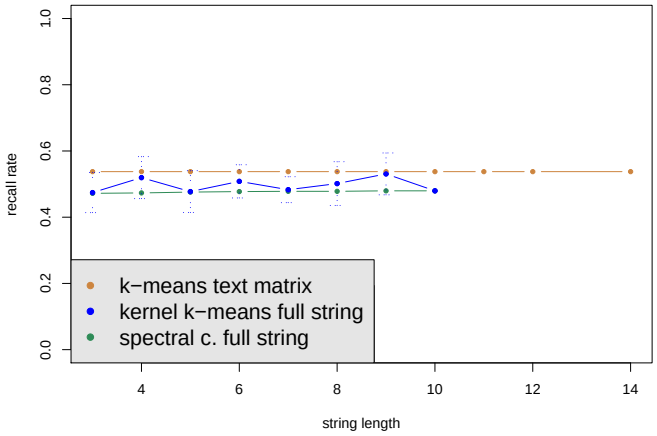



Figure 1: Average recall rate over 10 runs for the spectral clustering , kernel $k$-means, with full string kernels and $k$-means on a inverse frequencies term matrix methods. On the $y$ axis is the recall rate and the $x$ axis the string length hyper-parameter of the string kernel.

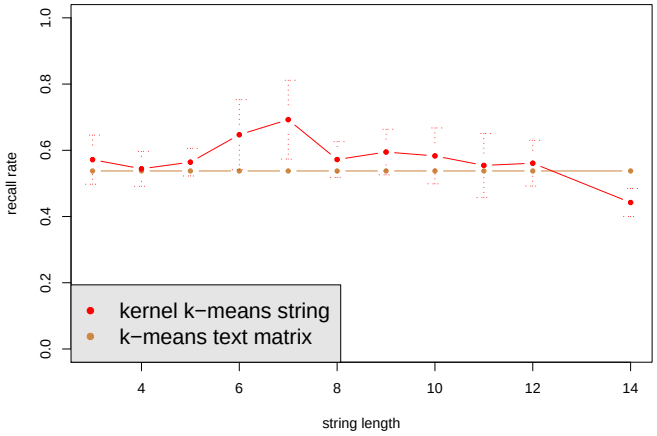



Figure 2: Average recall rate over 10 runs for the kernel $k$-means, with string/spectral kernels and $k$-means on a inverse frequencies term matrix methods. On the $y$ axis is the recall rate and the $x$ axis the string length hyper-parameter of the string kernel.

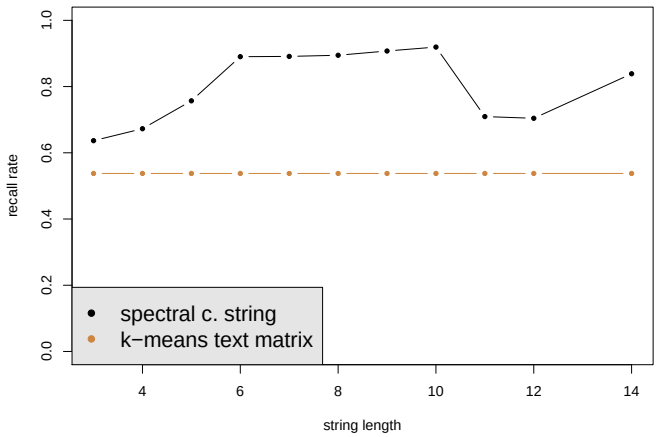



Figure 3: Average recall rate over 10 runs for the spectral clustering with a string/spectral kernel and the $k$-means on a inverse frequencies term matrix methods. The $x$ axis represents the string length hyper-parameter of the string kernel.

# 5 Conclusions

From the results it is clear that the spectral clustering technique combined with a string kernel outperforms all other methods and provides very strong performance even comparable to the classification performance of an SVM with a string kernel on similar dataset [11]. This is very encouraging and shows that kernel-based clustering methods can be considered as a viable text grouping method. The behavior of the kernel-based algorithms, particularly of the spectral clustering method, seem to strongly depend on the value of the string length parameter. It is an open question if the range of good values of this parameter $(6 - 10)$ on this dataset can be also used on other text datasets in the same or other languages to provide good performance. It is interesting to note that a string length of 6 to 10 characters corresponds to the size of one or two words in the English language. It would also be interesting to study the behavior of the method for string lengths higher than 14. The good performance of the spectral clustering technique could be an indication that graph partitioning methods combined with string kernels could provide good results on text clustering.

One drawback of the kernel based methods is the amount of time spend on the computation of the kernel matrix and, particularly for the spectral methods, the necessity to store a full $m \times m$ where $m$ the number of text documents, in memory. A suffix tree based implementation of the string kernels as in [16] combined with the Nystrom method [18] for computing the eigenvectors of the kernel matrix as in [4] by using only a sample of the data points could provide a solution to this issues. It would also be interesting to explore the application of some other types of string kernels on text clustering. Of particular interest would be the mismatch [9] kernels especially on raw (i.e. non pre-processed)

text data.

# References


[1] Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean-Michel Renders. Word-sequence kernels. *Journal of Machine Learning Research*, 3:1059–1082, 2003.

[2] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. A Unified view of Kernel k-means, Spectral Clustering and Graph Partitioning. Technical report, University of Texas at Austin, February 2005.

[3] Charles Elkan. Using the triangle inequality to accelerate $k$-means. *In Proceedings of the Twentieth International Conference on Machine Learning (ICML'03)*, pages 147–153, 2003.

[4] Charles Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nystrom method. *Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[5] Ralf Herbrich. *Learning Kernel Classifiers Theory and Algorithms*. Adaptive Computation and Machine Learning. The MIT Press, 2002.

[6] Thorsten Joachims. Making large-scale SVM learning practical. *In Advances in Kernel Methods — Support Vector Learning*, 1999.

[7] Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms*. The Kluwer International Series In Engineerig And Computer Science. Kluwer Academic Publishers, Boston, 2002.

[8] Duncan Temple Lang. *Rstem: Interface to Snowball implementation of Porter's word stemming algorithm.*, 2005. R package version 0.2-0.

[9] Christina Leslie, Eleazar Eskin, Adiel Cohen, Jason Weston, and William Stafford Noble. Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476, 2004.

[10] David Lewis. Reuters-21578 text categorization test collection, 1997.

[11] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *JMLR*, 2:419–444, 2002.

[12] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14, 2001.

[13] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005.

[14] Bernhard Schölkopf and Alex Smola. *Learning with Kernels*. MIT Press, 2002.

[15] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[16] S.V.N. Vishwanathan and A.J. Smola. Fast kernels for string and tree matching. In K. Tsuda, B. Schölkopf, and J.P. Vert, editors, *Kernels and Bioinformatics*, Cambridge, MA, 2004. MIT Press.

[17] C. Watkins. Dynamic alignment kernels. In A.J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39 – 50, Cambridge, MA, 2000. MIT Press.

[18] Christoper K. I. Williams and Matthias Seeger. Using the Nystrom method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682 – 688, Cambridge, MA, 2001. MIT Press.